




**TURUN
YLIOPISTO**
UNIVERSITY
OF TURKU

A large, stylized sunburst graphic in a lighter shade of teal, positioned on the left side of the cover. It has a dark teal central oval and radiating lines of varying lengths, creating a fan-like shape.

OPTIMIZING TEXT MINING METHODS FOR IMPROVING BIOMEDICAL NATURAL LANGUAGE PROCESSING

Farrokh Mehryary



**TURUN
YLIOPISTO**
UNIVERSITY
OF TURKU

OPTIMIZING TEXT MINING METHODS FOR IMPROVING BIOMEDICAL NATURAL LANGUAGE PROCESSING

Farrokh Mehryary

University of Turku

Faculty of Technology
Department of Computing
Computer Science
Doctoral Programme in Technology

Supervised by

Associate Professor Filip Ginter
TurkuNLP
Department of Computing
Faculty of Technology
University of Turku, Finland

Professor Tapio Salakoski
TurkuNLP
Department of Mathematics and
Statistics
Faculty of Science
University of Turku, Finland

Reviewed by

Dr. Fabio Rinaldi
Dalle Molle Institute for Artificial Intelli-
gence Research, Switzerland

Associate Professor Jin-Dong Kim
Database Center for Life Science
Research Organization of Information
and Systems, Japan

Opponent

Professor Udo Hahn
Jena University Language & Information
Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena
Germany

The originality of this publication has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

ISBN 978-951-29-8743-6 (PRINT)
ISBN 978-951-29-8744-3 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama, Turku, Finland, 2022

I dedicate this thesis to my parents, Abolhassan Mehryari and Farahdokht Mankoodehi, for their constant love, support and sacrifices throughout my life. Also, for the time that could have been spent with them and for them, but it wasn't, because I was doing my studies in Finland, far away from home. I am forever grateful for their love, encouragement and wisdom.

UNIVERSITY OF TURKU

Faculty of Technology

Department of Computing

Computer Science

MEHRYARY, FARROKH: Optimizing Text Mining Methods for
Improving Biomedical Natural Language Processing

Doctoral dissertation, 165 pp.

Doctoral Programme in Technology

December 2021

ABSTRACT

The overwhelming amount and the increasing rate of publication in the biomedical domain make it difficult for life sciences researchers to acquire and maintain all information that is necessary for their research. Pubmed (the primary citation database for the biomedical literature) currently contains over 21 million article abstracts and more than one million of them were published in 2020 alone.

Even though existing article databases provide capable keyword search services, typical everyday-life queries usually return thousands of relevant articles. For instance, a cancer research scientist may need to acquire a complete list of genes that interact with BRCA1 (breast cancer 1) gene. The PubMed keyword search for BRCA1 returns over 16,500 article abstracts, making manual inspection of the retrieved documents impractical. Missing even one of the interacting gene partners in this scenario may jeopardize successful development of a potential new drug or vaccine. Although manually curated databases of biomolecular interactions exist, they are usually not up-to-date and they require notable human effort to maintain. To summarize, new discoveries are constantly being shared within the community via scientific publishing, but unfortunately the probability of missing vital information for research in life sciences is increasing.

In response to this problem, the biomedical natural language processing (BioNLP) community of researchers has emerged and strives to assist life sciences researchers by building modern language processing and text mining tools that can be applied at large-scale and scan the whole publicly available literature and extract, classify, and aggregate the information found within, thus keeping life sciences researchers always up-to-date with the recent relevant discoveries and facilitating their research in numerous fields such as molecular biology, biomedical engineering, bioinformatics, genetics engineering and biochemistry.

My research has almost exclusively focused on biomedical relation and event extraction tasks. These foundational information extraction tasks deal with automatic detection of biological processes, interactions and relations described in the biomedical literature. Precisely speaking, biomedical relation and event extraction systems can scan through a vast amount of biomedical texts and automatically detect

and extract the semantic relations of biomedical named entities (e.g. genes, proteins, chemical compounds, and diseases). The structured outputs of such systems (i.e., the extracted relations or events) can be stored as relational databases or molecular interaction networks which can easily be queried, filtered, analyzed, visualized and integrated with other structured data sources. Extracting biomolecular interactions has always been the primary interest of BioNLP researcher because having knowledge about such interactions is crucially important in various research areas including precision medicine, drug discovery, drug repurposing, hypothesis generation, construction and curation of signaling pathways, and protein function and structure prediction.

State-of-the-art relation and event extraction methods are based on supervised machine learning, requiring manually annotated data for training. Manual annotation for the biomedical domain requires domain expertise and it is time-consuming. Hence, having minimal training data for building information extraction systems is a common case in the biomedical domain. This demands development of methods that can make the most out of available training data and this thesis gathers all my research efforts and contributions in that direction.

It is worth mentioning that biomedical natural language processing has undergone a revolution since I started my research in this field almost ten years ago. As a member of the BioNLP community, I have witnessed the emergence, improvement—and in some cases, the disappearance—of many methods, each pushing the performance of the best previous method one step further. I can broadly divide the last ten years into three periods. Once I started my research, feature-based methods that relied on heavy feature engineering were dominant and popular. Then, significant advancements in the hardware technology, as well as several breakthroughs in the algorithms and methods enabled machine learning practitioners to seriously utilize artificial neural networks for real-world applications. In this period, convolutional, recurrent, and attention-based neural network models became dominant and superior. Finally, the introduction of transformer-based language representation models such as BERT and GPT impacted the field and resulted in unprecedented performance improvements on many data sets. When reading this thesis, I demand the reader to take into account the course of history and judge the methods and results based on what could have been done in that particular period of the history.

TURUN YLIOPISTO

Teknillinen tiedekunta

Tietotekniikan laitos

Tietojenkäsittelytiede

MEHRVARY, FARROKH: Optimizing Text Mining Methods for Improving Biomedical Natural Language Processing

Väitöskirja, 165 s.

Teknologian tohtoriohjelma

Joulukuu 2021

TIIVISTELMÄ

Biolääketieteen alan tutkimusartikkeleiden määrä kasvaa valtavaa vauhtia. Tämän vuoksi alan tutkijoiden on vaikea hankkia ja ylläpitää kaikkea tutkimuksessa tarvittavaa tietoa. PubMed eli biolääketieteellisen tutkimuskirjallisuuden ensisijainen viitetaustietokanta sisältää tällä hetkellä yli 21 miljoonaa tutkimusartikkelin tiivistelmää. Pelkästään vuonna 2020 julkaistiin yli miljoona artikkelia.

Vaikka nykyiset tutkimustietokannat tarjoavat tehokkaita hakupalveluja, taval-
liset, jokapäiväiset haut palauttavat yleensä tuhansia relevantteja artikkeleita. Es-
imerkiksi syöpätutkija saattaa tarvita luettelon kaikista geeneistä, jotka ovat
vuorovaikutuksessa BRCA1-geenin (breast cancer 1) kanssa. PubMedin hakupalvelu
palauttaa hakusanalle BRCA1 yli 16500 artikkelitiivistelmää, joten haettujen asi-
akirjojen manuaalinen tarkastelu on epäkäytännöllistä. Yhdenkin vuorovaikutus-
geenin puuttuminen tässä skenaariossa voi vaarantaa mahdollisen uuden lääkkeen
tai rokotteen kehittämisen. Vaikka käsin kuratoituja biomolekyylien vuorovaiku-
tustietokantoja on olemassa, ne eivät yleensä ole ajan tasalla, ja niiden ylläpitäminen
vaatii huomattavan määrän ihmistyönä tehtävää ylläpitoa. Tämän vuoksi tiedot uu-
sista tutkimustuloksista jaetaankin tavallisesti tutkimusartikkeleissa. Hakupalvelujen
tehottomuuden vuoksi tiedon ja tutkimustulosten hukkumiselle kuitenkin kas-
vaa jatkuvasti.

Biolääketieteellisten tekstien louhintaan erikoistunut BioNLP-tutkimusala on ke-
hittynyt vastaamaan tähän haasteeseen. BioNLP pyrkii auttamaan biotieteiden tutki-
joita kehittämällä nykyaikaisia kielenkäsittely- ja tekstinlouhintatyökaluja, jotka
voivat skannata koko julkisesti saatavilla olevan tutkimuskirjallisuuden ja poimia, lu-
okitella ja koota siitä löytyvää tietoa. Näin esimerkiksi molekyylibiologian, lääketi-
eteellisen tekniikan, bioinformatiikan, geenitekniikan ja biokemian tutkijat pysyvät
ajan tasalla viimeisimmistä tutkimustuloksista, ja heidän tutkimustyönsä helpottuu.

Väitöstutkimukseni keskittyy lähes yksinomaan biolääketieteellisten relaatioiden
ja tapahtumien louhintaan. Tällä tarkoitetaan biolääketieteellisessä kirjallisuudessa
kuvattujen biologisten prosessien, vuorovaikutusten ja suhteiden automaattista tun-
nistamista. Biolääketieteellisten relaatioiden ja tapahtumien louhintajärjestelmät
voivat skannata valtavan määrän alan tekstejä ja havaita ja poimia automaattisesti

biolääketieteellisten nimettyjen entiteettien, kuten geenien, proteiinien, kemiallisten yhdisteiden ja sairauksien semanttisia suhteita.

Tällaisten louhintajärjestelmien tuottama rakenteinen tieto, eli poimitut relaatiot tai tapahtumat, voidaan tallentaa relaatiotietokantoihin tai molekyylien vuorovaikutusverkkoihin. Näistä voidaan helposti tehdä hakuja, tai niitä voidaan suodattaa, analysoida, visualisoida ja integroida muihin rakenteisiin tietolähteisiin.

Biomolekulaaristen vuorovaikutussuhteiden poimiminen on aina ollut BioNLP-tutkimuksen fokuksessa, koska tieto vuorovaikutussuhteista on ratkaisevan tärkeää monilla tutkimusaloilla, kuten täsmälääketieteessä, lääkekehityksessä, lääkkeiden uudelleenkohdistamisessa, hypoteesien luomisessa, signalointireittien rakentamisessa ja kuratoinnissa sekä proteiinien toiminnan ja rakenteen ennustamisessa.

Nykyiset relaatioiden ja tapahtumien louhintajärjestelmät perustuvat ohjattuun koneoppimiseen. Näiden järjestelmien kouluttaminen edellyttää manuaalisesti annotoitua aineistoa. Biolääketieteen alalla manuaalinen annotointi vaatii alan asiantuntemusta, ja se on erittäin aikaavievää. Näin ollen on yleistä, että tiedonlouhintajärjestelmien rakentamiseen on käytettävissä vain vähän koulutusaineistoa. Siksi vaaditaan menetelmiä, jotka pystyvät hyödyntämään mahdollisimman tehokkaasti kaiken käytettävissä olevan aineiston. Tämä tutkielma kokoaa yhteen tutkimustyöni, jossa paneudun tähän kysymykseen.

BioNLP on läpikäynyt ennennäkemättömiä muutoksia sen jälkeen, kun aloitin tutkimukseni tällä alalla lähes kymmenen vuotta sitten. BioNLP-yhteisön jäsenenä olen nähnyt monien menetelmien syntyminen, kehittymisen – ja joissakin tapauksissa katoamisen. Jokainen näistä menetelmistä on ollut askeleen edeltäjäänsä parempi. Viimeiset kymmenen vuotta voidaan jakaa karkeasti kolmeen ajanjaksoon.

Kun aloitin tutkimukseni, parhaiten menestyviä ja suosituimpia olivat piirrepohjaiset menetelmät, jotka pohjautuvat datan ominaisuuksia kuvaavien piirteiden tarkkaan suunnitteluun ja valitsemiseen. Sitten laitteistojen merkittävä kehitys sekä useat algoritmiikan ja menetelmänkehityksen läpimurrot mahdollistivat neuroverkkopohjaisen koneoppimisen hyödyntämisen reaali maailman sovelluksissa. Tällöin vallitseviksi tulivat konvoluutioneuroverkot, takaisinkytketyvät neuroverkot ja huomiopohjaiset rakenteet. Lopuksi transformer-arkkitehtuuriin pohjautuvat mallit, kuten BERT ja GPT, johtivat ennennäkemättömiin suorituskyvyn parannuksiin monissa eri BioNLP:n sovelluksissa. Pyydän lukijaa tätä tutkielmaa lukiessa huomioimaan kehityshistorian ja arvioimaan tuloksia sen perusteella, mitä kyseisenä ajankohtana oli mahdollista tehdä.

Acknowledgements

First and foremost, I would like to thank and express my deepest gratitude to my supervisors, Filip Ginter and Tapio Salakoski, who guided me through my doctoral studies. Tapio, thank you so much for putting your faith in me over the last several years. I wouldn't be where I am today without your support and encouragement. You created a very positive and an excellent atmosphere, not only for me, but for all the students, to study and work. This opportunity allowed us to have maximum focus on our research, and pursue our goals as energetically and passionately as possible. I'll forever be grateful for the support you gave me when I needed it and I will never forget all the nice occasions we, together with Filip, got to participate in different ceremonies, celebrating my achievements.

Filip, I don't even know where to start, but I will try to be right to the point and express my feelings concisely. In my honest opinion, you are one of the greatest teachers, supervisors, advisors and friends one can possibly have and I don't know how you manage to be all of these at the same time! I have seen how eager AI and NLP companies are to get a chance to hear your professional opinion on a matter, and now that I look back and recall we have been working together everyday, in the same room, and almost for a decade, I consider myself extremely lucky, having you by my side all these years. Thank you for contributing an incredible amount of your time and your effort, wisdom, patience, support and encouragement to my success, and thank you for your understanding and friendship all these years; I wouldn't be where I am today without your constant guidance and support. I will forever be in your debt and grateful for the rest of my life. Thank you for all the important lessons you have taught me and thank you for showing me how one should strive if he wants to become a great academic scholar. Not only you have guided and supported me to grow academically and professionally, you were always there for me and supported me even in my personal matters. Filip, simply thank you for being awesome!

Next, I would like to thank the reviewers of this thesis, Fabio Rinaldi and Jin-Dong Kim, for their constructive criticism and helpful comments, with which I was able to improve the thesis. In addition, I would like to thank Udo Hahn for agreeing

to act as my opponent.

While not a supervisor on this thesis, Sampo Pyysalo has also had a big impact on my progress in the field of biomedical text mining. Sampo, thank you for your invaluable support, sharing your knowledge and experience whenever I asked, even in the projects that you were not a co-author of the paper. I have learned a lot from you all these years and I am deeply grateful for your support. Similarly, I would like to thank Sofie Van Landeghem for the numerous interesting discussions we had about biomedical text mining methods. I would also like to express my warmest thanks to Martti Tolvanen, who taught me the basics of biochemistry and molecular biology (the knowledge that helped me a lot in pursuing my research goals in the field of biomedical text mining), and to Tapio Pahikkala and Antti Airola, who taught me various advanced topics in machine learning. I am especially thankful for teaching me *how NOT to use machine learning techniques*, and *how NOT to interpret the obtained results*. In addition, I am very grateful to Veronika Laippala for her numerous help, including providing a Finnish translation of my thesis abstract.

My heartfelt gratitude also goes to the post doctoral researchers and doctoral students who have collaborated with me over the last several years. Jari Björne, Hans Moen, Suwisa Kaewphan and Kai Hakala, thank you for your friendship and thank you for being very reliable in collaboration and excellent in teamwork! This work wouldn't have been possible if it were not for the help I have received from you all these years! In addition to the mentioned co-authors, I'd like to extend my sincere thanks to the current and past members of the TurkuNLP group, including Aki-Juhani Kyröläinen, Samuel Rönqvist, Jenna Kanerva, Juhani Luotolahti, Li-Hsin Chang, Alekski Vesanto, Antti Virtanen, Akseli Leino, Jouni Luoma, and Risto Luukkonen. Your friendship made my days better and our lives very pleasant all these years. Thank you for your friendship! Finally, I would like to thank all my colleagues in the Department of Computing, University of Turku. Especially, I like to express my warmest thanks to Juho Heimonen: Juho, I don't remember a single day that you saw me in the department and you didn't welcome me with a cheerful smile! Thank you for the few hundreds of occasions that I was exhausted and you brightened my day with your smile, friendly discussions, and your positive attitude!

I would also like to acknowledge the sources of funding that made this work possible: The University of Turku Graduate School (UTUGS); The Finnish Cultural Foundation (Suomen Kulttuurirahasto) for funding the project "Impacting Biomedical Research: Unsupervised Deep Learning for Literature Mining"; and Nokia Foundation Award (2018) for funding the biomedical entity and pair embeddings project. Last but definitely not least, I would like to thank my supervisors for providing the funds I needed for my studies.

I might not have ended up an NLP specialist, had it not been for Eija Nordlund who helped me to achieve my dream by introducing me to the field of biomedical text mining and arranging an official interview with Filip. Having obtained a master's degree certificate in software engineering in Iran and passing two courses in general machine learning and data mining methods, I remember I had become extremely passionate about getting a second master's degree in NLP or text mining, but the options seemed to be quite limited for me at the time. Either such a master's degree programme didn't exist in the countries I was searching for, or I had to pay a significant tuition fee that I couldn't afford. Therefore, I decided to apply for a master's degree programme in bioinformatics; After all, bioinformatics could allow me to apply machine learning methods at large-scale and on unstructured data (somehow similar to what NLP and text mining would). Hence, I searched for, applied and I finally got admitted to the master's degree programme in bioinformatics at the University of Turku and I moved to Finland. Even though I was enjoying what I was learning at the university, my heart was still in pursuing my dreams in NLP and text mining—*words carry and pass more information than nucleotides!* When I discussed my passion with Eija (the coordinator of our program), she pointed me to the field of biomedical text mining (a field I could enter while studying to get a degree in bioinformatics). She also informed me about the existence of an excellent NLP group at our own department and she arranged an official interview with Filip! Eija, thank you so much for your wisdom and care! I will never forget how you opened this door for me and helped me to achieve my dream!

Finally, and most importantly, I would like to thank and express my deepest gratitude to my brother Amir Hossein, for his love, and to my parents, Abolhassan Mehryari and Farahdokht Makodehi, for their constant love, support, encouragement, wisdom and sacrifices throughout my life. No words can possibly express the love and admiration I have for you. I am forever in your debt and everyday, I pray to have you by my side for the rest of my life. Thank you for everything! *I love you!*

December, 2021
Farrokh Mehryary

List of original publications

This dissertation is based on the following original publications, which are referred to in the text by their Roman numerals:

- I Mehryary, F., Kaewphan, S., Hakala, K., and Ginter, F. (2016b). Filtering large-scale event collections using a combination of supervised and unsupervised learning for event trigger classification. *Journal of Biomedical Semantics*, 7(1):1–13
- II Mehryary, F., Björne, J., Pyysalo, S., Salakoski, T., and Ginter, F. (2016a). Deep learning with minimal training data: TurkuNLP entry in the BioNLP shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 73–81, Berlin, Germany. Association for Computational Linguistics
- III Mehryary, F., Hakala, K., Kaewphan, S., Björne, J., Salakoski, T., and Ginter, F. (2017b). End-to-end system for bacteria habitat extraction. In *BioNLP 2017*, pages 80–90, Vancouver, Canada. Association for Computational Linguistics
- IV Hakala, K., Mehryary, F., Moen, H., Kaewphan, S., Salakoski, T., and Ginter, F. (2017). Ensemble of convolutional neural networks for medicine intake recognition in twitter. In *Proceedings of the 2nd Social Media Mining for Health Research and Applications (SMM4H 2017) Workshop*, pages 59–63. CEUR-WS.org. (equal contribution between Hakala and Mehryary)
- V Mehryary, F., Björne, J., Salakoski, T., and Ginter, F. (2018). Potent pairing: ensemble of long short-term memory networks and support vector machine for chemical-protein relation extraction. *Database : the journal of biological databases and curation*, 2018:bay120
- VI Mehryary, F., Moen, H., Salakoski, T., and Ginter, F. (2020). Entity-pair embeddings for improving relation extraction in the biomedical domain. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020 (online event)*, pages 613–618. i6doc.com publication

The list of original publications have been reproduced with the permission of the copyright holders.

List of co-authored publications not included in the thesis

- Hakala, K., Kaewphan, S., Bjorne, J., Mehryary, F., Moen, H., Tolvanen, M., Salakoski, T., and Ginter, F. (2020). Neural Network and Random Forest Models in Protein Function Prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1
- Zhou, N., Jiang, Y., Bergquist, T., Lee, A., Kacsoh, B., Crocker, A., Lewis, K., Georghiou, G., Nguyen, H., Hamid, M., Davis, L., Dogan, T., Atalay, V., Rifaioglu, A., Dalkran, A., Cetin Atalay, R., Zhang, C., Hurto, R., Freddolino, P., Zhang, Y., Bhat, P., Supek, F., Fernández, J., Gemovic, B., Perovic, V., Davidović, R., Sumonja, N., Veljkovic, N., Asgari, E., Mofrad, M., Profiti, G., Savojardo, C., Martelli, P., Casadio, R., Boecker, F., Schoof, H., Kahanda, I., Thurlby, N., McHardy, A., Renaux, A., Saidi, R., Gough, J., Freitas, A., Antczak, M., Fabris, F., Wass, M., Hou, J., Cheng, J., Wang, Z., Romero, A., Paccanaro, A., Yang, H., Goldberg, T., Zhao, C., Holm, L., Törönen, P., Medlar, A., Zosa, E., Borukhov, I., Novikov, I., Wilkins, A., Lichtarge, O., Chi, P., Tseng, W., Linial, M., Rose, P., Dessimoz, C., Vidulin, V., Dzeroski, S., Sillitoe, I., Das, S., Lees, J., Jones, D., Wan, C., Cozzetto, D., Fa, R., Torres, M., Warwick Vesztrocy, A., Rodriguez, J., Tress, M., Frasca, M., Notaro, M., Grossi, G., Petrini, A., Re, M., Valentini, G., Mesiti, M., Roche, D., Reeb, J., Ritchie, D., Aridhi, S., Alborzi, S., Devignes, M., Koo, D., Bonneau, R., Gligorijević, V., Barot, M., Fang, H., Toppo, S., Lavezzo, E., Falda, M., Berselli, M., Tosatto, S., Carraro, M., Piovesan, D., Ur Rehman, H., Mao, Q., Zhang, S., Vucetic, S., Black, G., Jo, D., Suh, E., Dayton, J., Larsen, D., Omdahl, A., McGuffin, L., Brackenridge, D., Babbitt, P., Yunes, J., Fontana, P., Zhang, F., Zhu, S., You, R., Zhang, Z., Dai, S., Yao, S., Tian, W., Cao, R., Chandler, C., Amezola, M., Johnson, D., Chang, J., Liao, W., Liu, Y., Pascarelli, S., Frank, Y., Hoehndorf, R., Kulmanov, M., Boudellioua, I., Politano, G., Di Carlo, S., Benso, A., Hakala, K., Ginter, F., Mehryary, F., Kaewphan, S., Björne, J., Moen, H., Tolvanen, M., Salakoski, T., Kihara, D., Jain, A., Šmuc, T., Altenhoff, A., Ben-Hur, A., Rost, B., Brenner, S., Orenge, C., Jeffery, C., Bosco, G., Hogan, D., Martin, M., O’Donovan, C., Mooney, S., Greene, C., Radivojac, P., and Friedberg, I. (2019). The CAFA challenge reports improved

protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biology*, 20(1)

- Sarker, A., Belousov, M., Friedrichs, J., Hakala, K., Kiritchenko, S., Mehryary, F., Han, S., Tran, T., Rios, A., Kavuluru, R., de Bruijn, B., Ginter, F., Mahata, D., Mohammad, S. M., Nenadic, G., and Gonzalez-Hernandez, G. (2018). Data and systems for medication-related text classification and concept normalization from Twitter: insights from the Social Media Mining for Health (SMM4H)-2017 shared task. *Journal of the American Medical Informatics Association*, 25(10):1274–1283
- Mehryary, F., Björne, J., Salakoski, T., and Ginter, F. (2017a). Combining support vector machines and LSTM networks for chemical-protein relation extraction. In *Proceedings of the BioCreative VI Workshop*, pages 175–179
- Kaewphan, S., Mehryary, F., Hakala, K., Salakoski, T., and Ginter, F. (2017). TurkuNLP entry for interactive Bio-ID assignment. In *Proceedings of the BioCreative VI Workshop*, pages 32–35
- Moen, H., Hakala, K., Mehryary, F., Peltonen, L.-M., Salakoski, T., Ginter, F., and Salanterä, S. (2017). Detecting mentions of pain and acute confusion in Finnish clinical text. In *Proceedings of the 2017 Workshop on Biomedical Natural Language Processing*, pages 365–372, Vancouver, Canada. Association for Computational Linguistics
- Jiang, Y., Oron, R. T., Clark, T. W., Bankapur, R. A., D’Andrea, D., Lepore, R., Funk, S. C., Kahanda, I., Verspoor, M. K., Ben-Hur, A., Koo, E. D. C., Penfold-Brown, D., Shasha, D., Youngs, N., Bonneau, R., Lin, A., Sahraeian, E. S. M., Martelli, L. P., Profiti, G., Casadio, R., Cao, R., Zhong, Z., Cheng, J., Altenhoff, A., Skunca, N., Dessimoz, C., Dogan, T., Hakala, K., Kaewphan, S., Mehryary, F., Salakoski, T., Ginter, F., Fang, H., Smithers, B., Oates, M., Gough, J., Törönen, P., Koskinen, P., Holm, L., Chen, C.-T., Hsu, W.-L., Bryson, K., Cozzetto, D., Minneci, F., Jones, T. D., Chapman, S., BKC, D., Khan, K. I., Kihara, D., Ofer, D., Rappoport, N., Stern, A., Cibrian-Uhalte, E., Denny, P., Foulger, E. R., Hieta, R., Legge, D., Lovering, C. R., Magrane, M., Melidoni, N. A., Mutowo-Meullenet, P., Pichler, K., Shypitsyna, A., Li, B., Zakeri, P., ElShal, S., Tranchevent, L.-C., Das, S., Dawson, L. N., Lee, D., Lees, G. J., Sillitoe, I., Bhat, P., Nepusz, T., Romero, E. A., Sasidharan, R., Yang, H., Paccanaro, A., Gillis, J., Sedeño-Cortés, E. A., Pavlidis, P., Feng, S., Cejuela, M. J., Goldberg, T., Hamp, T., Richter, L., Salamov, A., Gabaldon, T., Marcet-Houben, M., Supek, F., Gong, Q., Ning, W., Zhou, Y., Tian, W., Falda, M., Fontana, P., Lavezzo, E., Toppo, S., Ferrari, C., Giollo, M., Piovesan, D., Tosatto, C. S., del Pozo, A., Fernández, M. J., Maietta, P., Valencia, A., Tress,

- L. M., Benso, A., Di Carlo, S., Politano, G., Savino, A., Rehman, U. H., Re, M., Mesiti, M., Valentini, G., Bargsten, W. J., van Dijk, J. A. D., Gemovic, B., Glisic, S., Perovic, V., Veljkovic, V., Veljkovic, N., Almeida-e Silva, C. D., Vencio, N. R. Z., Sharan, M., Vogel, J., Kansakar, L., Zhang, S., Vucetic, S., Wang, Z., Sternberg, E. M. J., Wass, N. M., Huntley, P. R., Martin, J. M., O'Donovan, C., Robinson, N. P., Moreau, Y., Tramontano, A., Babbitt, C. P., Brenner, E. S., Linial, M., Orengo, A. C., Rost, B., Greene, S. C., Mooney, D. S., Friedberg, I., and Radivojac, P. (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(1):1–19
- Mehryary, F., Kaewphan, S., Hakala, K., and Ginter, F. (2014). Eliminating incorrect events from large-scale event networks by trigger word clustering and pruning. In *Proceedings of the 6th International Symposium on Semantic Mining in Biomedicine (SMBM 2014)*, pages 75–79
 - Hakala, K., Mehryary, F., Kaewphan, S., and Ginter, F. (2013a). Hypothesis generation in large-scale event networks. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine (LBM'13)*, pages 19–28

Abbreviations

ASR automatic speech recognition

Bi-LSTM bidirectional long short-term memory network

BioNLP biomedical natural language processing

BPE byte pair encoding

CNN convolutional neural network

CRF conditional random fields

DNN deep neural networks

GRU gated recurrent unit

IE information extraction

IR information retrieval

LSTM long short-term memory network

ML machine learning

NER named-entity recognition

NLM neural language model

NLP natural language processing

NLU natural language understanding

NMT neural machine translation

OCR optical character recognition

OOV out-of-vocabulary

POS part-of-speech

QA question answering

RNN recurrent neural network

SDP shortest dependency path

SVM support vector machine

WPM WordPieceModel

Contents

| | |
|--|--------------|
| Contents | xviii |
| 1 Introduction | 1 |
| 1.1 Biomedical relation and event extraction | 2 |
| 1.2 Research questions | 5 |
| 2 Foundations | 10 |
| 2.1 NLP pre-processing tasks | 10 |
| 2.2 Relation and event extraction | 28 |
| 2.3 Performance Metrics | 33 |
| 3 Research in brief | 37 |
| 3.1 Optimizing large-scale event databases | 37 |
| 3.1.1 Introduction | 37 |
| 3.1.2 Data | 38 |
| 3.1.3 Method | 39 |
| 3.1.4 Evaluation and results | 41 |
| 3.1.5 Discussion and conclusions | 43 |
| 3.2 Deep learning with minimal training data | 45 |
| 3.2.1 Introduction | 45 |
| 3.2.2 Methods | 47 |
| 3.2.3 Dealing with minimal training data: measuring and overcoming variance | 49 |
| 3.2.4 Results | 51 |
| 3.2.5 Discussion and conclusions | 52 |
| 3.3 Exploring different contexts for relation extraction | 54 |
| 3.3.1 Data | 55 |
| 3.3.2 Method | 55 |
| 3.3.3 Results | 58 |
| 3.3.4 Discussion and conclusions | 58 |
| 3.4 Hybrid relation extraction systems | 59 |
| 3.4.1 System-level combination | 60 |
| 3.4.2 Feature-level combination | 64 |

| | |
|---|-----------|
| 3.5 Entity-pair embeddings for improving relation extraction in the biomedical domain | 68 |
| 3.5.1 Introduction | 68 |
| 3.5.2 Method | 69 |
| 3.5.3 Results | 72 |
| 3.5.4 Discussion and conclusions | 74 |
| 4 Discussion and conclusion | 76 |
| 4.1 Contributions of the thesis | 76 |
| 4.2 Future directions for biomedical relation extraction | 77 |
| Bibliography | 79 |
| Original Publications | 91 |

1 Introduction

The amount of new discoveries in life sciences has been growing at an increasing rate over the past few decades, while the formal ways in which the new findings are announced and shared within the research community have roughly stayed the same and confined to scientific publication in relevant journals and conference proceedings.

PubMed¹ (the primary database of biomedical literature and life sciences journals) currently contains over 21 million article abstracts² and more than one million of them were published in 2020 alone. The overwhelming amount and the increasing rate of publication in the biomedical domain make it difficult for life sciences researchers to acquire and maintain all information that is necessary for their research. For instance, a cancer research scientist may need to acquire a complete list of genes that interact with BRCA1 (breast cancer 1) gene. The PubMed keyword search for BRCA1 returns no less than 16,531 article abstracts³, making manual inspection of the retrieved documents impractical. Missing even one of the interacting gene partners in this scenario may jeopardize successful development of a potential new drug. Even though manually curated databases of biomolecular interactions exist, they are usually not up-to-date and they require notable human effort to maintain. To summarize, new discoveries are constantly being shared within the community via scientific publishing, but unfortunately the probability of missing vital information for research in life sciences is increasing.

In response to this problem, the biomedical natural language processing (BioNLP) community of researchers has emerged and strives to assist life sciences researchers by building modern natural language processing (NLP), text mining, information extraction (IE) and information retrieval (IR) tools which can be applied at large-scale and scan the whole publicly available literature (PubMed article abstracts and PubMed Central Open Access (PMCOA) full article texts) and extract, classify, and aggregate the information found within, thus keeping life sciences researchers always up-to-date with the recent relevant discoveries and facilitating their research in numerous fields such as molecular biology, biomedical engineering, bioinformatics, genetics engineering and biochemistry. On the one hand, such tools should be

¹<https://www.ncbi.nlm.nih.gov/pubmed/>

²queried on 2021.03.25

³queried on 2021.03.25

able to deal with the inherent complexity and ambiguity of natural languages, and on the other hand, they should be able to normalize the variability of natural language statements, meaning that a single phenomenon, discovery, prediction or speculation can be stated in various ways in a natural language, but all variations of the same statement should be identified and mapped into a single unified form after extraction from the literature.

1.1 Biomedical relation and event extraction

Among all different tasks, biomedical relation and event extraction tasks have received much attention in the BioNLP community. These foundational information extraction tasks deal with automatic detection of biological processes, interactions and relations described in the biomedical literature. Precisely speaking, biomedical relation and event extraction systems can scan through a vast amount of biomedical texts and automatically detect and extract the semantic relations of biomedical *named entities* (phrases, nouns and abbreviations referring to genes, proteins, chemical compounds, diseases and other biomedical entities). The structured outputs of such systems (i.e., the extracted relations or events) can be stored as relational databases or molecular interaction networks which can easily be queried, filtered, analyzed, visualized and integrated with other structured data sources (see Figure 1).

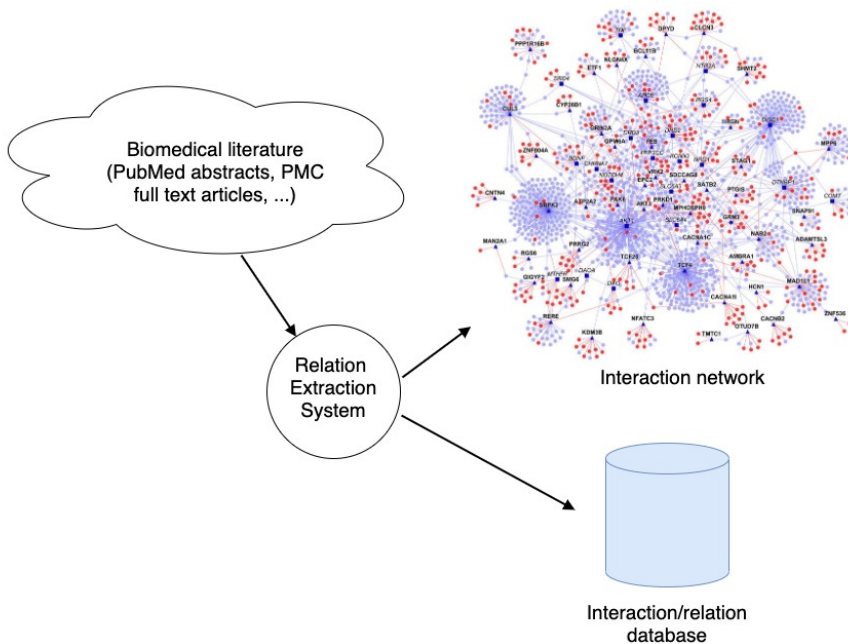


Figure 1. Illustration of relation/event extraction systems.

Biomolecules (e.g., proteins, nucleic acids, carbohydrates and lipids) are fundamental building blocks of living organisms that play indispensable roles in all life processes including metabolism, growth, active and passive transport, producing energy, reproduction, and defence against pathogens via the immune system (Björne, 2014). Having knowledge about biomolecular interactions is crucially important in various research areas including precision medicine, drug discovery, drug repurposing, hypothesis generation, construction and curation of signaling pathways, and protein function prediction. Therefore, the majority of the efforts within the BioNLP community has been spent on building resources and methods to extract information about the interactions of biomolecules.

Traditionally, biomedical relation extraction has focused on finding semantic relations between **pairs** of named entities. A number of shared tasks⁴ have been organized to promote the development and systematic evaluation of pairwise relation extraction methods for the biomedical domain. For example, the BioCreative II and BioCreative III challenges (Krallinger et al., 2008, 2011) focused on extracting protein-protein interactions (PPI), while the BioCreative V CDR Task focused on chemical-disease relation extraction (Wei et al., 2016). The two drug-drug interaction shared tasks (DDI-2011 and DDI-2013) focused on the detection of adverse interactions between pairs of drugs (Segura-Bedmar et al., 2011, 2013), and BioCreative VI Track 5 (also known as ChemProt Track) focused on extracting chemical-protein interactions (CPI) from PubMed article abstracts (Krallinger et al., 2017). However, not all shared tasks have focused on interactions of biomolecules. For example, the four Bacteria-Biotope (BB) tasks in BioNLP Shared Task 2011 (BioNLP-ST-11), BioNLP Shared Task 2013 (BioNLP-ST-13), BioNLP Shared Task 2016 (BioNLP-ST-16) and BioNLP Open Shared Tasks 2019 (BioNLP-OST-19) aimed at extracting the location of bacteria from scientific Web pages and PubMed abstracts (Bossy et al., 2011, 2013; Deléger et al., 2016; Bossy et al., 2019).

In pairwise relation extraction tasks, each input to the system is a text (e.g., a sentence or a paragraph) with two candidate named entities (using a named-entity recognition (NER) system, named entities are found in advance) and the system predicts whether the text states any relations/interactions between the two entities (see Figure 2).

Depending on the task definition, pairwise relations can be either `typed` or `untyped`. For example, if a protein-protein relation extraction task is defined as a simple binary classification problem, detecting whether the two protein mentions interact or not, the extracted PPIs are `untyped` (i.e., they do not specify anything particular about the type of the interaction between their arguments), but in a multi-class or multi-label classification setup, the aim is to extract the different types of relations

⁴Shared tasks in computer science are public competitions where the participants are given a common goal and within a certain time frame, they have to produce a solution to the defined problem (Björne, 2014).

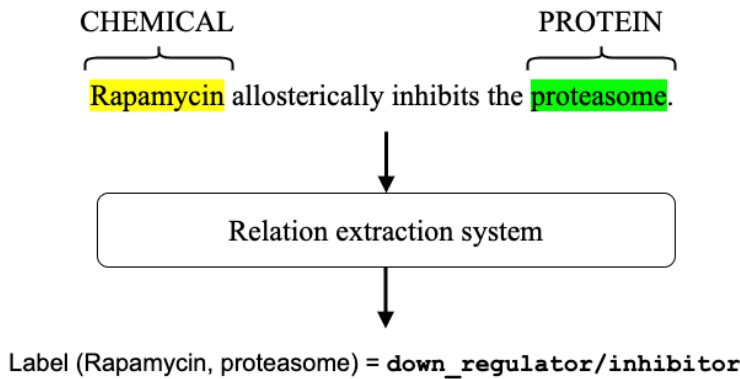


Figure 2. Chemical-protein relation extraction example. The input is a sentence with the two entities, and the output is the predicted label for the pair.

between the proteins (such as *Binding*, *Regulation* and *Phosphorylation*). In addition, pairwise relations can be either *directional* or *non-directional*. Some relations such as *Binding* are semantically non-directional, e.g., $Binding(P_1, P_2) \implies Binding(P_2, P_1)$, and some types such as *Regulation* are directional, e.g., $Regulation(P_1, P_2) \not\implies Regulation(P_2, P_1)$.

Although pairwise relation extraction can provide information that is detailed enough for many research applications, simple pairwise relations are not fully capable of capturing and representing the meaning of complicated statements written in biomedical texts. To deal with this problem, *biomedical events* have been proposed as an alternative for pairwise interactions. Biomedical event annotations were first introduced by the GENIA (Tateisi et al., 2005; Ohta et al., 2006; Kim et al., 2008) and BioInfer (Pyysalo et al., 2007) corpora, and biomedical event extraction was popularized by the BioNLP 2009 Shared Task on Event Extraction (Kim et al., 2009). Events are complex relations that aim to provide an annotation scheme detailed and flexible enough to fully capture, extract and represent the semantics of natural language with a formal representation. Events are characterized by (1) annotated *trigger words*, (2) directed and typed arguments and (3) the ability to nest other events. For example, the sentence “Protein A causes protein B to bind protein C” can be annotated with the nested event structure $Cause(A, Bind(B, C))$ (Björne, 2014). Obviously, biomedical events capture the semantics of the statements better than simple pairwise relations (and this might be useful for some applications in modern biology research), but at the same time, they add quite a lot of complication into the information extraction pipelines, because it is generally much harder to extract the nested events than it is to extract binary relations.

1.2 Research questions

Currently, state-of-the-art relation and event extraction methods are based on supervised learning methods. These methods require manually annotated data for training, but creating manually annotated data for the biomedical domain is considerably time-consuming and it requires domain expertise. Therefore, the much-needed training data for building biomedical relation and event extraction systems can sometimes be minimal.

For example, biomedical event description covers a wide variety of different event types. There is a fair amount of manually annotated training data for some event types (e.g., *Binding*, *Regulation* and *Phosphorylation*) that are discussed very frequently in the literature, whereas for some rarer event types (e.g., *Acetylation*, *Ubiquitination*, *Glycosylation*, and *Hydroxylation*) training data is limited. In other words, we only have massive/sufficient training data for a small selection of the event types, thus scaling up event extraction systems to cover all types of events is difficult and therefore we need to have methods that can efficiently deal with a very limited amount of training data.

Not having a fair amount of training data can lead to a poor extraction performance. On one hand, a system may fail to extract a considerable proportion of events/relations from the literature, and on the other hand, many of the extracted events/relations might be actually incorrect. These can lower the credibility of text mining-based resources and make them not so reliable for serious biomedical research. Therefore, thousands of expert-hours are still spent to keep “manually curated databases” up-to-date, but due to the significant rate of publication in the biomedical domain, the risk of missing vital information for biomedical research is unfortunately high.

The main focus of this dissertation is to improve the performance of event and relation extraction for the biomedical domain, either by optimizing machine learning, text mining and natural language processing methods (i.e., by making better event and relation extraction systems), or by improving the quality of the extracted data in post processing, after relation and event extraction systems have been executed at large-scale and extracted millions of relations/events from the literature⁵. In

⁵In contrast to my research topic which has primarily focused on supervised relation extraction, a separate line of research—introduced by Mintz et al. (2009) and further developed by Riedel et al. (2010) and Surdeanu et al. (2012)—has focused on using *distant supervision* and *multi-instance learning* to build relation extraction system when *no* labeled data is available. Such methods rely on an existing knowledge base of entity-pair relations to train a relation extraction classifier that can extract new relations from textual data. Typically, the schema of a used knowledge base (e.g., entity and relation types), dictates what sorts of new relations can be extracted from texts. For biomedical text mining applications, sometimes the required knowledge base does not exist beforehand, or the schema of the knowledge base is not in harmony with what a biologist need to be extracted from the texts. Therefore, distant supervision relation extraction methods have not become super popular for the biomedical domain. For a literature survey about distant supervision methods for biomedical relation extraction,

particular, we aim to address the following research questions.

Question 1: Improving the quality of extracted data in automatically generated event and relation databases

During the past decade, several event/relation extraction systems have been applied at large scale, extracting millions of events from massive text corpora (Van Landeghem et al., 2013a; Gerner et al., 2012). These large corpora, typically the totality of PubMed abstracts and PubMed Central full-text articles, contain a number of documents which are partly or entirely out-of-domain for these systems, being unlike the manually annotated shared task data that are carefully selected from narrow biological domains on which the systems have been trained. Facing documents from such previously unseen domains, the systems often produce suboptimal output. On one hand, these systems can produce thousands/millions of incorrect events (false positives events, i.e. incorrect relations that are not actually stated in the input texts), thus lowering the perceived precision, and on the other hand, they may fail to extract thousands/millions of correct events (i.e. false negatives), leading to lower recall values. Tuning the performance of these systems in the general domain requires further effort.

Increasing the recall usually translates to running a better information extraction system on all previously processed documents. This obviously demands a few thousand extra CPU/GPU hours, and for institutions with limited budget or access to computational resources, it is usually not considered to be a viable option. Increasing the precision, in contrast, can be done in post-processing and by finding and removing incorrect events/relations from the system output, i.e. from the event/relation databases or molecular interaction networks. The first research question addressed in this dissertation is what sort of approaches can be developed to automatically and efficiently target and remove incorrectly extracted relations/events from the output of relation/event extraction systems and improve the precision.

Question 2: Deep learning with minimal training data

State-of-the-art relation and event extraction methods are based on supervised machine learning. In particular, supervised deep learning-based methods have recently shown superior performance on many datasets. As discussed earlier, supervised methods require manually annotated texts for training and development, but the much-needed training data for building biomedical text mining systems is sometimes minimal. This hinders the utilization of the deep learning-based methods to their full potential.

A general rule of thumb is that more complex neural network architectures can

please refer to Boudjellal et al. (2020).

learn better representations of their inputs, leading to better performance on downstream tasks. However, adding more layers to a neural network, increasing the dimensionality of the layers or adding more edges to an architecture, all translate into having more neural network weights and demanding more data for neural network training. Generally, the more network weights an architecture includes, the more training data is required to learn those weights.

One solution is to use unsupervised methods to pre-train network components in advance on large corpora. For example, word vector representations (embeddings) can be pre-trained in advance and then plugged into different neural models. Similarly, recurrent neural network (RNN), convolutional neural network (CNN) or transformer-based encoders (e.g. BERT (Devlin et al., 2018)) can be pre-trained on large corpora and then fine-tuned with the task-specific training data for a particular relation extraction task at hand. However, even with pre-trained embeddings or encoders, sometimes neural models cannot efficiently learn the tasks when training and development (validation) data is very limited. Deep neural networks can have millions of weights and learning/fine-tuning the weights and finding the best hyperparameters (such as the learning rate, mini-batch size and number of epochs) become challenging with minimal training and development data.

The second research question of this thesis is how to efficiently train and optimize deep neural network models for biomedical relation extraction tasks with minimal training and development data. Instead of designing yet another neural network architecture, we aim to search for simple methods that can be utilized for any relation extraction task.

Question 3: Exploring different contexts for relation extraction

Normally, each input to a relation extraction system is a text (e.g., a sentence or a paragraph) and two candidate named entities and the system aims to predict if the text states any relations between the two entities or not. In this manner, the context plays a crucial role in detecting the semantic relations. Focusing on the entities that occur in the same sentence, relation extraction approaches can be broadly divided into three main categories, based on the type of context that they use.

The first approach is using all tokens of the sentence as the context for relation extraction. For example, in one approach, the words in the sentence are first divided into three groups: *before*, *between* and *after* the two entities. Each group is further represented with a bag-of-features. Mooney and Bunescu (2006) used this approach to build a subsequence kernel for each bag, with the final kernel function being simply the sum of the three kernels, which is further used with a support vector machine (SVM) classifier for relation classification. Another example is the convolutional neural network (CNN)-based relation extraction system developed by Nguyen and Grishman (2015). Their neural network model processes all tokens of the sentence

and relies on *position embeddings* to encode the relative distances of each word in the sentence to the two entities of interest.

The second approach is using the shortest dependency path (SDP) as the context. The shortest path connecting the two candidate named entities in the sentence dependency parse graph is known to contain most of the words relevant to characterizing their relationship, while excluding less relevant and uninformative words (Bunescu and Mooney, 2005). For this reason, many successful feature-based, kernel-based and deep learning-based relation extraction systems have been developed based on utilizing the SDP in various forms (Bunescu and Mooney, 2005; Airola et al., 2008; Chowdhury et al., 2011; Björne, 2014; Liu et al., 2015; Can et al., 2019; Li et al., 2019).

The third approach is relying on the SDP in conjunction with the full sentence tokens. For example, the Turku Event Extraction System (TEES) builds on the features that are extracted from the SDP, as well as features that are based on the words located in the distance of $[-3,+3]$ tokens to the two entities in the sentence (Björne, 2014). Another example is the neural network system that has been developed by Li et al. (2019) for clinical relation extraction. Their system uses a bidirectional long short-term memory network (Bi-LSTM) to capture the features in the sentence sequence, as well as a CNN and a Bi-LSTM to capture the syntactic context for target entities using SDP information. Similarly, the CNN-based model developed by Peng et al. (2018) relies on SDP and full sentence tokens for chemical-protein relation extraction.

Unfortunately, the relation extraction systems belonging to either one of the three approaches mentioned above, are usually evaluated on different corpora or rely on different sets of features and utilize different machine learning techniques for optimization. This makes it difficult to judge which of the mentioned approaches is superior for relation extraction. Therefore, a systematic comparison of the three approaches would be interesting. In particular, this thesis aims to methodically explore the effect of using full sentence tokens *besides* the SDP tokens for improving the performance of relation extraction in modern deep learning-based methods.

Question 4: Exploring hybrid relation extraction methods

According to Zhang et al. (2017), supervised relation extraction methods can be broadly divided into three main groups: (1) feature-based methods, (2) kernel-based methods and (3) deep learning-based methods, each requiring a distinct set of skills and expertise such as feature-engineering, kernel design and neural network architecture design and pre-training.

For example, feature-based methods heavily rely on feature engineering, thus improving relation extraction performance in this approach usually translates to designing and extracting better features to characterize and *represent* each example

(entity pair) to classification algorithms. During the past decades, the BioNLP community has put a lot of effort and engineered various creative features that can be extracted from the input and enhance relation extraction performance.

In contrast, deep learning-based methods can *automatically learn* efficient representations from their input that are suitable for the relation classification task at hand. For example, convolutional neural networks can automatically learn position-invariant semantic n-grams from the training data and recurrent neural networks are very efficient in learning good representations of input sequences. Therefore, for improving relation extraction performance in this approach, researchers have focused on designing better neural network architectures and utilized various pre-training methods to learn neural network weights in advance.

While there have been great advances in each group, a separate line of research has focused on building hybrid relation extraction methods. In particular, many successful relation extraction methods have been developed by combining feature-based and deep learning-based methods. This combination can be achieved in two levels: system-level combination (i.e., combining the predictions of various systems) and feature-level combination (e.g., incorporating engineered features into neural network models). In this thesis, we aim to explore and hopefully propose new methods for building hybrid relation extraction systems.

Question 5: Unsupervised pre-training of biomedical entity-pair embeddings to improve relation extraction performance

Biomedical literature includes a lot of information about the relations and interactions of biomedical named entities (e.g. genes, proteins, chemicals, and drugs). We aim to leverage this literature-wide information using unsupervised methods and for every unique named-entity pair (E_i, E_j) , capture all stated information about E_i and E_j and their relations and build embeddings (vector representations) of entities and entity pairs. Similarly to word2vec (Mikolov et al., 2013) for ordinary words, our objective is for similar entities (e.g., proteins and chemicals) and pairs (e.g., protein-chemical pairs) to obtain similar embeddings.

We are especially interested in investigating the possible effects of incorporating these entity and entity-pair embeddings into neural models, in order to improve the performance in relation extraction tasks in the biomedical domain. In addition, we are interested to explore different approaches for pre-training vector representations for biomedical entities and entity pairs.

2 Foundations

This chapter provides essential natural language processing (NLP) background knowledge that is necessary for understanding the methods discussed throughout this thesis.

Section 2.1 provides a brief overview of various NLP pre-processing tasks, with a special focus on the biomedical domain. The list of all pre-processing tasks is extensive, hence I will only focus on the tasks that are relevant to relation and event extraction tasks.

Since the majority of my research has been focused on biomedical relation extraction task, in section 2.2 I will discuss the basics of relation and event extraction. This section also aims to present a brief literature survey about the history of relation extraction methods that have been developed for the biomedical domain.

Not all text mining systems perform very satisfactory in large-scale real-world applications. A typical way to estimate the performance of a system on unseen data is to prepare a representative held-out test set and use suitable performance metrics for evaluation¹. Therefore, in section 2.3, I will briefly review the performance metrics that are used in my research and the shared tasks that I have participated in.

2.1 NLP pre-processing tasks

Natural language processing and text mining systems are typically implemented as *end-to-end* systems, meaning that they accept **raw texts** as their input and generate final outputs to address end-users' information needs. Traditionally, end-to-end systems are implemented as software pipelines that automatically execute chains of tools, each dealing with one or more particular NLP task(s) (e.g., sentence boundary detection, tokenization, part-of-speech (POS) tagging, syntactic parsing, named-

¹I underscore that in some scenarios, achieving a good performance on a held-out test set does not necessarily guaranty that the system will achieve a good performance in large-scale applications. For example, BioNLP shared tasks do normally have a genuine held-out test set, but these typically represent the same sub-domain as the training and development sets, i.e., the test set has the same data selection biases as the training and development data. After training and optimizing a system on carefully selected narrow-domain shared task data, the system can make a lot of mistakes if it is executed on all PubMed abstracts, because there are a number of documents which are partly or entirely out-of-domain for the system. Tuning the performance of systems in such scenarios requires further effort and this is extensively discussed in the next chapter.

entity recognition or relation extraction), either transforming/annotating the input and generating a set of features that are required by the subsequent tools in the pipeline, or generating the final set of outputs. This problem decomposition strategy allows NLP practitioners to focus on one specific task at a time, create annotated corpora for that task and develop better tools to address the task. It is worth mentioning that in modern text mining systems, sometimes two or more NLP tasks are performed *jointly* by a single neural network model that receives raw texts as its input, and for this reason, the term “end-to-end” is sometimes used to imply that the process is joint (instead of a pipeline), but that is a narrower definition of end-to-end system.

NLP includes an extensive list of different tasks. Some tasks such as sentence boundary detection, tokenization, stemming, lemmatization, POS-tagging and parsing are usually viewed as pre-processing tasks (thus appearing earlier in NLP pipelines) and other tasks such as named-entity recognition, event/relation extraction or sentiment analysis are usually the ultimate goal, providing the final results. In the subsequent sections, I will briefly discuss the NLP pre-processing tasks that are relevant to relation and event extraction tasks.

Tokenization

Recognizing the constituent words of a sentence (represented as a *sequence of characters*) considerably facilitates the understanding of its semantics as a whole. After transforming a sentence into the sequence of constituent words or *tokens*, we can determine the *type* of each token (e.g. being a noun, a verb or an adjective) by subsequent part-of-speech tagging. In addition, we can analyze the syntactic structure of the sentence and determine the grammatical relationships among the words/tokens (e.g., determining the subject(s) and object(s) for each verb in the sentence).

The task of segmenting a text into its constituent tokens is referred to as *lexical tokenization* or *tokenization* for short. This is usually a preliminary and necessary pre-processing step for many subsequent tasks including part-of-speech (POS) tagging and parsing, since these tasks generally require the sentence to be already segmented into its composing elements. Other tasks such as named-entity recognition or relation extraction **may or may not** rely on tokenization, depending on whether they work on token-level (versus working *purely* on character-level) and if they utilize POS-tagging/parsing features².

Typically, the domain and the particular requirements of downstream tasks, dictate the strategy for tokenization. One basic tokenization approach is to regard *white spaces* (e.g. space, tab or newline) and punctuation characters as separators. Although this works flawlessly for phrases like “*analysis of protein-chemical interac-*

²Modern neural network-based NLP and text mining methods rely on using sufficiently common *subwords* instead of words or tokens. Subword tokenization is discussed in the next section.

tions,” or “... *aspartate aminotransferase (AST)*, *cytoplasmic and phosphoenolpyruvate carboxykinase (ATP) (PEPCK)* were up-regulated.”, it can segment individual gene or protein names (such as *(MIP)-1 α* or *TrpEb_1*) into separate tokens (which might be improper for the subsequent tools in the pipeline). This simple example shows that the tokenizers that are developed for the general domain, may not perform very well on biomedical/clinical texts.

As discussed by He and Kayaalp (2006), choosing the right tokenizer for the biomedical domain is not trivial and several factors affect the choice of right tokenizer. For example, since tokenization affects the indexing for IR, it is necessary to fine-tune the tokenization rules to be able to address the information retrieval needs once the indexing is done. Similarly, the choice of POS-tagger/parser affects the choice of the tokenizer.

Tokenizers can be broadly divided into two main groups : (1) rule-based tokenizers and (2) trainable tokenizers which work based on supervised machine learning (He and Kayaalp, 2006). In the rule-based approach, experts carefully analyze the domain and existing texts and implement and optimize a set of rules (which determine the conditions for segmentation) to improve the tokenization for the downstream tasks at hand. For example, to improve chemical named-entity recognition performance, Akkasi et al. (2016) have developed the ChemTok tokenizer. They have manually implemented a set of tokenization rules from the training set of the CHEMDNER task (Krallinger et al., 2015) of the BioCreative IV challenge (Arighi et al., 2014) and have shown that when NER classifiers are trained on the output of the ChemTok, they achieve higher performance on the NER task, compared to when they are trained on the output of ordinary rule-based tokenizer available for the biomedical domain. The rule-based approach does not require any *annotated* corpus for training but manual design and fine-tuning the rules are generally time-consuming and labor-intensive.

In supervised learning methods, tokenizer is implemented as a classifier that is trained on an annotated corpus (a set of sentences and their constituent tokens) and learns to determine the segmentation positions in given sentences. To find token boundaries, the tokenizer either splits or joins textual objects through classification to form tokens. A good example is the tokenizer built by Barrett and Weber-Jahnke (2011) for the biomedical domain, which works based on regular expressions in conjunction with machine learning techniques.

An extensive review of the challenges in making efficient tokenizers for the biomedical domain is presented by Cruz Díaz and Maña López (2015). In addition, He and Kayaalp (2006) have systematically compared 13 tokenizers on MEDLINE® abstracts.

Subword tokenization

Subword tokenization is an important NLP pre-processing task that plays a crucial role in modern language processing methods, specially in neural sequence-to-sequence learning tasks such as neural machine translation (NMT), automatic speech recognition (ASR), and text generation. Similarly to lexical tokenization, subword tokenization also deals with segmenting input sentences (or individual tokens) into smaller elements known as *subword units* (e.g., *word-pieces* and *sentence-pieces*), but for different purposes: (1) dealing with infinite vocabularies and rare words of a language/domain and (2) allowing to train efficient deep neural networks (DNN) with less computations.

According to Zipf's law (Zipf, 1949), if we analyze any large corpus and rank all unique words based on their frequency (occurrence count in the corpus), the frequency of each word multiplied by its rank is a constant. For example, in any English corpus, there are some words such as “*the*” that constitute a significant proportion of the corpus while a lot of rare words with very low frequency exist in that corpus. In neural networks that accept natural language text as their input or generate text as their output, each unique word of the vocabulary is typically represented with a unique continuous-space vector known as *word embedding*. For instance, a neural model for biomedical relation extraction may contain 100K unique vectors for 100K unique words such as “*the*”, “*regulation*”, “ *β -arrestin1*” and “*EphA2*”. The word embeddings can be either pre-trained in advance (using unsupervised methods such as `Word2Vec` (Mikolov et al., 2013)) or learned from scratch during the actual neural network training for the specific task at hand. Naturally, there is a restriction on the vocabulary size, based on what can fit into GPU's memory. In other words, existing neural network models incorporate **finite** number of unique words (receiving in their inputs and yielding in their outputs), whereas some NLP tasks such as machine translation or text summarization usually deal with unrestricted (open) vocabulary (e.g. any unknown word may appear in the input text which needs to be addressed for translation). Consequently, word embeddings approach is not suitable for NLP applications that need to deal with open vocabularies. Another major issue with word embeddings is sparsity: neural network models usually cannot learn good representations (embeddings) for the rare words and this can negatively impact the performance of downstream tasks.

There is yet another problem with using word embeddings when dealing with large vocabularies. In neural models that generate text (sequence of words) as their output, in every time-step, a probability distribution over all words of the vocabulary is calculated (using the *softmax* activation function), hence the amount of required computation is correlated with the size of the vocabulary, and larger vocabularies slow down the training and prediction times. To summarize, the word embedding approach suffers from the following problems: (1) it is not a suitable approach for

dealing with unrestricted vocabularies; (2) the quality of embeddings for the rare words is usually poor; (3) word embeddings for large vocabularies may not fit into GPUs with limited memory; and (4) in neural network models that generate text as their output, large vocabularies slow down the training and prediction times.

Three main approaches have been proposed to deal with the aforementioned problems. The first approach is to simply focus only on the most frequent words. This clearly results in a significant number of out-of-vocabulary (OOV) words. Although this has been shown to work efficiently for some NLP tasks (e.g. text classification and relation extraction), it is definitely not ideal for tasks such as machine translation that should deal with open vocabularies. The second approach is to build purely character-level neural models, in which, sentences are represented as sequences of characters (with character embeddings) and not tokens. Even though character-level models have shown good performance on many tasks (for example, dos Santos and Guimarães (2015) and Gridach (2017) have used character embeddings besides the word embeddings to boost the performance of named-entity recognition), in general they demand more computations (compared to word-level models) for training and prediction. The third approach, *subword tokenization*, solves the shortcomings of the previous approaches. The idea here is to define a desired number of *subword units* in advance (N) and training a subword tokenizer on a large corpus. The training is usually unsupervised and the tokenizer automatically detects N best subword units in the given corpus. These subwords form a subword inventory which can further be used for actual segmentation. In other words, after the training and forming the subword inventory, the segmenter can segment any sequence of characters (e.g. a sentence or a token) into its constituent subwords, hence there will be theoretically no OOV words³ thus the neural network models can deal with unrestricted vocabularies.

In this approach, common words of a language/domain are represented with dedicated subword units (e.g. “*regulation*” could be represented with a single dedicated subword), whereas less common and rare words are segmented into their constituent subwords (e.g. “*aminoglycosides*” could be represented as [*“amino”, “gly”, “cos”, “ides”*]).

Similarly to words, each unique subword of the inventory can be represented with a unique continuous-space vector (i.e., a *subword embedding*) and subword embeddings can be either pre-trained in advance or learned from scratch during the actual neural network training for the specific task at hand. By setting a reasonable number for N (e.g. 20K or 30K subword units), all subword embeddings can easily fit into GPU memory and because all words can be segmented into subword units, there will be no problem in sequence-to-sequence learning tasks. In contrast to the

³In practice, if one tokenizes a massive corpus, there might be many rare Unicode characters and therefore some of the rarest characters are simply not included in the vocabulary and thus all subwords that contain them will be actually out-of-vocabulary.

word embeddings approach, the subword embeddings approach does not suffer from producing poor quality embeddings for the rare words because representing a rare word with its constituent subword embeddings is quite efficient. Finally, it should be highlighted that neural models that rely on subword embeddings have outperformed the models that rely on word and/or character embeddings.

Training a subword tokenizer on a relatively large corpus usually takes a few hours on a single computer and tokenization and detokenization (constructing the sentence based on the sequence of subwords in text generation tasks) are also relatively very fast. Without delving into the details, four subword tokenization algorithms and libraries worth mentioning:

1. Schuster and Nakajima (2012) proposed WordPieceModel (WPM) to deal with infinite vocabulary problem in building a speech recognition system for Japanese and Korean languages at Google. Later, it was successfully adopted in Google’s machine translation systems (Wu et al., 2016; Johnson et al., 2017) and Google’s neural language model, BERT (Devlin et al., 2018).
2. Sennrich et al. (2016) proposed byte pair encoding (BPE) subword tokenization algorithm to deal with the translation of rare and unknown words in NMT systems⁴. BPE was later used in OpenAI’s neural language models, GPT (Radford et al., 2018) and GPT-2 (Radford et al., 2019).
3. Kudo (2018) proposed a subword tokenization and regularization algorithm based on a unigram language model to improve the performance of NMT systems.
4. Kudo and Richardson (2018) developed the SentencePiece library, a language independent subword tokenizer and detokenizer that is freely available for public use⁵. SentencePiece implements two subword tokenization algorithms: BPE (Sennrich et al., 2016) and the unigram language model (Kudo, 2018), with the extension of direct training from raw sentences, which allows building purely end-to-end systems that do not depend on any language specific pre-processing. Similarly to WPM, in SentencePiece, whitespace is handled as a normal symbol, hence it can be applied on raw data (any sequence of characters), requiring no prior tokenization and consequently, it can successfully be used for languages that do not use space character between words (non-segmented languages) such as Japanese and Chinese. Subword tokenization in SentencePiece is *lossless*, meaning we can detokenize a sequence of subword units without any ambiguities, making this method ideal for neural network-based text generation systems (e.g. NMT systems).

⁴An implementation of BPE (with slight modifications) can be found in subword-nmt library: <https://github.com/rsennrich/subword-nmt>

⁵<https://github.com/google/sentencepiece>

Defining the desired number of subwords is tricky and more or less domain/task dependent. However, there are a few important points to consider. The ultimate goal for subword tokenization is to find (and subsequently learn *dedicated* embedding vectors for) the common words/subwords of a language/domain/corpus, while still being able to represent all rare words with sequences of existing subwords. For example, since “P53” is very frequent in the biomedical domain, it is generally preferred to learn a dedicated embedding for this word in a neural model rather than representing it as a sequence of separate vectors (e.g. “P” and “53”). This hopefully assists the subsequent layers of the network for addressing the main task at hand (e.g. relation extraction).

Another fact to consider is the negative correlation between the vocabulary size (number of unique subwords) and the average length of the sentence (represented as a sequence of subwords). Recurrent neural networks (including LSTM and GRU networks) have shown better performance on shorter sequences, i.e. they have trouble learning from very long sequences. Therefore, when working with RNNs, it may be interesting to investigate the effect of choosing different sizes for the subword inventory, since it directly affects the average length of the input sequences.

Sentence boundary detection

Many NLP tasks require their input to be segmented into sentences. As mentioned by Xuan et al. (2007), isolating sentences is a prerequisite for any syntactic analysis of texts such as POS-tagging, chunking (shallow parsing) (Jurafsky and Martin, 2009) and constituency/dependency parsing. Other NLP tasks that have such requirement include text summarization, sentence alignment (Brown et al., 1991), machine translation and any NLP task that demands measuring the similarity between the sentences (Achananuparp et al., 2008). Besides task requirements, there are technical reasons for segmenting larger texts into the sentences. Neural networks are typically executed on GPUs, and GPUs have limited amount of memory. Hence, it might be totally impossible to feed a large amount of text (e.g. a whole book) into the GPU memory at once, specially when using deep networks that have millions of free parameters (e.g. BERT (Devlin et al., 2018) encoder⁶).

Sentence boundary detection (also known as sentence boundary disambiguation and sentence segmentation) is an NLP pre-processing task that aims to detect where sentences begin and end in larger bodies of text (e.g. articles or books). In the English language, sentences usually end with period, exclamation mark or question mark. Consequently, sentence boundary detection algorithms focus on punctuation occurrences in the input text and decide which of them denotes a sentence termination. The period is the most ambiguous punctuation because it can occur inside or

⁶BERT_{BASE} has about 110M parameters and BERT_{LARGE} has about 340M parameters.

collocate with almost any words, including abbreviations, numbers, dates, file names, proper names (e.g. protein/gene/chemical names) and titles. Xuan et al. (2007) have analyzed 6 million MEDLINE® abstracts and have found that about 33% of periods are ambiguous.

As discussed by Xuan et al. (2007), sentence boundary detection in the biomedical domain and the clinical domain is more challenging than the general English domain. In the clinical domain, abundance of acronyms and abbreviations, misspellings, punctuation errors and incomplete sentences are the main source of error in sentence boundary detection (Kreuzthaler and Schulz, 2015). After a comprehensive investigation of characteristics of sentence boundaries in biomedical literature, Xuan et al. (2007) have identified the following unique features that differentiate biomedical literature from regular English text for the SBD task.

1. The biomedical literature contains a large amount of abbreviations (e.g. “*E.coli*”, “*i.c.v.*”, “*N. lactamdurans*”, “*Hs.1259*”, “*C. elegans*”, “*S. pombe*”) which can appear anywhere in a sentence and having a large amount of abbreviations with period is a major source of errors in sentence boundary detection.
2. Unlike the general domain, proper names in the biomedical domain (such as gene or protein names) can start with lower-cased letters (e.g. “*hA2aR*”, “*hKv beta 3*”). In addition, it is not uncommon to begin sentences with such proper names (e.g. “*hRCE1 activity was ...*”). In general English texts, when an abbreviation is followed by a period and the next word is a number or lower-cased, that period does not typically denote a sentence boundary (Mikheev, 2000). This assumption does not hold in biomedical literature since there are many cases that a sentence ends with an abbreviation and the next sentence starts with a lower-cased proper name. In addition, in general domain, a common postulation in rule-based approaches stipulates if the word immediately before an ambiguous period is a single uppercase letter then the period does not denote a full-stop. However, biomedical domain includes a lot of gene or protein names that end with a single uppercase letter (e.g. “*cyclin F*”) and it is very common to have such names at the end of a sentence.
3. On one hand, the biomedical domain contains a significant number of compound words, and on the other hand, lack of naming convention has led to a naming chaos, having multiple names all referring to a single entity. Unfortunately, not all names of entities are recorded in the dictionaries. This situation significantly degrades the results of classic POS taggers. As a result, SBD algorithms based on POS taggers will obtain less accuracy on biomedical corpora than on general corpora.

4. Biomedical literature contains a considerable amount of articles that are converted to text from images using optical character recognition (OCR) systems. These systems can make a lot of mistakes during the conversion (e.g. “1.5-3.0 mM” may become “1 . 5-3 . 0 mM”) which complicate sentence boundary detection. In addition, various forms of citations appear frequently in MEDLINE® **abstracts**. These contain a lot of abbreviated author and conference/journal names. General domain sentence boundary detection systems are not usually equipped to handle such cases efficiently.

Sentence boundary detection systems can be broadly divided into three main categories: (1) rule-based systems that utilize hand-crafted regular-expressions, heuristics and dictionaries, (2) supervised learning-based systems that require an annotated corpus for training, and (3) fully unsupervised methods. Finally, it should be mentioned that sentence boundary detection can be performed either before, after or even simultaneously with lexical tokenization. For example, in the UDpipe software (Straka et al., 2016; Straka and Straková, 2017), sentence segmentation and tokenization is performed jointly, using a bidirectional GRU network that for each character in the input text predicts whether it is the last one in a sentence, the last one in a token, or not the last one in a token.

Stemming and lemmatization

Lemmatization is an NLP pre-processing task that converts a word (as it appears in a text) into its dictionary or base form, also known as **lemma**. Lemmatization transforms each noun into its singular form, each verb into its infinitive form, and each adjective/adverb into its base positive form. For example, “sing”, “sings”, “sang” and “sung” are transformed into their common lemma “sing”. Similarly, “regularization” and “regularisation” are both converted into “regularization”, “am”, “is” and “are” are converted into “be”, and “mouse” and “mice” are transformed into “mouse”.

Having knowledge about the morphological rules of the language is important for lemmatization and for this reason, lemmatizers often rely on using a comprehensive lemma dictionary (Shatkey and Craven, 2012). In addition, it is generally required to consider the context of a word for lemmatization, because the part-of-speech (POS) and morphology tags of a word in the context are needed to disambiguate the correct corresponding lemma (Kanerva et al., 2020). For example, the word “lives” in English language can be either a verb (e.g. “He *lives* in Helsinki.”) or a plural noun (e.g. “This study examines the *lives* of the rich and famous.”), with the former being lemmatized into “**live**” and the latter into “**life**”.

Lemmatization acts as a normalization process, mapping all morphological variants of a word into the same underlying lemma, hence all of them will be seen as the same term in the subsequent steps of NLP pipelines. In other words, by reduc-

ing the total number of distinct terms, lemmatization decreases the complexity of the input texts, and therefore brings important benefits to downstream tasks such as information retrieval, information extraction, document classification and clustering and natural language understanding (NLU) (Liu et al., 2012). For example, in relation extraction systems, lemmas (instead or in addition to the original words) can be used when representing the shortest dependency path which connects the two candidate entities in the parse graph. In the GENIA event extraction (GE) shared task of the BioNLP-ST-11 challenge (Kim et al., 2011b), 4 out of 9 participating teams used lemmatization features to improve the performance of their systems (Liu et al., 2011; McClosky et al., 2011; Vlachos and Craven, 2011; Emadzadeh et al., 2011).

Stemming is another NLP pre-processing task which aims to reduce all morphological variants into a single **stem** form by trimming the suffixes. In contrast to lemmatization, stemming does not necessarily generate a valid dictionary form. For example, lemmatization of “*regulated*” results in the base verb form “*regulate*”, whereas stemming may generate “*regul*” as the stem. Stemming can mistakenly reduce words with different meanings into the same stem. For example, “*experiment*” and “*experience*” could be both reduced into “*experi*” (Shatkay and Craven, 2012) and “*activates*”, “*activations*” and “*activities*” could all be reduced into “*activ*” or “*act*” by most stemming algorithms (Liu et al., 2012). In addition, stemming does not typically take the part-of-speech (POS) of the words into account, does not recognize the relationship between “*mouse*” and “*mice*” and can sometimes fail to reduce semantically relevant words into the same stem form (under-stemming).

According to Liu et al. (2012), general domain lemmatizers such as `morpha` (Minnen et al., 2001) or WordNet-based (Fellbaum, 1998) lemmatizers which are not specifically developed for the biomedical domain, often fail to produce the correct lemmas for the biomedical terms. One important reason is that many domain-specific terms (e.g. “*methylation*” or “*phosphorylation*”) are usually not recorded in general English thesaurus such as the WordNet. Consequently, applying a general domain lemmatizer to the biomedical literature results in some loss in performance.

For this reason, BioNLP researchers have either developed lemmatizers that are specifically designed for the biomedical domain, or they have adapted and re-trained general domain lemmatizers on biomedical corpora.

For example, Liu et al. (2012) have developed the `BioLemmatizer` tool. This tool is based on the `MorphAdorner`⁷ toolkit, but is specifically designed and optimized for the biomedical domain, through the integration of several lexical resources related to molecular biology. Another example is the recent work by Ngo et al. (2019). They achieved the first rank in the Structural Annotation Task (dependency parsing) of the CRAFT-2019 shared task (Baumgartner et al., 2019) using an approach that builds primarily on the Turku neural parser (Kanerva et al., 2018), Universal lemmatizer

⁷<http://morphadorner.northwestern.edu/morphadorner/>

(Kanerva et al., 2020), and various domain-adaptation techniques.

Part-of-speech (POS) tagging

The words in any language can be categorized into different grammatical classes known as *parts of speech* (POS). The part of speech of a word indicates how it functions in meaning as well as grammatically within a given sentence. Words in English language can be divided into two major groups: (1) words that always have the same part of speech, regardless of the context (e.g. “methylation” is always a *noun*), (2) words that can have different part of speech, depending on the context they appear in. For example, *work* can be either a *noun* or a *verb* (e.g. “I work in the TurkuNLP group.” v.s. “My work is not easy.”). Dictionaries usually list the most common meanings with their corresponding part of speech for the entries.

The exact number of part of speech categories for a natural language depends on how the language is analyzed by a linguist (coarse-grained v.s fine-grained classes). For example, most English grammars would include at least 8 part of speech categories (*noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection*), whereas the part of speech annotation scheme used in creation of the first version of Penn Treebank (Marcus et al., 1993) includes 36 distinct POS tags⁸ (see Table 1).

The task of recognizing the correct POS class (also known as POS-tag) for the words in a given sentence is referred to as *part of speech tagging* (POS-tagging). Knowing how the constituent words grammatically function in the sentence is very beneficial for syntactic and semantic analysis, hence, POS-tagging is often regarded as a preceding step for lemmatization, parsing, named-entity recognition, relation extraction, speech synthesis, machine translation and many other NLP tasks⁹.

State-of-the-art POS taggers are based on supervised learning (Jurafsky and Martin, 2009), requiring an annotated corpus. The GENIA corpus (Tateisi et al., 2005) (which originally contained 500 manually annotated MEDLINE[®] abstracts) and the CRAFT corpus (Verspoor et al., 2012) (97 annotated full-text biomedical articles) are the most famous and widely used annotated corpora for developing and evaluating POS-tagging systems for the biomedical domain.

POS-tagging systems can be broadly divided into traditional feature-based systems and neural network-based systems. Nguyen and Verspoor (2019) have retrained and evaluated six established POS-tagging systems on GENIA and CRAFT corpora:

⁸In the subsequent versions of the schema, a few basic punctuation marks (such as period, comma, colon, parentheses, dollar, and opening/closing quotation marks) are added as additional POS-tags.

⁹As discussed earlier, modern end-to-end neural NLP systems often differ from multi-stage NLP pipelines in the sense that they accept raw texts as their input and provide the final desired results (e.g., lemmas or named-entity tags) as their output, without relying on or running other NLP tools. Hence, such systems do not depend on POS-tags as features and part of their inputs.

| # | Tag | Description |
|----|------|--|
| 1 | CC | Coordinating conjunction |
| 2 | CD | Cardinal number |
| 3 | DT | Determiner |
| 4 | EX | Existential <i>there</i> |
| 5 | FW | Foreign word |
| 6 | IN | Preposition or subordinating conjunction |
| 7 | JJ | Adjective |
| 8 | JJR | Adjective, comparative |
| 9 | JJS | Adjective, superlative |
| 10 | LS | List item marker |
| 11 | MD | Modal |
| 12 | NN | Noun, singular or mass |
| 13 | NNS | Noun, plural |
| 14 | NNP | Proper noun, singular |
| 15 | NNPS | Proper noun, plural |
| 16 | PDT | Predeterminer |
| 17 | POS | Possessive ending |
| 18 | PRP | Personal pronoun |
| 19 | PP\$ | Possessive pronoun |
| 20 | RB | Adverb |
| 21 | RBR | Adverb, comparative |
| 22 | RBS | Adverb, superlative |
| 23 | RP | Particle |
| 24 | SYM | Symbol |
| 25 | TO | infinitive <i>to</i> |
| 26 | UH | Interjection |
| 27 | VB | Verb, base form |
| 28 | VBD | Verb, past tense |
| 29 | VBG | Verb, gerund or present participle |
| 30 | VBN | Verb, past participle |
| 31 | VBP | Verb, non-3rd person singular present |
| 32 | VBZ | Verb, 3rd person singular present |
| 33 | WDT | Wh-determiner |
| 34 | WP | Wh-pronoun |
| 35 | WP\$ | Possessive wh-pronoun |
| 36 | WRB | Wh-adverb |

Table 1. Penn Treebank POS tagset

GENIA POS-tagger (Tsuruoka et al., 2005), MarMoT (Mueller et al., 2013), NLP4J-POS (Choi, 2016), Bidirectional LSTM+CRF (Huang et al., 2015), Bidirectional LSTM+CRF+CNN-char (Ma and Hovy, 2016), and Bidirectional LSTM+CRF+LSTM-char (Lample et al., 2016). They have reported that all the POS-taggers perform very similarly, achieving $\sim 98\%$ accuracy on GENIA and $\sim 97\%$ accuracy on CRAFT corpus. These results are very impressive

since they are very close to the human agreement on these corpora.

Traditionally, POS-tagging and parsing were closely intertwined because many parsers depended on receiving POS-tags as part of their input. In other words, such parsers utilized the POS-tags as an important feature set. Consequently, one could not swap one POS-tagging system with another one and expect the same performance from parser, unless he re-trained the parser from scratch with the new POS-tags produced by the second POS-tagging system. For this reason, older parsers usually had an internal POS-tagging component and the part of speech tagging was usually performed as a preceding step, by the parser pipeline itself. However, in most modern neural parsers such as UDify (Kondratyuk and Straka, 2019), POS tagging and parsing are done simultaneously as a joint prediction tasks, hence, POS tagging is no longer an essential preceding task for parsing.

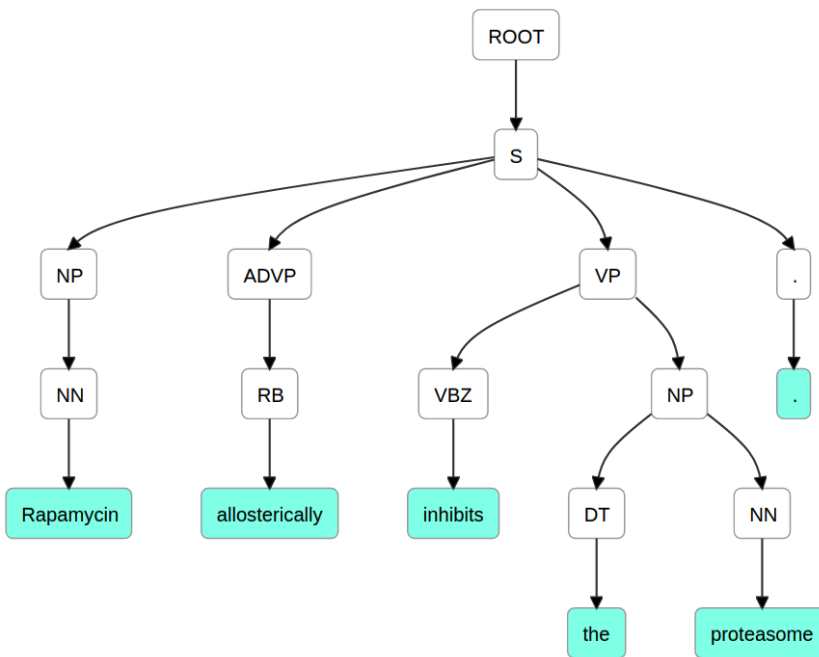
Syntactic parsing

Even though POS-tagging detects the grammatical type for the constituent words of a given sentence (e.g., being a noun, an adjective or a verb), it does not provide sufficient information about the syntactic structure of the sentence. For example, a sentence may contain a number of nouns and verbs, so it is essential to know what the subject(s) and object(s) are for each recognized verb. For this reason *parsing* (*syntactic analysis*) is used.

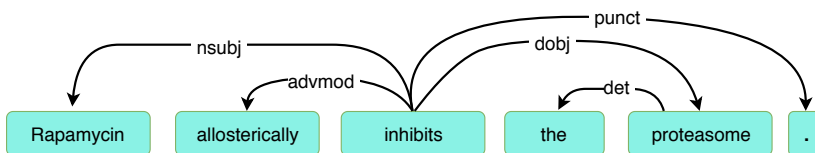
Parsing is one of the most important tasks in NLP which provides corresponding *parse graphs* for the input sentences, based on a *constituency* (*phrase-structure*) or *dependency* grammar. Parse graph representation reveals the grammatical relationships among the constituents (phrases) or the words of the sentence, hence providing beneficial information for downstream tasks such as sentence classification, sentiment analysis or relation extraction¹⁰. In natural language processing, usually two types of sentence parsing are used. In constituency parsing, the aim is to find the hierarchy of constituents (phrases and words) in a given sentence, based on the rules of a context-free grammar and in a recursive manner. In contrast, the aim in dependency parsing is to directly find the grammatical relationships (known as *dependencies*) among the words. Hence, in dependency graphs, words are represented by nodes and dependencies are represented using directed and typed edges. Figure 3 shows a comparison between constituency and dependency parsing for an example sentence from PubMed.

Modern parsers are based on supervised learning, requiring a manually annotated *treebank* (a corpus of sentences with their corresponding parse graphs and POS-tags). It is worth noting that the annotation schema and the grammar used in the creation of a treebank dictates how the parsers trained on that treebank should work. Similarly

¹⁰I highlight that modern end-to-end neural NLP systems may not depend on parse graph information as their features.



(a) Constituency parse graph



(b) Dependency parse graph

Figure 3. Comparison of different parsing methods for an example sentence: "Rapamycin allosterically inhibits the proteasome."

to POS-tagging, the parsers that are trained on a general domain corpus will not perform well on biomedical texts. For this reason, parsers should be re-trained on biomedical treebanks such as the GENIA corpus (Tateisi et al., 2005).

An extensive review and comparison of modern parsers that are developed for the biomedical domain (during the CRAFT 2019 Shared Tasks) is presented by Baumgartner et al. (2019). In addition, Nguyen and Verspoor (2019) have performed an empirical study and compared state-of-the-art feature-based and neural network-based dependency parsers on two benchmark biomedical corpora, GENIA and CRAFT.

Co-reference resolution

Co-reference resolution is the task of clustering mentions in text that refer to the same underlying real world entities. It plays a crucial role in language understanding and can considerably affect the performance of NLP tasks such as relation extraction and question answering.

In relation/event extraction tasks, performing co-reference resolution can increase the recall of detected events. For instance, in the following example from Li et al. (2018a), we notice a binding relation between “Grb2” and “Shc” is stated in the text, and the pronoun “It” refers to the mention “Grb2”:

“... Grb2 binds directly to the EGFR at Y-1068. It also binds Shc ...”

In this example, “Grb2” and “Shc” entities are located in separate sentences. Hence, any relation extraction system that is not trained to detect cross-sentence relations will fail to detect this binding relation. However, by running a co-reference resolution system on the text we can for example replace the pronoun “it” with “Grb2”, thus helping relation extraction systems to detect the relation.

Co-reference resolution systems can be broadly divided into traditional feature-based systems that heavily rely on parsing features and modern neural network-based system, both working based on the supervised learning approach. The BioNLP Protein Coreference dataset (Nguyen et al., 2011) and the CRAFT corpus (Cohen et al., 2017) are two annotated corpora that are built for developing co-reference resolution systems for the biomedical domain.

Named-entity recognition

Named-entity recognition (NER) is an important task in NLP that deals with finding mentions of real-world entities (e.g. people or organization names) in input texts. In the biomedical domain, the types of named entities that are of interest include genes, proteins, chemicals, drugs, diseases and disorders, chromosomal locations, cell lines, and cell types.

Named-entity recognition (in conjunction with entity linking) can be performed as an optional pre-processing step for information retrieval. By finding the occurrences of named entities in a collection of documents (and associating them with canonical forms or database identifiers¹¹) and indexing the documents based on the entities they contain, a search engine will be able to answer complex user queries and retrieve the documents that discuss a particular named-entity, such as a particular gene or protein of interest. Of course, one can alternatively use the full-text indexing approach, but it usually leads to a lower precision and recall for the retrieved documents.

Named-entity recognition is an indispensable preceding step for relation and

¹¹See named-entity normalization in the next section.

event extraction since these tasks by definition focus on the detection of semantic relations among named entities in the given texts. In other words, named entities are typically known and given as part of the input in event and relation extraction tasks¹².

According to Shatkay and Craven (2012), there are a number of reasons that make biomedical named-entity recognition difficult. For example, in the case of recognizing gene and protein names:

1. Some gene and protein names are short. These names are sometimes called gene *symbols*. For some species, gene symbols are actually abbreviations for longer, more descriptive names (e.g. “BRCA1” is short for “breast cancer type 1 susceptibility protein”). The longer forms for some species are very common in the literature.
2. The names for some genes are homonyms of ordinary English words. For example, “And”, “lot”, “stuck”, “lush”, and “dreadlocks” are a few gene or protein names for the fruit fly.
3. Some gene names coincide with phrases that have common meanings in English language (e.g. “cheap date”, “onion rings” or “Sunday driver”) or have whimsical interpretations in English (e.g. “pray for elves” and “sonic hedgehog”).
4. It is very common in the biomedical domain that same name is used for a gene and its products. Hence, for each mention in the text, NER systems should decide whether this particular mention refers to the gene, to its protein product, its transcribed RNA, or a combination of these possibilities. This ambiguity is a complicating factor for the NER systems that have to make a distinction among such alternatives.
5. Some protein names are composed of other protein names. For example, “MAP kinase 8” and “MAP kinase 8 interacting protein 1” are different protein names. Similarly, “MAP kinase 1” and “MAP kinase kinase 1” are distinct protein names.

The aforementioned examples show that biomedical NER is a challenging task and simple dictionary matching or rule-based approaches may fail to provide high precision and recall. Currently, state-of-the-art NER systems are based on supervised learning approach and utilize conditional random fields (CRF) in conjunction with

¹²In end-to-end neural relation or event extraction systems, named-entity recognition and relation/event extraction are done jointly. For example, DeepEventMine system (Trieu et al., 2020) extracts nested biomedical events from raw texts, without any pre-processing needed.

deep neural networks. For example, in the recent PharmaCoNER track (Pharmacological Substances, Compounds and proteins Named Entity Recognition) (Gonzalez-Agirre et al., 2019) of The BioNLP Open Shared Tasks 2019 (Jin-Dong et al., 2019), Xiong et al. (2019) achieved the first rank in the competition by using a classifier that utilized a CRF layer on top of a pre-trained BERT (Devlin et al., 2018) encoder. There have been a number of shared task challenges that have provided manually annotated data to support the development of NER systems for a variety of biomedical entities ranging from mentions of genes, proteins, cell lines and diseases to anatomical entities. Recently, Crichton et al. (2017) have done an impressive work and gathered 15 different biomedical NER corpora into a single repository¹³ and they have unified all corpora by conversion to the CoNLL format. This facilitates the development and evaluation of new NER methods for the biomedical domain. For further discussions about the recent advancements in the deep learning-based NER methods, refer to Yadav and Bethard (2018) and Li et al. (2018b).

Named-entity normalization

The typical output of an NER system consists of the type and beginning/end offsets for each entity that is detected in the input text. For many text mining applications this output is not sufficient, particularly when we need to know *which* real-world entities are actually mentioned in the given texts. For instance, in automatic creation or curation of gene-related databases, we need to be able to retrieve a set of sentences that discuss a particular gene. By running a gene NER software on PubMed abstracts, we obtain the offsets of gene mentions in each document, but we still do not know which genes are actually referenced in those documents. To address this problem, named-entity detection is often followed by named-entity normalization (also known as entity linking). Entity normalization is an NLP task which entails associating each recognized entity mention with a canonical identifier for the entity being referenced (Shatkay and Craven, 2012).

Entity normalization in the biomedical domain is often challenging. For instance, in gene (or protein) entity normalization tasks, we usually face the following problems:

1. Some names and symbols in the biomedical domain are polysemes for multiple genes. Similarly, there are many cases in which the same name is used for related genes (or proteins) in different species. For example, “p53” is used for any isoform of a protein encoded by homologous genes in various organisms, including “TP53” (humans) and “Trp53” (mice). Consequently, it is necessary to first recognize which organism(s) is discussed in a given text, in order to assign the correct identifier(s) to “p53” mentions in that text.

¹³<https://github.com/cambridgeltl/MTL-Bioinformatics-2016>

2. It is often the case that there are several different names for the same gene and a multitude of typographical variants for each one. For instance, the gene “tumor necrosis factor” has other names such as “tumor necrosis factor alpha”, “TNF”, “TNF alpha” and “TNF α ”. When maintaining gene-related databases, it is necessary to map all different mentioned names into a single standard name (or database identifier), to be able to retrieve all relevant documents that discuss a particular gene, regardless of how the gene is mentioned in the texts.
3. Sometimes authors paraphrase or use permutations instead of using standard gene names. For instance, “ciliary neurotrophic factor receptor” may be referred to as “receptor for ciliary neurotrophic factor”. To be able to retrieve all relevant documents that discuss a particular gene, it is necessary to detect and map all variants into a single standard name (or database identifier) when indexing the documents.

In the biomedical domain, gene entity mentions are usually normalized into NCBI Entrez identifiers (Benson et al., 2013; Maglott et al., 2011), protein mentions are normalized into UniProt entry names or accession numbers, organism mentions are normalized into NCBI Taxonomy identifiers (Federhen, 2012), chemical mentions usually into ChEBI (Chemical Entities of Biological Interest) identifiers (Hastings et al., 2016) or NCBI PubChem identifiers (Kim et al., 2019), and disease mentions into MEDIC concepts (Davis et al., 2012).

Since normalization requires mapping all possible variants into unique names/identifiers (e.g. ontology concepts or database identifiers), a common approach is to compile and utilize a dictionary that associates each canonical identifier with all possible names and symbols. For example, a dictionary that associates each NCBI Entrez identifier with all possible variants of gene names and symbols. Unfortunately, simple dictionary lookup or simple string matching algorithms do not work very well in the biomedical domain because it is often the case that there are numerous ambiguous mentions in each article that can be either mapped into *more than one unique identifier* or *not mapped to any identifier* at all. Currently, state-of-the-art entity normalization methods work based on supervised learning and learning surface and semantic similarities between mentions and concept names, directly from the training data. A simple approach is to first represent both mentions and the names as weighted Term Frequency-Inverse Document Frequency (TF-IDF) vectors and then train a classifier (e.g. a SVM classifier) which learns to associate each mention to the correct corresponding name (for example, see **Paper III**). Alternatively, we can learn various semantic vector representations for mentions and the terms, and use the cosine similarity measure to calculate the similarity of a mention with all possible names and choose the best candidate name. Finally, a variety of neural network-based methods have been recently proposed for entity normalization by various authors (Li et al., 2017; Deng et al., 2019; Ji et al., 2019; Tang et al., 2021).

2.2 Relation and event extraction

Among different tasks, biomedical relation and event extraction tasks have received much attention in the BioNLP community. These foundational information extraction tasks deal with automatic identification of biological processes, interactions and relations described in biomedical literature. Precisely speaking, biomedical relation and event extraction systems can scan through a vast amount of biomedical texts and automatically detect and extract the semantic relations of biomedical named entities. The structured outputs of such systems (i.e., the extracted relations or events) can be stored as relational databases or molecular interaction networks which can easily be queried, filtered, analyzed, visualized and integrated with other structured data sources.

While biomedical NER and entity normalization systems can find mentions of real-world entities in given texts, biomedical relation and event extraction systems can extract semantic relations among the detected entities, thus providing additional information to biomedical researchers. For instance, while it might be interesting for researchers to know which PubMed Central articles discuss a particular protein (addressed by NER and entity normalization), what the researchers are usually interested about is obtaining information about the interactions of that particular protein with other entities (such as proteins or chemicals), and this can be addressed by relation extraction systems.

Biomedical relation extraction has traditionally focused on pairwise relations, i.e., the interactions or relations between two candidate named-entities. In these cases, each input example to the system is a text (e.g. a sentence or a paragraph) with two candidate named entities and the system predicts whether the text states any relations between the two entities (see Figure 4). As discussed in Section 1.1, depending on the task definition and annotation schema, pairwise relations can be either typed or untyped and they can be either directional or non-directional. Hence, a relation extraction task can be cast into a binary, multi-class or multi-label classification problem, predicting the existence, type and possibly the direction of the relation(s) for each candidate named-entity pair in the given input texts.

A brief review on relation extraction methods

Early relation and event extraction systems were based on rules or patterns. Such rule-based or pattern-matching approaches can lead to low recall and/or precision, because it is difficult to come up with a comprehensive set of rules that can cover all possible ways that biomedical relations can be discussed in the literature, without making many mistakes. State-of-the-art relation and event extraction systems are thus based on supervised machine learning, relying on manually annotated data to train a classifier (e.g. an SVM, an ANN or a Naive Bayes classifier) capable of

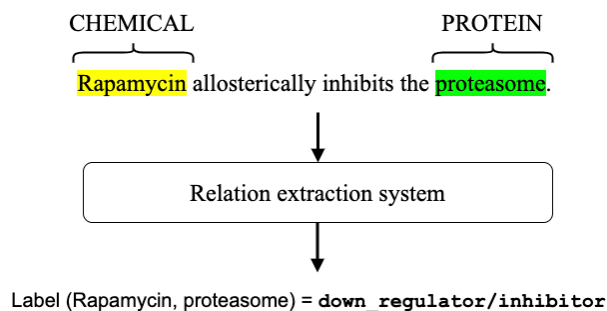


Figure 4. Chemical-protein relation extraction example. The input is a sentence with the two entities, and the output is the predicted label for the pair.

detecting statements of relations in texts.

According to Zhang et al. (2017), supervised relation extraction methods can be broadly divided into three main groups: (1) feature-based methods, (2) kernel-based methods and (3) deep learning-based methods¹⁴.

Feature-based methods extract a series of relevant features from the text in order to train a relation extraction classifier. In these methods, each entity pair is represented with a corresponding numerical feature vector that is further used for either training the classifier or for detection of the relation(s) (Zhang et al., 2017). The list of features usually includes (but is not necessarily limited to) bags-of-words/lemmas/POS/DTs or their n-grams in the sentence or along the shortest dependency path (SDP). The Turku Event Extraction System (TEES) (Björne, 2014)–previously developed by members of our research group–is an example of such a system, using a rich set of features to train an SVM classifier. TEES achieved 62.99 F-score in the DDI-2011 task (Segura-Bedmar et al., 2011), 58.7 F-score in the DDI-2013 task (Segura-Bedmar et al., 2013), and state-of-the-art performance (42.00 F-score) in the Bacteria-Biotope 2013 relation extraction task (Deléger et al., 2016). Another example is the VERSE system, developed by Lever and Jones (2016), which obtained the state-of-the-art result with an F-score of 55.8 in the Bacteria-Biotope 2016 relation extraction task (Deléger et al., 2016). Similarly to TEES, VERSE also extracts a rich set of features in order to train a linear SVM, but utilizes a fea-

¹⁴While the mentioned categorization is pretty common in NLP community, one could correctly argue that a distinction between feature-based and neural/deep learning-based methods lacks precision, because deep learning-based methods can still use manually engineered features (e.g., features that are extracted from sentence parse graph), thus the actual distinction should be made between *manually engineered* versus *automatically learned* features instead of methods or systems. While this is very correct, I prefer to use the mentioned categorization of the methods, because my aim in this section it to provide a historical perspective about how different approaches for building relation extraction *systems* have emerged, instead of focusing on different features that have been used in such systems, being manually engineered or automatically learned.

ture selection component for optimization. Finally, Raihani and Laachfoubi (2016) achieved the impressive F-score of 71.14 on the DDI-2013 corpus with a feature-based system utilizing lexical, phrase, verb, syntactic and auxiliary features.

Kernel-based methods use kernel functions that are able to directly calculate the similarity between two instances (i.e. two machine learning examples) to train a relation classification model (Zelenko et al., 2002; Zhang et al., 2017). In kernel methods, examples retain their original representation (e.g. as bag-of-words in the sentence, sentence dependency parse graph or sentence shallow parse graph) and the kernel method is able to assign a label to a given novel example by computing and comparing its similarity to all labeled training set examples (Zelenko et al., 2002; Culotta and Sorensen, 2004; Zhang et al., 2017). An advantage of kernel methods is that they can search a feature space much larger than could be represented by a feature-based approach, because the kernel functions can explore an implicit feature space when calculating the similarity between two examples (Culotta and Sorensen, 2004). Kernel functions are usually used in conjunction with classifiers like support vector machines and voted perceptron (Freund and Schapire, 1999). Several kernel functions have been suggested and applied for relation extraction. In a bag-of-features kernel approach, the words in the sentence are divided into three groups: before, between and after the two entities. Each group is further represented with a bag-of-features. Mooney and Bunescu (2006) used this approach to build three subsequence-kernels for each bag, with the final kernel function being simply the sum of the three kernels, which is further used with an SVM classifier for relation classification. Another popular family of kernels are tree/graph kernels. Zelenko et al. (2002) developed kernels capable of comparing the similarity of shallow parse trees and used them with SVM and voted perceptron classifiers for relation extraction. Culotta and Sorensen (2004) extended the previous work by introducing the Dependency Tree Kernel for relation extraction and showed that their model outperforms bag-of-words kernel approach by 20 percentage points in F-score. Reichartz et al. (2009) developed the All-Pairs Dependency Tree Kernel, and the Dependency Path Tree Kernel and showed their kernels with richer structural features significantly outperform all published approaches for kernel-based relation extraction from dependency trees. Finally, Airola et al. (2008) developed the All-Paths Graph Kernel for biomedical relation extraction and showed that their method achieves the state-of-the-art performance on five protein–protein interaction corpora.

Feature-based methods rely extensively on natural language processing tools (e.g. tokenizers, POS taggers, lemmatizers, syntactic parsers, etc.) and require heavy feature engineering to transform the input data into a *representation* (i.e. a feature vector) that can lead to a successful relation classification. On one hand, feature engineering is skill-dependent and time-consuming (Zhang et al., 2018), on the other hand, the errors in the NLP tools are amplified in the relation extraction systems, negatively impacting their performance (Zhang et al., 2017).

In contrast, the aim in **deep learning approach** is to *automatically learn* efficient representations, suitable for the relation classification task at hand. Deep learning achieves this by introducing representations that are expressed in terms of other, simpler representations and allowing the computer to automatically learn complex concepts out of simpler concepts (Goodfellow et al., 2016). For example, the representation of a sentence can be expressed by phrases, while phrases are composed of words and syntactic dependencies among them. This allows a modular design and training of a hierarchy of representations, with the root (of the hierarchy) as the final representation used for a prediction task. A key feature is that lower-level representations (i.e. *embeddings*) can be pre-trained in advance, in an unsupervised fashion and with training data other than the training data available for the prediction task at hand. A successful example is pre-trained *word embeddings*, the vector representations for words in a language that are trained on millions of unannotated sentences, so that words with similar meanings have similar corresponding vectors in the vector space model (Mikolov et al., 2013). Several studies have shown that integrating pre-trained word embeddings into deep neural networks (DNN) can improve the performance of downstream prediction tasks.

Deep learning-based relation extraction methods have recently outperformed feature/kernel-based methods on different corpora. For example, on the DDI-2013 corpus (Segura-Bedmar et al., 2013), all top performing methods are based on DNNs (Zhang et al., 2018). The only exception is the feature-based system of Raihani and Laachfoubi (2016) with 71.1 F-score, on par with the recent deep learning-based methods. In earlier approaches, recurrent neural network (RNN) and convolutional neural network (CNN) were the two main neural structures that have been extensively utilized in DNNs for achieving state-of-the-art performance in various NLP and text mining tasks, including syntactic parsing, sentence classification, sentiment analysis, text summarization, machine translation, named-entity recognition and relation extraction. CNNs are inherently efficient in learning *local* or *position-invariant* features through discrete convolution with different size filters (also known as kernels¹⁵), because they extract the features based on n-grams of the sentences. In contrast, RNNs can directly model sequential data, such as the sequence of words in sentences. Long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) networks (Cho et al., 2014) are variant of RNNs that utilize memory cells and/or gating mechanisms to deal with the vanishing or exploding gradients (Bengio et al., 1994), a problem associated with vanilla RNNs which negatively impacts their training and prediction performance.

Yin et al. (2017) have systematically compared the performance of CNNs with LSTMs and GRUs on various NLP tasks and have shown that the performance of CNN and LSTM/GRU networks are very close for relation extraction on the

¹⁵CNN kernels should not be confused with kernels in kernel-based methods.

SemEval-2010 corpus (Hendrickx et al., 2010). For example a literature survey on DDI-2013 corpus shows that Sun et al. (2019) have achieved 75.48 F-score with a novel recurrent hybrid convolutional neural network (RHCNN) architecture; Lim et al. (2018) have achieved F-score of 73.5 with an ensemble of Tree-LSTMs; Zhou et al. (2018) have achieved 73.0 F-score with position-aware attention-based bidirectional LSTM networks and multitask learning; Zhang et al. (2018) have achieved 72.9 F-score using hierarchical bidirectional LSTM networks. The dependency-based CNN developed by Liu et al. (2016) has achieved 70.8 F-score, the multi-channel CNN developed by Quan et al. (2016) has achieved 70.21 F-score and the Syntax CNN (SCNN) developed by Zhao et al. (2016) has achieved 68.6 F-score.

I highlight that the performance of a neural relation extraction system does not boil down only to the neural network architecture it uses, but also the inputs it receives, the feature set that it uses and the training and optimization procedures that are used to build the system. In addition, there are certain tried-and-true creative techniques that can be used to improve performance. For example, when working with unbalanced data sets, if the proportion of negative examples is considerably higher than of positive examples (which often happens to be the case in many biomedical relation extraction corpora), one can utilize *negative sub-sampling* (also known as *negative instance filtering*) to make a balanced data set. Another innovative method is utilizing *multi-task learning* methods to improve the performance. One main advantage of DNNs is their ability to learn from a combination of several different prediction tasks, each contributing in a unique manner to a single shared representation of the input (Crawshaw, 2020). In this manner, the networks can act as a method to fuse information about individual objects from vastly varied data and hold promise of major performance improvements in problems where there is only a limited amount of training data for the primary task at hand. Hence, multi-task learning setup is ideal for building efficient biomedical relation extraction systems, when actual training data for the main task is limited, but several relevant *proxy* or *auxiliary* tasks can be found, for which sufficient training data already exist. The general idea here is to train a DNN with several input and output layers at once, with a shared internal representation, each input/output pair contributing to this joint representation. In the field of natural language processing it has been demonstrated that even relatively different tasks can contribute to the learned representation and support rather than fight each other, resulting in more robust representations and higher accuracy in downstream applications.

Another established method for improving the performance is a *system combination* approach. In this approach, *different* relation extraction systems are trained on the *same* training data and their outputs (i.e. predictions) are aggregated to provide a final set of predictions for the task at hand. Although all individual systems are trained on the same data, since they can rely on different sets of features or different neural architectures, they have an opportunity to learn different aspects about

the complicated relation extraction task at hand, hence by aggregating their results we can hope to achieve higher performance for the task. System combination has been a very popular approach for biomedical relation extraction. For example, in the BioCreative VI ChemProt relation extraction shared task (Krallinger et al., 2017), the first rank with 64.10 F-score has been achieved by Peng et al. (2018) with a system combination approach. Their method is an ensemble of three separate systems: (i) a CNN-based relation extraction system that receives the sentence sequence and the SDP sequence as inputs, (ii) an RNN-based system that utilizes a bidirectional LSTM network to learn from the full sentence sequence and (iii) an SVM-based system that generates features based on the full sentence and SDP. To aggregate the predictions of the three systems, they trained a random forest (RF) classifier which relies on prediction confidence scores as features. We also participated in this shared task with a similar system combination approach that utilized an SVM-based system and a deep learning-based system that utilized SDP tokens and we achieved the third rank in the task (60.99 F-score) (Mehryary et al., 2017a). After the shared task we substituted our neural model with a better architecture that utilized SDP *and* full sentence tokens and improved our previous results by 2.11 percentage points, achieving 63.10 F-score (**Paper V**).

Recently, the introduction of transformer-based language representation models such as BERT (Devlin et al., 2018), GPT (Radford et al., 2018) and GPT-2 (Radford et al., 2019) impacted the field of natural language processing and resulted in unprecedented performance improvements on many data sets. Such encoders can be pre-trained in advance on large corpora and then fine-tuned with the actual training data for a particular task at hand. For instance, by pre-training a BERT encoder on PubMed abstracts and fine-tuning it with a decision layer on the ChemProt data, Lee et al. (2019) achieved an impressive F-score of 76.46, improving the best previous result of Krallinger et al. (2017) by more than 12 percentage points. With significant performance improvements that have been achieved with transformers, it seems unlikely to see another state-of-the-art method based on simple RNNs or CNNs on any data set.

2.3 Performance Metrics

In this section, I will briefly discuss the evaluation metrics that are commonly used for assessing the performance of relation and event extraction tasks.

Pairwise relation extraction tasks are usually cast to binary classification problems. For instance, in the Bacteria Biotope relation extraction shared task (BB3-event) (Bossy et al., 2013), for each (*Bacteria*, *Habitat*) named-entity pair, the aim is to detect whether the mentioned *Bacteria* lives in the mentioned *Habitat* (a *positive* label) or not (a *negative* label) according to the given input text.

When measuring the classification performance against a validation (or test) set,

we investigate how many examples of each class are correctly classified and how many of them are misclassified. These numbers are typically represented with a *confusion matrix*, or equivalently with the number of *true-positives (TP)*, *true-negatives (TN)*, *false-positives (FP)* and *false-negatives (FN)*. Calculating these values is usually followed by calculating three performance metrics: *precision*, *recall*, and *F-score*.

In binary relation extraction, precision demonstrates *the fraction of extracted relations that are actually correct*, while recall (also known as sensitivity) shows *the fraction of correct relations that are actually extracted* (Equation 1).

$$Precision = \frac{TP}{TP + FP}, \text{ Recall} = \frac{TP}{TP + FN} \quad (1)$$

For example, when measuring the performance of a relation extraction system against BB3 test set, precision denotes how many of the extracted relations are actually correct (i.e., the mentioned bacteria lives in the mentioned habitat), while recall denotes how many of the total relations in the test set, the system has been able to extract.

Usually there is an inherent *trade-off* between the precision and recall, because intuitively speaking, if we want to increase the recall and find more instances of something, we are probably going to make more mistakes, which can lead to a lower precision. Choosing which performance measures should be prioritized, depends on the application. For example, there could be some scenarios when we are looking for extremely high recall results and/or we have some post-processing methods that can later filter out the noise. In contrast, there could be some other scenarios in which we want to be extremely conservative, hence we prefer a very high precision level, even though we know the system will fail to detect many of the relevant instances.

F-measure (also known as *F-score*) is a combined measure to assesses the precision-recall trade-off. It is the *weighted harmonic mean* of precision and recall and it is often used for as a single measure for ranking different machine learning methods (Equation 2).

$$F = \frac{1}{\alpha \frac{1}{Precision} + (1 - \alpha) \frac{1}{Recall}} = \frac{(\beta^2 + 1) * Precision * Recall}{(\beta^2 * Precision) + Recall} \quad (2)$$

The weight in the F-measure formula allows us to control how much we would like to emphasize precision or recall. This can be used for classifier optimization, as we can give more weight to precision (or recall) in the formula and optimize the classifier against that formula. With $\alpha = \frac{1}{2}$ (or equivalently $\beta = 1$), we keep a balance between precision and recall and the resulting formula is called *F1-measure* or *F1-score*. This is the most common performance measure for evaluating and optimizing classifiers since it gives the same weight to precision and recall. Two

other common measures are F_2 -score (*i.e.*, $\beta = 2$) which weights recall higher than precision, and $F_{0.5}$ -score (*i.e.*, $\beta = 0.5$) which emphasizes precision more.

When doing *multi-class classification*, we can similarly compute precision and recall separately for each class. However, to assess the overall performance of the whole system, we need measures that combine the separate measurements. The two common ways to combine the results are called *macro-averaging* and *micro-averaging*.

macro-averaging

In macro-averaging, performance is computed per-class, then the results are averaged over the classes (Equation 3). For example if there are 100 different classes, we have to compute 100 precision scores and 100 recalls and then average them all.

$$Precision = \frac{\sum_{i=1}^{|C|} precision_i}{|C|}, \quad Recall = \frac{\sum_{i=1}^{|C|} recall_i}{|C|} \quad (3)$$

In macro-averaging each class participates equally, because this metric puts emphasis on good performance for all classes, even if they are very small. Thus, macro-averaged precision and recall will be high if the classifier performs very well for all of the classes, regardless of how common they are. In other words, macro-averaging does not weigh by the class sizes, therefore small classes matter as much as large classes and they affect the results equally much as large classes.

micro-averaging

In micro-averaging, individual decisions are summed over all classes. In other words, we collect decisions for all classes into one *single* confusion matrix, and then calculate precision and recall for the confusion matrix (Equation 4).

$$Precision = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FP_i}, \quad Recall = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FN_i} \quad (4)$$

As we see, the micro-average is dominated by the score on common classes. In fact micro-averaging provides a measure of overall performance of the system. Hence it is often used *if the goal is to maximize the number of correct predictions regardless of class*. For example, assume we have five classes, and in the test set, classes 1 to 4 have only 10 instances each, while class 5 has 3000 instances. In micro-averaging if the classifier predicts class 5 very well, we will have very high overall precision and recall, even if the classifier performs very poorly for the other classes, since this metric shows the performance of the system in terms of overall correct predictions of instances, regardless of class.

Finally, in both cases (micro/macro averaging), the F-score is calculated using the micro/macro averaged recall and precision, with Equation 2.

3 Research in brief

In this chapter, I will very briefly present my research papers. Since the reprints of the original papers are included in the thesis, the aim of this chapter is to highlight the main contributions of each paper with regard to the research questions of the thesis and report the results, without discussing every detail of the methods. Instead of presenting research papers one by one, I have organized this chapter by topics, each section corresponding to one of the research question discussed in Chapter 1.

3.1 Optimizing large-scale event databases

In this section, I present the contributions of the thesis in response to **research question 1**: Improving the quality of extracted data in automatically generated event and relation databases.

As discussed in Section 1.2, application of biomedical event extraction systems at large scale, e.g., the totality of PubMed abstracts and PMCOA articles, can result in poor precision and recall, because these corpora contain a large number of documents that are partly or entirely out-of-domain for these systems, being unlike the manually annotated shared task data on which the systems have been trained. Increasing the recall requires running a better event extraction system on all previously processed documents, hence it might not be a practical solution for institutions that have limited access to computational resources. In this section, I will briefly present **Paper I** and discuss a few supervised and unsupervised methods we have developed that can be executed in post-processing to automatically find and remove incorrect events from the output of event extraction systems (e.g., from large-scale event databases or molecular interaction networks), thus improve the precision of such resources without sacrificing recall. This can increase the perceived credibility of such resources for modern biology research.

3.1.1 Introduction

In biomedical event extraction, the extraction of every event is based on the recognition of an occurrence of a *trigger word* in the underlying sentence. Trigger words are textual spans expressing the biological processes underlying the events. For example, if the sentence states “*GENE_A is regulated by GENE_B.*”, the word “regulated”

is the trigger word and based on this, a regulation event with two event arguments (GENE_B being the *regulator* and GENE_A being the *regulatee*) is extracted.

A single unique trigger word, such as *modify*, may have a number of occurrences in the data, acting as a trigger for many events. It is important to note that these events may be of different types. For instance the trigger word *expression* acts as a trigger for both *gene-expression* and *transcription* events, depending on the context.

There are a number of ways to approach the problem of finding incorrect events in large-scale event databases, but a study published by Van Landeghem et al. (2013b) points out that a large portion of the false event predictions originates from the trigger detection phase, i.e. false identification of text spans as trigger words which can lead to the generation of incorrect events. It is worth noting that top-ranking event extraction systems are based on machine learning (Bossy et al., 2011; Kim et al., 2011b; Nédellec et al., 2013), meaning that they do not rely on a list of “safe” trigger words (which could lead to a low recall). Instead, any word can be predicted as a trigger word, and this occasionally leads to wildly incorrect predictions and that lowers the precision of the event databases. Hence, in this work we focus on event triggers with the objective of automatically identifying *obviously incorrect* triggers (i.e., the words that can never act as triggers, regardless of their contexts). By finding incorrect trigger words, we can then remove all events that are extracted based on those triggers and increase the precision of the extracted data.

3.1.2 Data

There are a number of large-scale biomedical event databases but in this work, we mainly focus on the EVEX resource¹ (Van Landeghem et al., 2013a), containing 40,190,858 events of 24 different types such as *binding*, *positive-regulation*, *negative-regulation*, and *phosphorylation*. EVEX events are extracted using the TEES system (Björne, 2014) from 6,392,824 PubMed abstracts and 383,808 PMC-OA full-text articles that were published up to 2012 and which contain at least one gene/gene-product mention.

In total, there are 137,146 unique event triggers that are not comprised purely of numbers and not containing unicode special characters. Different trigger words have event frequencies in the system ranging from 1 to 3,909,759 corresponding events for the trigger word “expression”. As expected, the vast majority of events in EVEX correspond to a small number of highly frequent trigger words. For instance, there are only 3,391 trigger words with frequency above 300 (i.e. corresponding to at least 300 event occurrences), but these words account for fully 97.1% of all events in EVEX. We thus first focus on these highly frequent triggers since finding incorrect triggers among them can have a huge impact on the events.

¹<http://evexdb.org/>

3.1.3 Method

In this section, I will briefly discuss the methods that we have developed for finding incorrect trigger words. These methods are discussed in **Paper I** in detail.

Unsupervised method

Our unsupervised method is based on hierarchical/agglomerative clustering of the 3,391 top-most frequent trigger words, which results in obtaining the binary cluster tree with many coarse/fine grained sub-clusters that contain trigger words with biologically similar meaning. For clustering, we induce a vector space representation for the trigger words, and hierarchically cluster the triggers based on this representation. *Cosine similarity* is used as the clustering metric with the *Ward's variance minimization* algorithm defining the distances between newly formed clusters. To build the vector space representations, we train a `word2vec`-based model (Mikolov et al., 2013) on biomedical literature (details are discussed in **Paper I**). After the clustering, the leaves of the tree present individual trigger words and the root is the group containing all trigger words.

We prepare a list of *safe* trigger words from the BioNLP ST-09, BioNLP ST-11, and BioNLP ST-13 (Kim et al., 2009, 2011b; Nédellec et al., 2013) training and development sets and then try to match EVEX triggers (in the tree) against these triggers and mark them as *safe* or correct. We developed an approximate matching algorithm that relies on splitting multi-token trigger words, lemmatization of the parts, removing certain prefixes and suffixes and a manually engineered stop list. Our matching algorithm resulted in marking 1,483 EVEX trigger words in the tree as safe. The 1,908 remaining triggers in the tree were marked as *possibly incorrect*. At this stage, leaves (trigger words) in the binary cluster tree are marked as being either *safe (correct)* or *possibly incorrect*. Table 2 shows a few example words from EVEX triggers that were matched against shared task trigger words, parts, or lemmas.

| EVEX trigger word | Shared task trigger word/Part/Lemma |
|-----------------------|-------------------------------------|
| co-transcribed | transcribed |
| calcium-induced | induced |
| co-immunoprecipitates | immunoprecipitate |
| downregulating | downregulate |
| recognise | recognize |
| preceding | precede |
| analyzing | analyse |

Table 2. Examples of matching EVEX trigger words against Shared Task exact trigger words or their corresponding parts/lemmas

Finally, we developed a recursive bottom-up algorithm which processes the tree

by starting from the leaves and moving up to the root. The algorithm prunes all sub-clusters that were composed of **only** “possibly incorrect” trigger words. Thus, our pruning algorithm removes a *proportion* of possibly incorrect trigger words, those that are grouped together in different sub-clusters. After the pruning, all remaining trigger words were marked as safe or correct. The clustering and pruning algorithm results in finding more correct EVEX triggers, comparing to what we had originally been able to match against the shared task triggers.

Manual annotation method

As it will be discussed in the results section, the unsupervised method mistakenly removes a number of correct trigger words which lowers the recall. As an alternative method, we gave the list of 3,391 EVEX triggers to an annotator with prior experience in biomedical domain annotation for manual annotation. This resulted in annotating 2083 triggers as *correct* and 577 triggers as *incorrect*. Unfortunately, 731 triggers were ambiguous and remained *undecided*. These words have multiple meanings and they are used in both biomedical and general domains (e.g. “conserved”, “deletion”, and “development”). Hence, it is possible to construct sentences where these words are valid triggers, but the annotator was not able to find any evidence supporting the use of these words as triggers from the existing literature. The *undecided* triggers are later addressed in our Aggregation method.

Aggregation method

We further aggregate the results from tree pruning and manual annotation. We prioritize the manual annotation, i.e., in the aggregated data a trigger remains *correct* or *incorrect* if labeled as such in the manual annotation. The *undecided* triggers are assigned using the tree pruning method. As a results, the final set is comprised of 2,242 *correct* triggers and 1,149 *incorrect* triggers.

Classification of low frequency trigger words

Previous methods focused on classifying 3,391 top most frequent trigger words (those with at least 300 corresponding events), which account for 97.1% of all EVEX events. Obviously, manual annotation cannot be applied to the huge number of triggers with lower frequency that exist in EVEX. For this aim, we train a support vector machine (SVM) classifier. As training data, we use the aggregated trigger set from the previous section, assigning *correct* and *incorrect* triggers as *positive* and *negative* examples. For features, we use the vector representations of the trigger words (based on the `word2vec` model that we had trained) as well as a carefully engineered set of features for the task. For hyper-parameter optimization, we use grid search com-

bined with 5-fold cross-validation, and after finding the best parameters, we use all training data to train the final model and classify all low frequency EVEX triggers. This results in finding 16,674 negative (presumably incorrect) triggers with a total frequency of 232,748 respective events in EVEX.

We underscore that although low frequency trigger words account for $\sim 3\%$ of the events in the EVEX, these events can appear in end-user queries, even for important genes such as 'P53', so removing incorrect events from this group does make a difference in the perceived quality of the data by the users.

3.1.4 Evaluation and results

To evaluate the effect of trigger filtering methods on event extraction, we focus on the official test set of the BioNLP Shared Task 2011 and BioNLP Shared Task 2013 GE task (Kim et al., 2011b; Nédellec et al., 2013). As the starting point, we consider the outputs of the TEES system entry in 2011 (3rd place) (Björne et al., 2012) and in 2013 (2nd place) GE tasks (Björne and Salakoski, 2013), and for the 2013 Shared Task, also the winning EVEX entry (Hakala et al., 2013b). We prune the outputs of these systems by removing events whose trigger words are identified as *incorrect* using the aforementioned methods and evaluate the resulting pruned set of events using the official evaluation services of the respective Shared Task on the held-out test sets. Results are shown in Table 3.

I underscore that the magnitude of the F-score improvement is modest because the top-ranking systems are well optimized and major improvements have been hard to achieve regardless of the approach. In addition, filtering methods (such as the ones we have developed in this work), cannot increase the recall because they are not able to produce new events. Our main focus in this work thus is on increasing the precision (by removing incorrect events) while trying to retain the recall, which should result in increasing the credibility of large-scale event databases in general.

Evaluation of the **unsupervised method** (event filtering based on the incorrect triggers obtained after pruning the binary cluster tree) shows that in all three instances, when compared to the TEES and EVEX baselines, there is an improvement in both precision and F-score with a relatively small drop in recall. Interestingly, the Pruned-EVEX (Unsupervised Method) achieves a new state-of-the-art result for the task.

Evaluation of the **manual annotation method** shows that this method preserves the recall better than our unsupervised method. However in all three instances, its precision and F-score is less than the precision and F-score of our unsupervised method. The higher precision of the unsupervised pruning strategy shows that some cases not clear for a human annotator, can be classified with this method. As precise annotation was not possible for many trigger words, we have 731 *undecided* top most frequent triggers, and many incorrect trigger words might actually be among them.

| | Predictions | P | R | F1 |
|----------------------|--|-------|-------|-------|
| TEES-2011 (ST-11) | Original TEES | 61.76 | 48.78 | 54.51 |
| | Pruned-TEES (Unsupervised Method) | 62.39 | 48.75 | 54.74 |
| | Pruned-TEES (Manual Annotation Method) | 62.04 | 48.78 | 54.62 |
| | Pruned-TEES (Aggregation Method) | 62.26 | 48.78 | 54.70 |
| | Pruned-TEES (Aggregation Method + SVM) | 62.27 | 48.78 | 54.71 |
| TEES-2013 (ST-13) | Original TEES | 56.32 | 46.17 | 50.74 |
| | Pruned-TEES (Unsupervised Method) | 57.13 | 46.02 | 50.97 |
| | Pruned-TEES (Manual Annotation Method) | 56.63 | 46.17 | 50.87 |
| | Pruned-TEES (Aggregation Method) | 56.97 | 46.17 | 51.00 |
| | Pruned-TEES (Aggregation Method + SVM) | 57.01 | 46.17 | 51.02 |
| EVEX-2013 (ST-13) | Original EVEX | 58.03 | 45.44 | 50.97 |
| | Pruned-EVEX (Unsupervised Method) | 58.77 | 45.29 | 51.15 |
| | Pruned-EVEX (Manual Annotation Method) | 58.32 | 45.44 | 51.08 |
| | Pruned-EVEX (Aggregation Method) | 58.66 | 45.44 | 51.21 |
| | Pruned-EVEX (Aggregation Method + SVM) | 58.71 | 45.44 | 51.23 |

Table 3. Performance comparison of the different pruning approaches and the baseline methods (TEES/EVEX) on the official BioNLP Shared Tasks 2011 and 2013 GE data sets

To summarize, the manual annotation has produced an *almost pure but incomplete* set of incorrect trigger words. In comparison to the baselines, our manual annotation method does increase the precision and F-score while retaining the recall, but its precision and F-score are not as high as our unsupervised method.

Evaluation of the **aggregation method** shows that this method retains the recall and increases the precision and F-score. Interestingly, in all three cases, in comparison with manual annotation method it has a higher precision and F-score. Consequently, we conclude that our unsupervised method is indeed able to find incorrect trigger words elusive to the human annotator.

Finally, we combine the *incorrect* triggers obtained from the aggregation method with the *incorrect* triggers we obtained by running the SVM classifier on low-frequency EVEX triggers and use this set to prune the outputs of the three baselines. Comparing this method (**Aggregation Method + SVM**) against the best previous method (aggregation only) shows slight improvements in precision and F-score while retaining the same recall, suggesting this approach produces the most complete set of *incorrect* trigger words. We emphasize that the improvements are small since most of low-frequency triggers found by the SVM classifier are unlikely to be found in carefully selected in-domain shared task data sets.

We further randomly selected 700 low-frequency EVEX triggers and manually annotated them and built a test set (by retaining the *positive* and *negative* examples and excluding 104 *undecided* triggers for simplicity) and evaluated the performance of the SVM classifier on this test set. The evaluation showed 0.98 positive recall which translates to preserving a significant proportion of correct events, while the classifier has 0.44 negative recall, meaning that it is able to identify about half of the

incorrect events.

Evaluation of event removal on the EVEX resource

For focusing on 3,391 top most frequent EVEX trigger words (accounting for 97.1% of all EVEX events), we rely on our aggregation method because it had the best performance. The aggregation method resulted in labeling 1,149 triggers as incorrect, and these account for 1,105,327 events in EVEX. For the rest of EVEX triggers (low-frequency triggers) we use the SVM classifier for prediction. This resulted in the identification of 16,674 incorrect triggers with 232,748 respective events in EVEX. In total we were able to identify 17,823 presumably incorrect triggers in the whole EVEX resource with 1,338,075 events, which constitute 3.3% of all events in EVEX². Removing such events from EVEX resource improves the quality of data and increases the perceived credibility of this resource for the end-users.

3.1.5 Discussion and conclusions

In this work, we proposed novel methods which can be used for the identification of incorrect trigger words and removing incorrect events from the output of large-scale event extraction systems.

Focusing on top most frequent EVEX triggers (those with event frequency of at least 300 that account for 97.1% of the whole EVEX events), our unsupervised method achieves a modest improvement over the winning system of the BioNLP 2013 Shared Task on GENIA event extraction and establishes a new best score on the task. The aggregation of manual annotation results with our unsupervised method results further increases the precision and F-score of the unsupervised method, while the original event extraction recall is preserved. Focusing on low frequency EVEX triggers (those with event frequency below 300), our SVM classifier on one hand achieves 0.98 positive recall which translates to preserving a significant proportion of correct events, while on the other hand the classifier has 0.44 negative recall, meaning that it is able to identify about half of the incorrect events. Combining the results of our aggregation method with incorrect trigger words identified by applying the classifier on all low frequency EVEX triggers resulted in recognition of 17,823 presumably incorrect triggers with 1,338,075 respective events which constitutes about 3.3% of all events in the EVEX resource.

It should be noted that even though our manual annotation or aggregation methods are able to preserve the recall when evaluated against official predictions of

²**Paper I** also includes discussion about the evaluation of the hierarchical cluster tree before and after pruning incorrect trigger words, and specially with regard to the types of events that they represent. Here, I have omitted those discussions as they are secondary to the main contribution of the paper, i.e., increasing the quality of data in large-scale event databases by removing incorrect events.

Shared Task test sets, it is not guaranteed that the same performance will be achieved when applying them on a large-scale resource such as EVEX. In fact there might be *correct* triggers which are not present in the ST'11 or ST'13 test sets, but are mistakenly labeled as incorrect by the human annotator, our unsupervised method, or the classifier. Consequently, in the evaluation against official Shared Task test sets, we do not delete these triggers so we do not detect any drop in recall. However, based on our evaluation results, we are optimistic that most of the correct events will be preserved when the method is applied on the EVEX resource.

Finally, I underscore that our methods can be adapted and used by other research groups for increasing the perceived credibility of other large-scale event collections similar to EVEX. For example, if a group does not have manual annotation expertise, they can still use our unsupervised method to filter out a proportion of false events and improve event extraction results. If the group members can perform manual annotation but they are not experienced in machine learning, they can use our manual annotation approach, and if they are experienced in both machine learning and manual annotation, they can follow our aggregation method (with or without building a predictive model for classification).

There are several possible future research directions for our work. First, our focus here was on incorrect trigger identification and we studied the impact of event removal on large-scale event resources. Even though our evaluations show only minimal drop in recall, however persistent removal of events might have undesired effects. The alternative would be developing a scoring system that considers both event extraction classification confidence (derived from TEES) and incorrect trigger identification confidence (derived from our approach), to *rank* events when they are shown to the end-user. Therefore, likely incorrect events will be given lower scores, but will remain in the collection for use-cases that demand extremely high recall, but can tolerate or overcome the noise in the data (e.g. through post processing).

Second, it would be interesting to investigate how our method can be extended to address *correct but mistyped* events, thus increasing the precision and recall even further. For instance, it is possible that a detected *regulation* trigger should in fact be classified as *positive-regulation*, a subtype of *regulation*, but the trigger classifier has not been able to make this distinction. By observing where the given trigger word is located in the hierarchical cluster tree, these errors could be possibly corrected.

For clustering and classification of the trigger words we heavily relied on their Word2Vec vector representations. It is worth mentioning that this research work was initiated in 2013 and completed in 2016 (**Paper I**). Biomedical natural language processing methods have extensively improved since then. Therefore, instead of using simple vector representations for clustering/classification of the trigger words, it would be interesting to investigate and possibly utilize modern neural network-based encoders (e.g. BERT (Devlin et al., 2018)) for these tasks. It might be easy in the future to instead of judging the trigger words globally, to focus only on certain

types of contexts in which the trigger words appear, and this may give us the ability to make more precise decisions.

3.2 Deep learning with minimal training data

In this section, I present the contributions of the thesis in response to **research question 2**: Deep learning with minimal training data.

As discussed in Section 1.2, state-of-the-art relation and event extraction methods are based on supervised machine learning, requiring manually annotated data for training. Unfortunately, training data for building biomedical text mining systems is sometimes minimal and this hinders the utilization of the deep learning-based methods to their full potential. The second research question of this thesis is how to efficiently train and optimize deep neural network models for biomedical relation extraction tasks with minimal training and development data. Instead of designing yet another neural network architecture, our aim is to develop simple methods that can be utilized for any biomedical relation extraction task at hand.

In this section, I will briefly review **Paper II** and **Paper III** and discuss a simple method that we developed to deal with the aforementioned problem. **Paper II** describes the methods and results of our participation in the Bacteria Biotope event extraction task (BB3-event) of the BioNLP Shared Task 2016 (Deléger et al., 2016), which resulted in obtaining the second rank among nine participants. In the BB3-event task, named entities (*Bacteria*, *Habitat*, and *Geographical*) are manually annotated and given as input data and the focus of the task is on extracting locations of bacteria by predicting correct labels for (*Bacteria*, *Habitat*) and (*Bacteria*, *Geographical*) pairs.

Paper III discusses an end-to-end system that we developed after the shared task. This system is capable of named entity detection, normalization and relation extraction for extracting information about bacteria and their habitats from biomedical literature. The official evaluation of joint entity detection and relation extraction on the test of the BB3-event+ner task showed that our system outperforms the winning team of the Shared Task by 19 percentage points (pp) on F-score, establishing a new top score for the task. We also achieved state-of-the-art results in the normalization task, as evaluated on the test set of BB3-cat task. Since my contribution was focused on building the relation extraction system in **Paper III**, in this thesis I will focus on describing the details of the relation extraction method rather than entity detection or normalization.

3.2.1 Introduction

The BB3-event task of the BioNLP Shared Task 2016 focused on extracting the locations of bacteria from PubMed abstracts. In this task, three types of named entities

(*Bacteria*, *Habitat*, and *Geographical*) are manually annotated and given as the input data and the task focus is on extracting directed binary associations between (*Bacteria*, *Habitat*) and (*Bacteria*, *Geographical*) candidate named-entity pairs. It should be mentioned that although the associations are directed, in the task, the direction is implied, and determination of the direction is not a part of the task.

For the purposes of machine learning, we thus cast the BB3-event task as binary classification, taking a candidate entity pair as input and predicting whether or not a *Lives_in* relation holds between the *Bacteria* and the location (*Habitat* or *Geographical*). The BB3-event+ner task, in contrast, focused on the joint development and evaluation of named-entity detection *and* relation extraction systems. Hence, named entities in BB3-event+ner test set are not given and they must be predicted with an entity detection system and then provided to a relation extraction system as input. Consequently, the performance of the NER system has a direct impact on the relation extraction system and subsequently on the performance of an end-to-end system.

One interesting aspect of the BB3-event and BB3-event+ner data sets is the minimal number of positive relations. Table 4 shows the statistics of these data sets. As the table shows, there are only 327 annotated relations in the training sets, which make it challenging to train deep neural network architectures for relation extraction. In addition, the development sets are also very small (they include only 223 positive relations). These limitations can lead to overfitting on small training data and poor generalization to unseen data. In the following section, I will first briefly discuss the details of our deep learning-based relation extraction method and then continue by discussing the problems that can arise in training with this minimal data.

| | BB3-event | | | | BB3-event+ner | | | |
|--|-----------|-------|--------|--------|---------------|-------|--------|--------|
| | Train | Dev | Test | Total | Train | Dev | Test | Total |
| Documents | 61 | 34 | 51 | 146 | 71 | 36 | 54 | 161 |
| Words | 13,850 | 8,491 | 13,039 | 35,380 | 16,295 | 8,890 | 13,933 | 39,118 |
| <i>Bacteria</i> | 358 | 238 | 336 | 932 | 375 | 244 | 401 | 1,020 |
| <i>Habitat</i> | 687 | 454 | 720 | 1,861 | 747 | 454 | 621 | 1,822 |
| <i>Geographical</i> | 35 | 38 | 37 | 110 | 36 | 38 | 27 | 101 |
| Total entities | 1,080 | 730 | 1,093 | 2,903 | 1,158 | 736 | 1,049 | 2,943 |
| <i>Lives_in</i> events (<i>Habitat</i>) | 294 | 186 | 312 | 792 | 294 | 186 | 288 | 768 |
| <i>Lives_in</i> events (<i>Geog.</i>) | 33 | 37 | 28 | 98 | 33 | 37 | 26 | 96 |
| Intra-sentence events | 240 | 165 | 248 | 653 | 240 | 165 | 231 | 636 |
| Inter-sentence events | 87 | 58 | 92 | 237 | 87 | 58 | 83 | 228 |
| Total <i>Lives_in</i> events | 327 | 223 | 340 | 890 | 327 | 223 | 314 | 864 |

Table 4. The statistics of BB3-event and BB3-event+ner data sets (Deléger et al., 2016).

3.2.2 Methods

Our relation extraction approach is based on the shortest dependency path (SDP) between the two entities in the sentence dependency parse graph. The syntactic structure connecting two entities $e1$ and $e2$ in various forms of syntactic analysis is known to contain most of the words relevant to characterizing the relationship $R(e1, e2)$, while excluding less relevant and uninformative words (Bunescu and Mooney, 2005). This observation has served as the basis for many successful relation extraction approaches in both general and biomedical domain NLP (Bunescu and Mooney, 2005; Airola et al., 2008; Nguyen et al., 2009; Chowdhury et al., 2011; Liu et al., 2015; Can et al., 2019; Li et al., 2019). However, there is a downside: since dependency parse graphs connect words to others in the same sentence, most of the approaches that are based on the SDP focus on a single sentence at a time³ and consequently, they fail to detect cross-sentence relations, i.e., the relations between the entities that are located in different sentences. Cross-sentence relations usually constitute a small proportion of the total relations but they are known to present particular challenges for event and relation extraction systems, which rarely attempt their extraction (Kim et al., 2011a). Since our approach is based on SDP of single sentences, we also exclude cross-sentence examples from the data⁴.

We propose a multi-channel neural network for relation extraction (see Figure 5). The network architecture is centered around three distinct RNNs (chains of LSTM units) for processing the words, POS tags and dependency types along the SDP which connects the *Bacteria* mention to the location mention (*Habitat* or *Geographical*) in the sentence parse graph. For a given example, the sequences of words, POS tags and dependency types along the SDP are first mapped into vector sequences by three separate embedding lookup layers. These vector sequences are then input into the three RNNs. The outputs of the last LSTM unit of each of the three chains are then concatenated, and the resulting vector is fed into a fully connected hidden layer. The hidden layer finally connects to a single-node binary classification layer. The sigmoid activation function is applied on the output of all LSTM units, the hidden layer and the output layer.

The POS and dependency type embeddings are initialized randomly, in contrast to the word embeddings, which are initialized with pre-trained word embeddings provided by Pyysalo et al. (2013). These word embeddings are induced by training the skip-gram model of the *word2vec* method (Mikolov et al., 2013) on PubMed and PMCOA texts. During the training, the POS and dependency type embeddings are trained and the pre-trained word embeddings fine-tuned.

³Unless the *root* nodes of the parse graphs of individual sentences are connected to form a *document graph*, as proposed by Quirk and Poon (2017) and Peng et al. (2017).

⁴Such relations are calculated as false-negatives of our system when it is evaluated against the official development and test set data.

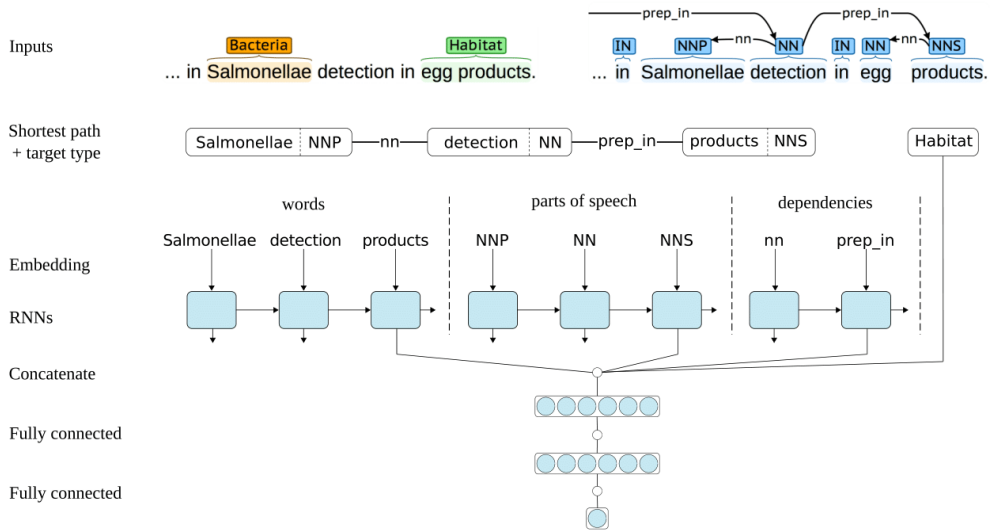


Figure 5. Proposed network architecture for relation extraction.

For training the network we use the Adam optimization algorithm with the parameters suggested by Kingma and Ba (2015). We found that this algorithm yields considerably better results compared to conventional stochastic gradient descent in terms of classification performance.

Determining how long to train a neural network model is critically important for its generalization performance. This is usually determined by the number of epochs, i.e., the number of passes through the training set. If the network is under-trained, model parameters will not have converged to good values. Conversely, over-training leads to overfitting on the training set. A conventional solution to this problem is *early stopping*, which means training the model (typically for one epoch) and evaluating the model performance on the development (validation) set and continuing the training as long as performance on the development set is improving. By repeating this approach for 15 different runs, using the exact hyper-parameters but with different initial random initialization of the model, we experimentally found that the optimal length of training for our neural network architecture on this data is four epochs⁵. Finally, to decrease the risk of the overfitting, we used the *dropout* method (Srivastava et al., 2014) on the output of the hidden layer with a dropout rate of 0.5.

⁵In neural network training, there is a strong interplay between three hyper-parameters: *number of epochs*, *mini-batch size*, and *learning rate*. The best way to find an optimal set of values for these parameters is grid search. If the grid search is expensive, one can keep two of the mentioned parameters fixed and iterate over possible values for the third.

3.2.3 Dealing with minimal training data: measuring and overcoming variance

At the beginning of training, the weights of the neural network are initialized randomly. As we are only using pre-trained embeddings for words, this random initialization also applies to the POS and dependency type embeddings.

Since the number of weights is high and the training set is very small, the initial random state of the model can have a significant impact on the final model and its generalization performance. As discussed earlier, having a limited number of training examples is known to represent significant challenges for leveraging the full power of deep neural networks, and we found this to be the case also in this task.

To study the influence of random effects on our model, we evaluate it with 15 different random initializations, training each model for four epochs on the training data with the *same* set of hyper-parameters, and evaluating on the development set using the standard precision, recall and F-score metrics. Table 5 shows the results.

| Run | Recall | Precision | F-score |
|-----------|--------|-----------|---------|
| 12 | 76.3 | 60.3 | 67.3 |
| 14 | 71.2 | 63.0 | 66.8 |
| 13 | 75.7 | 59.3 | 66.5 |
| 10 | 78.0 | 56.3 | 65.4 |
| 3 | 80.8 | 54.0 | 64.7 |
| 15 | 79.1 | 54.3 | 64.4 |
| 1 | 66.1 | 62.2 | 64.1 |
| 11 | 65.0 | 62.8 | 63.9 |
| 2 | 67.8 | 59.4 | 63.3 |
| 5 | 55.9 | 69.7 | 62.1 |
| 7 | 57.6 | 66.7 | 61.8 |
| 9 | 53.1 | 70.2 | 60.5 |
| 8 | 50.9 | 74.4 | 60.4 |
| 6 | 50.3 | 73.6 | 59.7 |
| 4 | 46.9 | 78.3 | 58.7 |
| \bar{x} | 65.0 | 64.3 | 63.3 |
| σ | 11.3 | 7.3 | 2.6 |

Table 5. Development set results for 15 repetitions with different initial random initializations with mean (\bar{x}) and standard deviation (σ). Results are sorted by F-score.

As Table 5 shows, we find that the primary evaluation metric, the F-score, varies considerably, ranging from 58.7% to 67.3%. The 8.6% difference in F-score clearly illustrates the extent to which the random initialization can impact the performance of the model on unseen data in this particular task. While the method is shown to obtain on average an F-score of 63.3% on the development set, it must be kept in mind that given the standard deviation of 2.6, individual trained models may perform substantially better (or worse). It is also important to note that *due to the small size*

of the development set, individual models that achieved high performance in this experiment *will not necessarily generalize well to unseen data*. To deal with these issues and make a robust relation extraction system, we introduce a straightforward voting procedure that aggregates the prediction outputs of the 15 classifiers based on a given threshold value $t \in \{1, \dots, 15\}$:

1. For each example, predict outputs with the 15 models;
2. If at least t outputs are positive, label the example positive, otherwise label it negative.

With this simple ensemble voting approach, t becomes a hyper-parameter to be optimized against the primary evaluation metric, F-score. Clearly, the most conservative threshold is $t = 15$, where a relation is voted to exist only if *all* the 15 classifiers have predicted it. Conversely, the least conservative threshold is $t = 1$ where a relation is voted to hold if *any* classifier has predicted it. Table 6 shows the development set results for the voting algorithm with different threshold values.

| Threshold (t) | Recall | Precision | F-score |
|-------------------|-------------|-------------|-------------|
| 1 | 83.6 | 53.2 | 65.1 |
| 2 | 79.7 | 54.0 | 64.4 |
| 3 | 78.5 | 57.0 | 66.0 |
| 4 | 78.0 | 59.0 | 67.2 |
| 5 | 75.7 | 60.1 | 67.0 |
| 6 | 70.6 | 60.7 | 65.3 |
| 7 | 67.8 | 61.5 | 64.5 |
| 8 | 65.5 | 62.0 | 63.7 |
| 9 | 62.2 | 65.5 | 63.8 |
| 10 | 58.2 | 66.5 | 62.1 |
| 11 | 57.1 | 69.7 | 62.7 |
| 12 | 52.5 | 70.5 | 60.2 |
| 13 | 51.4 | 72.8 | 60.3 |
| 14 | 48.6 | 74.8 | 58.9 |
| 15 | 45.2 | 80.0 | 57.8 |

Table 6. Development set results for voting based on the predictions of the 15 different classifiers. Best results for each metric shown in bold.

As table shows, the threshold $t = 1$ produces the highest recall (83.6%) with the lowest precision (53.2%). With increasing values of t , precision increases while recall drops, and the highest precision (80.0%) is achieved together the lowest recall (45.2%) with $t = 15$. According to the table, the highest F-score is obtained with $t = 4$, where an example is labeled positive if at least four classifiers have predicted it to be positive, and negative otherwise. Figure 6 shows the precision-recall curve for these 15 threshold values.

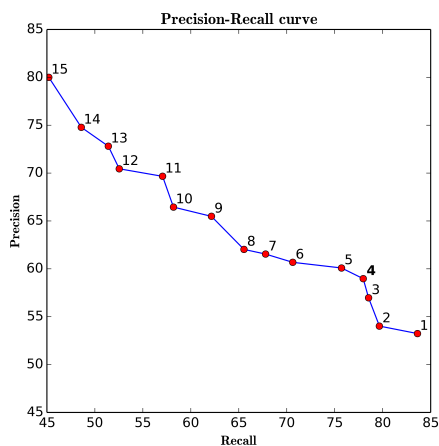


Figure 6. Precision-recall curve for different values of the threshold t (shown as labels on the curve).

As it can be seen in Table 6, the voting algorithm results in obtaining 67.2% F-score on the development set (which is lower than 67.3% F-score, achieved by our best performing neural network model, see Table 5). However, we emphasize again that due to the small size of the training set and the development set, there is no guaranty that our best performing single model can generalize well to unseen data. The ensemble method and voting algorithm in contrast, promise to make a more robust system (as it can be seen based on the results on the held-out test set that we achieved in the shared task).

For prediction of the test set we applied the proposed voting approach: 15 neural networks were trained on the combination of training and development data. Each trained model was then used to produce one set of predictions for the test set. To obtain the final test set predictions, the outputs of the 15 classifiers were aggregated using the voting algorithm with a threshold $t = 4$.

3.2.4 Results

Our method achieved an F-score of 52.1% with a recall of 44.8% and a precision of 62.3%, ranking second among the entries to the BB3-event shared task, being the *best performing deep learning-based system* in this task⁶. The first rank in the shared task was achieved by Lever and Jones (2016) (55.8% F-score, 61.5% recall and 51.0% precision) with a feature-based system that relies on an extensive set of manually engineered features, feature-selection, and an SVM (or logistic regression) classifier.

In the end-to-end system that we developed after the shared task (**Paper III**),

⁶Refer to Deléger et al. (2016) for detailed comparison of the shared task entries.

we followed the same ensemble voting approach for relation extraction, except for neural network models, we replaced the three LSTM networks with three convolutional neural networks followed by global max-pooling. CNNs are significantly much faster to calculate compared to LSTM networks. The evaluation of this approach on the BB3-event test set resulted in achieving an F-score of 51.2% (44.4% recall and 60.5% precision), performing very similarly to the LSTM-based ensemble. Finally, on BB3-event+ner task, the joint evaluation of our CNN-based relation extraction system and our NER system showed an F-score of 38.1%, improving the best previous result (LIMSI developed by Grouin (2016)) by 19 percentage points and achieving the new state-of-the-art score for the task. We emphasize that no other data than BB3 is used for training/optimization of our relation extraction system in any way. In addition, the test sets of BB3-event and BB3-event+ner tasks differ, hence, the results are not comparable.

While training 15 neural models and predicting the development/test sets for 15 times seem to be counter-intuitive for large-scale real-word applications, I underscore that on a single machine equipped with a consumer-grade GPU, training one neural network model is relatively fast (taking about 10 minutes). This is mainly because the training set is very small. Prediction of the development set using a trained model is very fast, taking only about 10 seconds. Finally, the voting algorithm executes in less than a minute for all 15 thresholds⁷. Therefore we conclude that the proposed ensemble voting approach in practice is quite feasible, as the time-consuming training process only needs to be done once and prediction on unseen data is quite fast.

We emphasize that training an ensemble of differently initialized models and aggregating their predictions would not be necessary if sufficient training data was available. For example, on the ChemProt corpus (Krallinger et al., 2017) we observed only $\sim 1\%$ variance in F-score with a similar neural network architecture (**Paper V**) (in contrast to the $\sim 9\%$ variance observed on the BB3-event corpus). This is because the ChemProt corpus has a comparatively much larger training set (4,157 chemical-protein interactions). Since the variance is small on the ChemProt corpus, it can be neglected. To summarize, ensemble training is not critical for corpora that have sufficient training data, but since having limited supervised data is a common case in biomedical text mining, therefore our method can benefit many potential relation extraction tasks.

3.2.5 Discussion and conclusions

Having a limited number of training and validation examples is known to represent significant challenges for leveraging the full power of deep neural networks. When

⁷For more details about runtime performance, see **Paper II**.

the number of weights in a neural network architecture is high and the training set is very small, the initial random state of the model can have a significant impact on the final model and its generalization performance. Since having minimal training data is a common case in biomedical text mining, I would like to bring attention to the often underestimated effect of network initialization when working with small data sets and state that it might produce substantial variance in F-score (or similar performance metrics), even in cases that a few hundred training examples are available. For example, on BB3-event corpus (with 327 training examples) we showed that F-score can vary up to 8.6 percentage points. Therefore, I encourage BioNLP researchers to measure and take into account the variance of the performance of their neural network models, specially when working with small data sets. When supervised data is minimal, a single experiment (training a neural model on the training set and obtaining a high or low score on the development set) does not necessarily reflect the real performance. For example, achieving a high score on the development set can easily happen just by chance and due to the initial random weights, but this does not necessarily translate to good generalization performance in large-scale real-world applications. Therefore, it is crucial to measure the variance and address the issue if it is necessary. It is worth noting that even with a larger number of training examples, a variation in the evaluation metrics will happen, but the variance in such cases is usually small, thus it can be neglected.

In this work we proposed a simple ensemble training and voting method that can overcome the variance, stabilizes the output and results in robust neural network-based relation extraction systems. Our approach consists of training the neural network model for a number of times with different initial random states, predicting the development/test set examples and aggregating the classifiers' predictions using a simple voting algorithm and an aggregation threshold hyper-parameter (t) that can be used to select different precision-recall trade-offs. If the aim is to obtain the best overall performance, we can investigate which threshold parameter produces the highest F-score. Alternatively, for applications that specifically require high recall or high precision, a different threshold value can be selected to optimize the desired metric. Our approach resulted in achieving the second rank in the BioNLP 2016 BB3-event shared task, being the best deep learning-based method among the participating entries. We emphasize that the proposed ensemble training and voting approach can be easily extended to multi-class and multi-label classification tasks. For example, for multi-class classification, we can add the prediction confidence scores of individual neural models (i.e., the outputs of the softmax activation functions of the individual decision layers) and then judge the label based on the element which has the maximum confidence.

Finally, if the development set is really small, we suggest to use the proposed ensemble voting approach in a cross-validation setup. There are several possible ways to couple the two ideas but one simple way is to first combine the training and de-

velopment sets into a bigger set, and then partition this set into a *model-building* set (e.g. 70% of the total data) and an *ensemble-aggregation* set (e.g. 30% of the total data). A K-fold cross-validation can then be applied on the model-building set. For example, this set can be partitioned into 5 folds. In each iteration, 4 folds (80% of the data) are used to train a neural network model, and the remaining fold (20% of the data) is used for hyper-parameter optimization. By selecting the best performing model from each iteration, we will obtain 5 neural network models that are trained and optimized on different parts of the data. To make an ensemble and find the best value for the voting hyper parameter (t), we can predict the ensemble-aggregation set with these trained models and select a value that leads to the highest F-score on this set. Alternatively, Peng et al. (2018) proposed to train a meta-classifier (e.g. an SVM classifier) which can learn to predict the label for each example, based on the confidence scores, produced by the individual neural network models for that example. By combining the ensemble training idea with cross-validation, we can reduce the risk of overfitting and move one further step toward obtaining better generalization on unseen data.

3.3 Exploring different contexts for relation extraction

In this section, I present the contributions of the thesis in response to **research question 3**: Exploring different contexts for relation extraction.

As discussed in Section 1.2, relation extraction systems can be broadly divided into three main categories, based on the type of context that they use: (1) systems that use all tokens of the sentence as the context for relation extraction, (2) systems that only rely on the shortest path that connects the two candidate named entities in the sentence dependency parse graph, and (3) systems that rely on the SDP in conjunction with the full sentence tokens.

Unfortunately, the relation extraction systems belonging to either one of the three groups mentioned above, are usually evaluated on different corpora or rely on different sets of features and utilize different machine learning techniques for optimization. This makes it difficult to judge which of the mentioned approaches is superior for relation extraction. Therefore, a systematic comparison of the three approaches would be interesting. In particular, this thesis aims to systematically explore the effect of using full sentence tokens *besides* the SDP tokens for improving the performance of relation extraction in modern deep learning-based methods.

For this aim, in this section I will briefly compare the two neural network systems (discussed in **Paper V**) that we developed for chemical-protein relation extraction. Both systems are trained, optimized and evaluated on the ChemProt corpus (Krallinger et al., 2017) and they rely on similar sets of features. The first system (ST-ANN) relies on the SDP as the context for extracting relations, whereas the second system uses SDP *and* full sentence tokens as the context.

3.3.1 Data

The ChemProt corpus is a pairwise relation data set that was created for the Text mining chemical-protein interactions track of the BioCreative VI challenge (Krallinger et al., 2017). In this task, all entities are given as known data to the participants, thus the task is to predict the relations for valid pairs of these entities. The relations are directed, *always* connecting a GENE-type entity (gene or protein) to a CHEMICAL-type entity⁸. A large set of distinct types are used for annotating the relations, but these types are combined into 10 groups that are used as the actual classes for this task. Further, only five of these classes are taken into account in the task evaluation. The micro-averaged F-score of the five target classes is the official metric used for evaluation.

Cross-sentence relations constitute less than 1% of the total relations in the ChemProt training set. In addition, only 10 pairs in the training set have been labeled with multiple relation types. Hence, we formulate the task as a multi-class classification task where we classify each valid pair of entities as one of the 10 annotated relation types or as a “negative”, and we only focus on candidate pairs belonging to the same sentence.

3.3.2 Method

The first system (ST-ANN) is an ensemble of four neural network models with *identical* architectures, that are initialized with different random weights to stabilize the output and overcome the variance in the measured performance⁹. The neural networks in this ensemble are based on the SDP that connects the two candidate entities in the sentence. Each neural model utilizes three separate LSTM chains for representing the words, POS tags, and dependencies (i.e. directed and typed edges in the parse graph) along the SDP that connects the chemical entity to the protein (also referred to as gene) entity (see Figure 7).

As Figure 7 shows, the shortest dependency path in the parse graph is first found. The path is traversed from the chemical entity to the protein entity, producing the sequence of words, the sequence of POS tags, and the sequence of dependency types along the path. The words, POS tags and dependency types are then mapped into their corresponding vector representations using embedding lookup layers and then input to three separate LSTM chains. The outputs of the last LSTM units of the three chains are concatenated together and the resulting higher dimensional vector (i.e. *the SDP vector representation*) is input to a hidden dense layer. The hidden layer finally

⁸Although the associations are directed, in the task, the direction is implied, and determination of the direction is not a part of the task.

⁹As discussed earlier, we noticed ~ 1 percentage point variation the F-scores, which suggests that making an ensemble of neural network for this corpus is not critical, but we chose to train an ensemble anyway, to make a robust system.

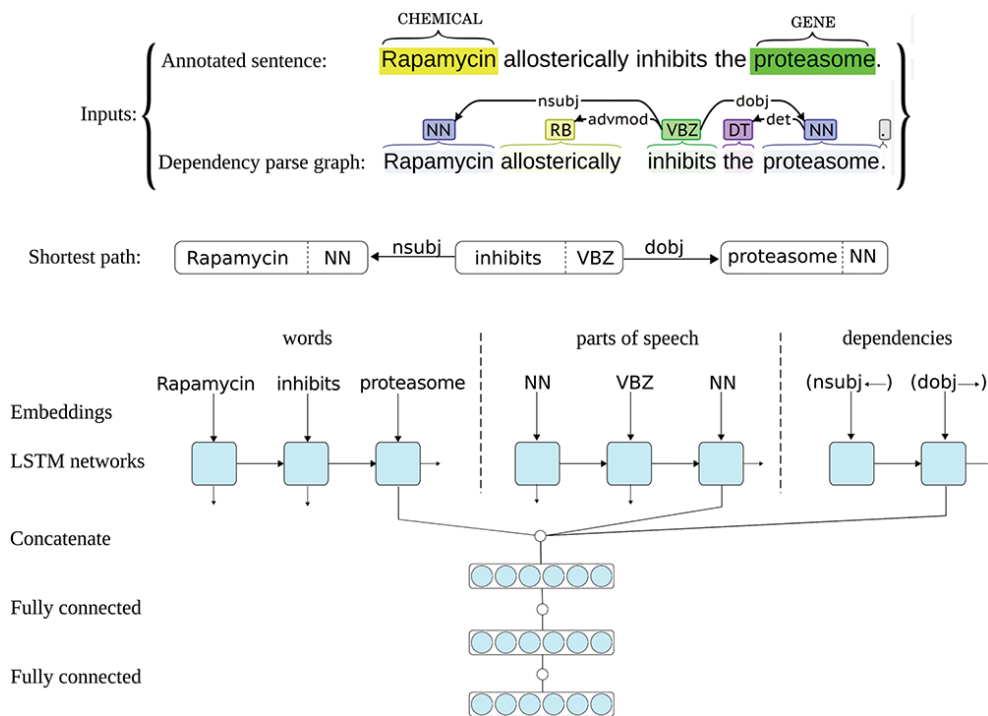


Figure 7. Architecture of one neural network in the ST-ANN ensemble.

connects to the decision (classification) layer, which has a softmax activation.

The second system (I-ANN) is also an ensemble of four neural network models with identical architectures, that are initialized with different random weights to stabilize the output and overcome the variance in the measured performance. The neural models in I-ANN ensemble are similar to the ST-ANN models (i.e. each model utilizes three LSTM chains for representing the sequence of words, POS tags and dependencies along the SDP), but a bidirectional LSTM (forward and backward chains) is also added to the architecture for learning a representation of the full sentence and the two entities of interest in it (see Figure 8).

As Figure 8 shows, the input to the model is a sentence and its dependency parse graph. The model utilizes three LSTM chains for learning a *SDP vector representation* and two LSTM chains (forward and backward) for learning a *full sentence vector representation*. The words, POS tags and dependency types along the SDP connecting the chemical entity to the protein entity are mapped into their corresponding vector representations (embeddings) and input to the three SDP LSTM chains. In addition, for each token of the sentence, its word, POS tag, position to the first entity, position to the second entity, and token-type are mapped into their embeddings and concatenated. Forward and backward sequences of the resulting token representations are input to the forward and backward sentence LSTM chains, resulting in

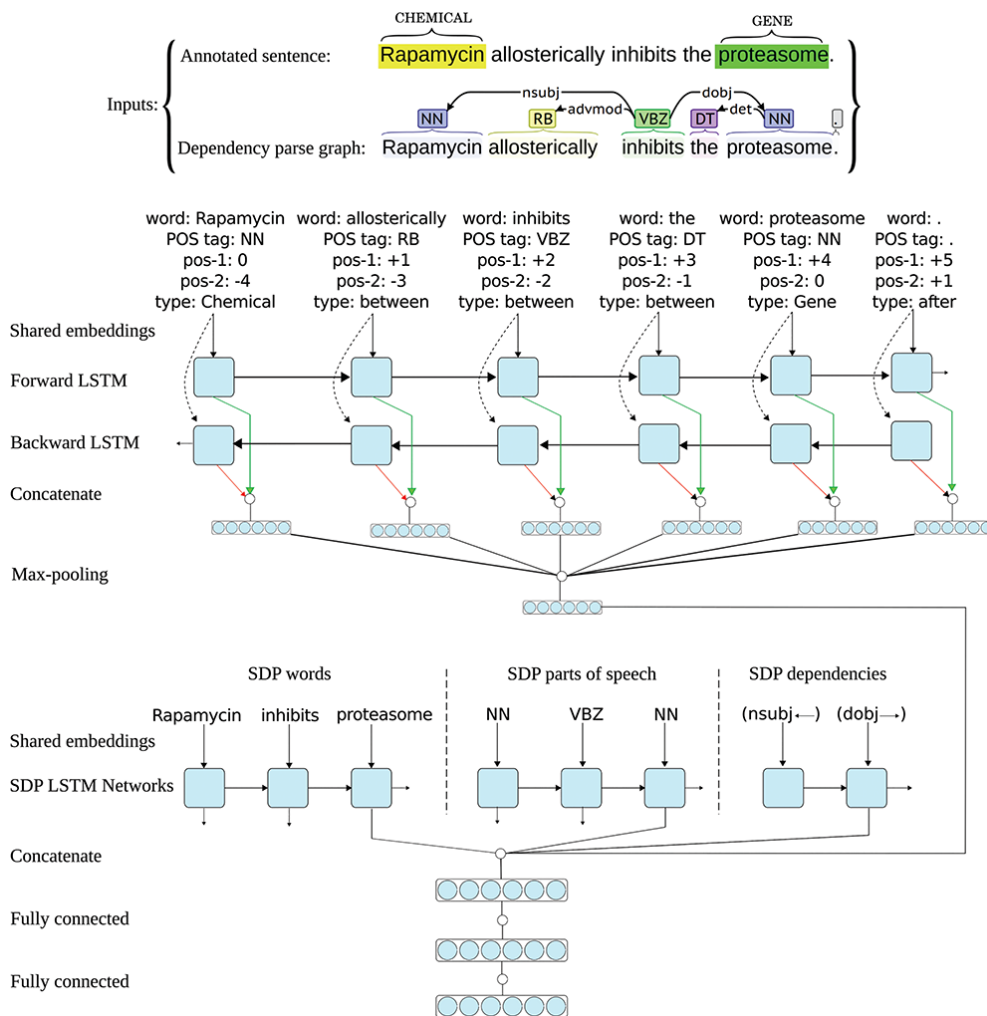


Figure 8. Architecture of one neural network in the I-ANN ensemble.

two hidden representation for each token (forward and backward), which are further concatenated to obtain the final representations of the sentence tokens. Applying max-over-time pooling on these representations produces a vector representation for the full sentence. The outputs of the last LSTM units of the three SDP chains and the full sentence vector representation are concatenated together and the resulting higher dimensional vector is input to a hidden dense layer. The hidden layer finally connects to the decision (classification) layer, which has a softmax activation. The word and POS tag embeddings are shared among the SDP and the full sentence LSTM chains.

3.3.3 Results

We train ST-ANN and I-ANN neural networks on the ChemProt training set and optimize the hyper-parameters by performing a grid search and predicting the development set (for a comprehensive list of hyper-parameters, see **Paper V**). We use the official evaluation script provided by the organizers¹⁰. After training, we notice the F-score of proposed models can vary up to 1 percentage point when evaluated on the development set. Therefore, instead of training a single model for ST-ANN or I-ANN systems, we train an ensemble of four neural network models (with identical architectures but different initial random weights) for each system. Each neural network in ST-ANN or I-ANN ensemble predicts a set of confidences for each development/test set example. The final prediction for an example is generated by summing the confidences of all networks in an ensemble and selecting the label with the highest overall confidence. Table 7 shows the evaluation results.

| System | Development set | | | Test set | | |
|--------|-----------------|--------|---------|-----------|--------|---------|
| | Precision | Recall | F-score | Precision | Recall | F-score |
| ST-ANN | 60.51 | 58.01 | 59.23 | 61.55 | 53.93 | 57.49 |
| I-ANN | 63.18 | 56.25 | 59.51 | 62.39 | 57.81 | 60.01 |

Table 7. Evaluation results on ChemProt development and test sets (result are from row 3 and row 4 in Table 4 from **Paper V**)

As Table 7 shows, on both development set and test set, I-ANN system outperforms ST-ANN system, showing that utilizing full sentence tokens besides the SDP features can actually improve the performance of relation extraction in this task.

3.3.4 Discussion and conclusions

We showed that utilizing full sentence tokens besides SDP features can actually improve the performance of relation extraction. This suggests there are clues outside of the SDP which can help in characterizing the semantic relationships between the candidate entity pairs, therefore, one should not solely rely on SDP features for building state-of-the-art relation extraction systems.

We emphasize that our work is by no means an ablation study. For instance, we do not have the results for an LSTM-based ensemble system that only relies on full sentence tokens (and not the SDP). We first developed the ST-ANN system for our

¹⁰As discussed earlier, the relations in the training and development sets are annotated with 10 different possible class labels but only 5 of these class labels are taken into account in the task evaluation. In our neural networks, the decision layer is an 11-dimensional vector, predicting either one of the 10 positive class labels or the "negative" class label when there is no relation between the chemical and protein entities. The official evaluation script automatically discards any predicted relation if its type does not belong to one of the five target class labels.

participation in the BioCreative VI ChemProt track (Mehryary et al., 2017a). After the shared task and inspired by the CNN-based system developed by Peng et al. (2018), we developed I-ANN system to improve our previous results.

In addition, we emphasize that the performance of a relation extraction system does not boil down to the type of context that it uses, but also on the machine learning and optimization methods. For example, the highest F-score (64.10) in the BioCreative VI ChemProt track was achieved by Peng et al. (2018) with a system combination approach. Their method is an ensemble of three separate systems: (1) a CNN-based relation extraction system that receives the sentence sequence and the SDP sequence as inputs, (2) an RNN-based system that utilizes a bidirectional LSTM network to learn from the full sentence sequence, and (3) an SVM-based system that generates features based on the full sentence and the SDP sequences. After the shared task, Lee et al. (2019) achieved an F-score of 76.46 by pre-training a BERT encoder (Devlin et al., 2018) on PubMed abstract and fine-tuning it on ChemProt training set examples using a decision layer. Finally, we achieved the state-of-the-art result on ChemProt corpus with an F-score of 77.19 using a pre-trained BERT encoder and pre-trained entity and pair embeddings (**Paper VI**). In this work, I mainly focused on neural network-based relation extraction system. For additional discussion regarding the effects of context in non-neural systems, refer to Miwa et al. (2010).

3.4 Hybrid relation extraction systems

In this section, I present the contributions of the thesis in response to **research question 4**: Exploring hybrid relation extraction methods.

As discussed in Section 1.2, according to Zhang et al. (2017), supervised relation extraction methods can be broadly divided into three main groups: (1) feature-based methods, (2) kernel-based methods and (3) deep learning-based methods, each requiring a distinct set of skills and expertise such as feature-engineering, kernel design and neural network architecture design and pre-training.

While there have been great advances in each group, a separate line of research has focused on building hybrid relation extraction methods. In particular, many successful relation extraction methods have been developed by combining feature-based and deep learning-based methods. This combination can be achieved in two levels: system-level combination (i.e., combining the predictions of various systems) and feature-level combination (e.g., incorporating engineered features into neural network models). In this section, I will briefly review the existing methods and discuss the new methods for building hybrid relation extraction systems that we have developed.

3.4.1 System-level combination

The easiest way to combine feature-based and deep learning-based systems is via system combination. This approach is not new and numerous successful text mining systems are developed using this approach. In this setup, two or more relation extraction systems are *separately* trained and optimized. For each machine-learning example (i.e., a candidate entity pair), each system provides a set of confidence scores (based on the number of classes in the task). The predictions of individual systems are systematically aggregated together, providing a final set of predictions for the pairs. This combination approach usually makes sense when the individual systems rely on distinct sets of features or they are based on different machine learning methods or they are trained and optimized differently. In this section, I will first discuss a simple system combination method that we adopted to improve the performance of chemical-protein relation extraction on the ChemProt corpus (**Paper V**), and then I will briefly discuss a better system combination method which is based on training a meta-classifier.

System combination for improving chemical-relation extraction performance

For participation in the BioCreative VI ChemProt track, we developed the ST-ANN ensemble system (discussed in Section 3.3.2) and combined its predictions with the TEES system (Björne, 2014)¹¹. TEES is a feature-based relation extraction system that relies on support vector machines for classification.

After the shared task, for improving our previous results, we developed the I-ANN ensemble system (see Section 3.3.2) and combined its predictions with the TESS system (**Paper V**). As discussed earlier, the I-ANN system outperforms the ST-ANN system. Therefore, I only focus on the results of combining the predictions of TEES with the I-ANN system.

The SVM system (i.e. TEES) and the I-ANN system differ substantially. This is a potential case for testing whether combining predictions of the two systems could help in achieving better performance for chemical-protein relation extraction task. We implement this system combination by merging the relation predictions from the two systems as either a *union* (OR) or an *intersection* (AND) operation, resolving overlapping predictions with conflicting types using the classifier confidence scores. Since all entities are known data in this task, the predictions from the two systems can be aligned using pairs of gold standard entities.

If only one system predicts a relation for a given pair of entities, it is either included in (OR) or discarded from (AND) the combination. If both systems predict

¹¹The results for combining predictions of the ST-ANN system with TEES system can be found in Mehryary et al. (2017a).

a relation, the relation with the higher confidence score is included in the combination. Both SVM and I-ANN systems produce confidence scores in their own ranges. These ranges are normalized into the 0 – 1 interval for both systems, after which the normalized scores are compared. We experiment with combining all predictions (all 11 possible classes, including the “negative” class), only positive predictions (all 10 possible classes) or only predictions for the classes that are evaluated in the shared task (the five target classes) and find that system combination in fact leads to better performance scores on the task¹². Figure 9 illustrates how the predictions of the I-ANN system and the SVM system are combined to produce a final set of predictions for the development and the test set.

¹²The official evaluation script always discards any relation if its type does not belong to the 5 evaluated classes, and then calculates the micro-averaged F-score for the task.

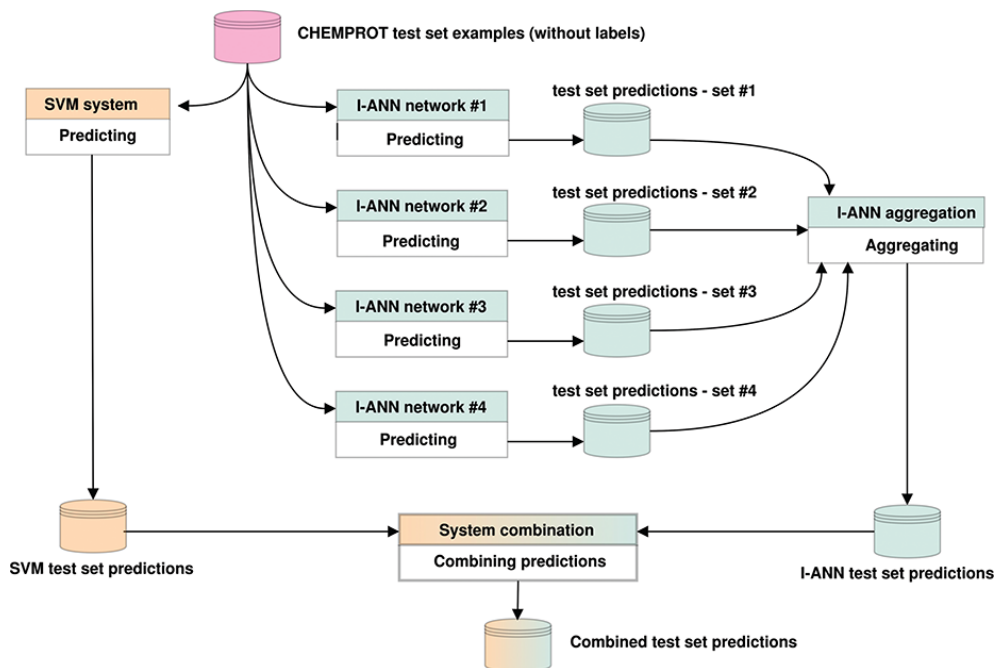


Figure 9. Predicting labels for ChemProt test set examples. The figure illustrates how the predictions of the I-ANN system and the SVM system are combined to produce a final set of predictions for the test set. Each neural network in the I-ANN ensemble predicts a set of confidences for each test set example. The confidences for each example are summed together and the label with the highest overall confidence score is selected as the relation type for that example. This aggregation procedure produces the final set of predictions by the I-ANN ensemble. The SVM system also predicts a set of confidences for each example, and the label with the highest confidence score is selected as the predicted relation type for that example. The confidence scores of the I-ANN and the SVM system predictions are further normalized into 0-1 interval. Using one of the aforementioned system combination methods (e.g. intersection or union), the two prediction sets are combined together, producing a final, combined set of predictions for the test set. The same procedure is applied for predicting labels for the development set and test set examples.

| System | Development set | | | Test set | | |
|-------------------------------------|-----------------|--------------|--------------|--------------|--------------|--------------|
| | P | R | F | P | R | F |
| SVM | 64.55 | 54.72 | 59.23 | 66.08 | 56.62 | 60.99 |
| I-ANN | 63.18 | 56.25 | 59.51 | 62.39 | 57.81 | 60.01 |
| SVM + I-ANN (OR, positive classes) | 58.70 | 63.78 | 61.14 | 61.65 | 66.66 | 64.05 |
| SVM + I-ANN (AND, positive classes) | 74.73 | 48.47 | 58.80 | 74.45 | 50.23 | 59.99 |
| SVM + I-ANN (OR, all classes) | 65.56 | 56.17 | 60.50 | 65.66 | 58.21 | 61.71 |
| SVM + I-ANN (AND, all classes) | 65.56 | 56.17 | 60.50 | 65.68 | 58.16 | 61.69 |
| SVM + I-ANN (OR, eval classes) | 57.65 | 65.81 | 61.46 | 59.05 | 67.76 | 63.10 |
| SVM + I-ANN (AND, eval classes) | 79.36 | 46.94 | 58.99 | 77.79 | 48.21 | 59.53 |

Table 8. Performance of the systems on the development set and the test set

Table 8 shows the evaluation results of different approaches on the development and the test set, using the official evaluation script provided by the task organizers. According to the results, the SVM and I-ANN systems perform similarly on the development data, achieving 59.23 and 59.51 F-score respectively. However, by combining the predictions of SVM and I-ANN systems, further performance improvements have been achieved. The best F-score on the development set (61.46, SVM + I-ANN (OR, eval classes)) is achieved by first removing the negative and non-evaluated predictions and then taking the union of the predictions, and resolving overlapping predictions with conflicting types by using the normalized classifier confidence scores. This approach results in an F-score of 63.10 on the test set, and hence, this is our best F-score for the task, 2.11 pp higher than our best test set submission during the shared task. This is also very close to the highest F-score (64.10), achieved by Peng et al. (2018) in the shared task.

Training meta-classifiers for improved system combination

Even though our easy-to-implement system combination method was shown to improve the overall performance, I believe it is not the best way of combining the predictions of individual systems.

Ideally, one should use a system combination approach in a cross-validation setup. In addition, one can try to train a classifier to *automatically learn* the best way of combining the predictions of the individual systems based on the confidence scores. Such classifiers are typically referred to as meta-classifiers since they use other classifiers' predictions as their features.

While there are several ways to implement the aforementioned ideas, here I will briefly discuss the approach used by Peng et al. (2018), the winning team in the BioCreative VI ChemProt shared task. They participated with an ensemble system composed of an SVM system, an RNN-based system and a CNN-based system and for system combination they either used majority voting or they combined the predictions of the three systems by training a random forest classifier.

They first combined the training and development set and then partitioned the total data into 5 folds. During the development and for hyper-parameter optimization, they used 3 folds for training a system, one fold for training the random forest classifier and one fold for validation. However, for their final submission (entries to the shared task), they used 4 folds to build every SVM, CNN and RNN model and one fold to build the ensemble system. Since they used 5-fold cross-validation, they obtained five SVMs, five CNNs and five RNNs in total. During the ChemProt task, they submitted five runs as they final submissions. Each of their final submissions is *based on the best method for each of the folds*. For example, Submissions 1 and 2 use a majority voting system. These runs were chosen based on their respective folds performance on the validation dataset. Thus, voting was the best method on folds 1

and 2 and random forest combination was best for Folds 3 – 5. Their Submission 2 (which is based on majority voting) achieved the first rank in the shared task with an F-score of 64.10.

3.4.2 Feature-level combination

In this section, I will first briefly review the methods that have been proposed to incorporate engineered features into neural network models, then I will briefly discuss **Paper IV** and introduce a novel approach that we developed during our participation in the 2nd Social Media Mining for Health Applications Shared Task - Task 2 (Sarker et al., 2018).

The most popular and common way for feature level combination is via automatic learning of feature vector representation. For example, POS tags for each token can be first extracted from the input text with a POS-tagger. Each possible POS-tag can be represented with a dedicated n-dimensional vector representation (i.e., embedding) within the neural network model. Using an embedding lookup layer, the sequence of POS-tags in the input sentence (or SDP) can then be mapped into the corresponding sequence of POS-tag vector representations and then fed to the subsequent layer(s) in the network such as a CNN or an RNN. Alternatively, the POS-tag embedding for each token can first be concatenated to its word embedding (and other possible embeddings), making a higher-dimensional vector representation of the token, and then the sequence of tokens' representations be fed into the subsequent layers in the network. When referring to “engineered” features, it makes sense to make a distinction between POS-tag embeddings (and other engineered embeddings) and ordinary word embeddings, because words naturally exist in the raw input sequence, hence they do not require running a dedicated feature extractor such as a POS-tagger to be extracted from the input.

The embeddings in a neural network model are usually initialized randomly, but upon the training with the training data, they will be automatically learned, such that similar elements (e.g., words, POS tags or dependency types) will have similar vector representations in the vector space model. Alternatively, the embeddings can be pre-trained in advance with unsupervised algorithms and then incorporated into neural network models. For instance, word embeddings can be pre-trained with methods such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). After incorporation, the embeddings will be further *fine-tuned* based on the supervised signal and the actual training data for a particular task at hand (e.g., relation extraction).

Various types of embeddings have been proposed by researchers. The most popular embeddings include POS-tag embeddings, dependency-type embeddings, named-entity embeddings (if the token is part of a specific entity type or not) (Peng and Lu, 2017), token-type embeddings (if the token is part of a specific entity type or it is located *before* the first entity, *between* the two entities, or *after* the second entity)

(Mehryary et al., 2018), position embeddings (e.g., the relative distance of the token to the first or second occurring entity of interest in pairwise relation extraction) (Zeng et al., 2014; Nguyen and Grishman, 2015), on-dep-path embeddings (shows whether the token is on the dependency path between the two entities or not) (Fu et al., 2017), and path embeddings and shortest-path embeddings (the incoming or outgoing dependency type for each token if it is located on the SDP) (Björne and Salakoski, 2018), each requiring a different amount of work to be implemented and extracted from input texts.

One important aspect of the mentioned embedding approach is assignment of a feature vector to each token in the input sequence. This is suitable for many types of features since there is a natural order in the tokens in an input sequence (e.g., sentence or SDP). Once the sequence of embeddings is obtained, the subsequent layers in the network architecture can take over and utilize this input sequence in various ways. For example, CNNs can automatically learn position-invariant semantic n-grams and RNNs can effectively learn a good representation of the input sequence. However, during the past decades, the NLP community has developed numerous binary, categorical and numerical features (useful for different text mining applications), that can not necessarily be attributed to single tokens. For example, various types of patterns can be extracted from the input and be represented as binary features.

If the number of such features is limited, they can directly be given as input to the network and then concatenated to the sentence or SDP vector representations (e.g., concatenated to the output of an RNN which processes the sentence/SDP sequence). However, feature-based text mining methods are famous for using substantially massive but sparse matrices which can include hundreds of thousands of columns for representing the features. If the number of features is substantially high, efficient training of neural models can become challenging. The neural network models may become excessively large with millions of weights and we may not have enough training data to efficiently learn all the weights from the sparse features. Therefore, we propose to use dimensionality reduction methods in such scenarios to first map the excessively large feature vector representations into lower-dimensional condensed vectors and then use these vectors as additional inputs to the network to further improve the performance of the neural network models. We successfully adopted this approach (**Paper IV**) during our participation in the 2nd Social Media Mining for Health Applications Shared Task - Task 2 (Sarker et al., 2018). Adding condensed feature vectors to the neural network model resulted in 2 percentage points improvement on the development set and the test set. Here, I will briefly discuss the task and our method.

Ensemble of convolutional neural networks for medicine intake recognition in Twitter

The goal of the task was to develop systems capable of recognizing mentions of medication intake in Twitter. Each of the provided tweets were to be assigned with one of the following three classes: *personal medication intake*, *possible medication intake* or *non-intake*. This is an important preliminary task for extracting adverse drug reactions (ADRs) from social media, since it can filter out the majority of tweets that mention drugs without any indications of personal intake.

We participated in this task with two systems: (1) a feature-based system which uses a linear SVM for classification (our baseline system), and (2) an ensemble of neural networks with features generated by word- and character-level convolutional neural network channels and a condensed weighted bag-of-words representation.

For our baseline approach, we form term frequency-inverse document frequency (TF-IDF) weighted sparse bag-of-words (BOW) representations for all given tweets. These representations are not only constructed for single tokens, but also token bi-grams, trigrams and character n-grams of length 1 to 4. These representations are then fed as features to a linear SVM classifier. The regularization parameter is selected to optimize the micro-averaged F-score of *intake* and *possible intake* classes, the official evaluation metric on the development set. For the final submission in the shared task we merge the training and development sets and train the system on the combined data set.

Our CNN-based system is composed of an ensemble of neural network models with identical architectures that are initialized with different random weights (to stabilize the output and overcome the variance in the measured performance). Each neural model in the ensemble receives two inputs: the sequence of words, and sequence of characters in the input tweet. The separate inputs are processed with separate convolutional channels. Each element in these sequences is represented with a latent feature vector, i.e. an embedding. The word embeddings are initialized using word2vec model, trained with approximately 1 billion drug related tweets as provided by Sarker and Gonzalez (2017). The character embeddings are initialized randomly, but the networks were allowed to backpropagate to both word and character embeddings. The convolutional kernels are applied on the aforementioned two sequences using sliding windows. The outputs are subsequently max-pooled and concatenated. The concatenated vectors are further fed through two densely connected layers, the latter having the output dimensionality corresponding to the number of labels in the data set with softmax activation. The networks are trained on the official training data using the Nadam optimization algorithm. The networks are regularized with dropout rate of 0.2 after the first dense layer, no explicit regularization is applied on the convolutional part of the network. The training is stopped once the performance on the development set is no longer improving, measured with

the official evaluation metric. We perform hyper-parameter optimization using the grid-search and predicting the development set (for a comprehensive list of hyper-parameters, see **Paper IV**).

Training the network on this data set resulted in relatively large variance in the measured performance, caused by the random initialization of the weights. Thus we stabilize the system by training 15 networks, all identical apart from the initial (random) weights. We then select the optimal subset of these networks, as measured on the development set, for the final system where the final predictions are created by summing the confidences of all selected networks and choosing the label with the highest overall confidence. The final system included a subset of 6 neural networks out of the 15. We state that this approach may potentially overfit on the training set.

Evaluations on the development set showed that unfortunately, the CNN-based ensemble was on par with the SVM-based system, i.e., we did not notice any significant differences in the task metric between the two systems.

To further improve the performance of the CNN-based system, we thus used an additional input to the neural network models, in conjunction with the sequences of words and characters: in addition to the convolutional layers, we utilized the same TF-IDF weighted sparse vector representations as in the baseline method. As these representations have dimensionality in the order of hundreds of thousands, we first densify the representations to 4000 dimensional vectors using truncated singular-value decomposition (SVD) method (Halko et al., 2011). These vectors are concatenated alongside the CNN outputs. This dimensionality reduction is performed mainly due to computational reasons since the approach was prototyped on a consumer grade GPU with limited amount of memory. Projecting the sparse vectors to 4K dimensions preserves 74% of the variance in the data. Adding these additional features resulted in ~ 2 percentage points increase, both on the development and the test set. Table 9 shows the results of the two systems and the results of the winning team in the shared task (InfyNLP).

| | Development set | | | Test set | | |
|---------|-----------------|--------|---------|-----------|--------|---------|
| | Precision | Recall | F-score | Precision | Recall | F-score |
| SVM | 72.3 | 67.0 | 69.6 | 69.2 | 60.1 | 64.3 |
| CNN | 74.2 | 71.2 | 72.7 | 70.1 | 63.0 | 66.3 |
| InfyNLP | - | - | - | 72.5 | 66.4 | 69.3 |

Table 9. Overall performance of our SVM and CNN-based systems. The development set results are measured with our own evaluation whereas the test set scores are as reported by the organizers. For comparison we have added the results of the best performing team: InfyNLP.

By incorporating engineered features into the CNN-based system, we achieved a relatively strong performance, with an F-score of 66.3 according to the official evaluation, resulting in the 5th place in the shared task with performance close to the winning systems.

I emphasize that due to Twitter policy, the organizers only provided the IDs of the tweets instead of their actual content. Since we started investigating this task much later than the data was released, we were only able to obtain the contents for 7444 tweets out of the 8000 annotated tweets as some of the content had been already removed by the Twitter users, i.e. we had 7% less training data than teams who were involved in the shared task since the very beginning. The organizers also provided a separate development data set, which consists of 2260 annotated tweets, of which we also lost roughly the same proportion. As we only had access to a partial training data, we try to estimate how much the performance of the systems could have been improved with additional data. To accomplish this, we train the CNN-based system with different subsets of the training data, starting from 5K training examples and incrementally increasing the size in steps of 300 up to the whole training data available to us. After every increment we evaluate the system's performance on the development set. To stabilize the output caused by different initial random weights, we train 5 networks with each subset of the training data, and calculate the average performance for each subset. Fitting a linear regression on the resulting measurements shows that in this region, the learning curve is fairly linear and decent performance improvements can be gained by adding more training data. Assuming a performance increase equal to the slope of the fitted regression line, having the full training data set would have increased our performance by 0.7pp in F-score, ranking our system higher in the shared task.

3.5 Entity-pair embeddings for improving relation extraction in the biomedical domain

In this section, I present the contributions of the thesis in response to **research question 5**: Unsupervised pre-training of biomedical entity-pair embeddings to improve relation extraction performance.

I will briefly review **Paper VI** and I discuss a novel method that we have developed to pre-train embeddings (vector representations) of biomedical named entities and entity pairs. Our main goal is to incorporate these embeddings into neural network architectures to improve the performance of relation extraction for the biomedical domain.

3.5.1 Introduction

Biomedical literature includes a significant number of sentences that discuss the relations and interactions of biomedical named entities (e.g. genes, proteins, chemicals, and drugs). We aim to leverage this literature-wide information using unsupervised methods and for every unique named-entity pair (E_i, E_j) , capture all stated information about E_i, E_j and their relations and build embeddings (vector representations)

of entities and entity pairs.

In this work we mainly focus on chemical-protein entity pairs and we evaluate our approaches on the ChemProt relation extraction corpus (Krallinger et al., 2017), but our methods can be used for other relation extraction tasks such as protein-protein relation extraction. Similarly to the `word2vec` method for ordinary words (Mikolov et al., 2013), our objective is for similar proteins, chemicals and chemical-protein pairs to obtain similar embeddings and feed these embeddings into neural network models to improve the performance of relation extraction (see Figure 10).

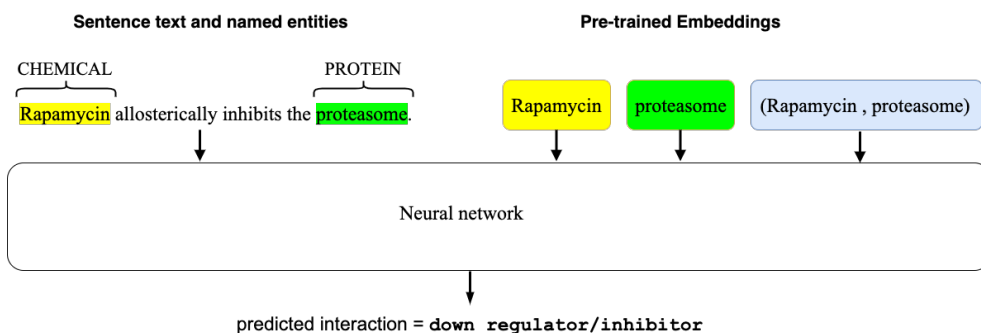


Figure 10. Incorporating entity and pair embeddings into a neural network to improve the performance of relation extraction. Inputs are the sentence text with the two entities as well as the pre-trained embeddings of the chemical and protein entities and the chemical-protein pair. The output is the predicted label for the pair, `down_regulator/inhibitor`, according to the sentence.

3.5.2 Method

We explore different approaches for pre-training vector representations for biomedical entities and entity pairs. Our approaches are inspired by the work of Levy and Goldberg (2014), using richer contexts to extend the skip-gram architecture of `Word2Vec` model introduced by Mikolov et al. (2013).

Pre-training entity and pair embeddings

Typically, unsupervised embedding learning methods require a vast amount of text for pre-training (Mikolov et al., 2013; Pennington et al., 2014). Therefore, we first obtain the list of all chemical-protein pairs in the Chemprot corpus and then find all sentences in PubMed and PMCOA texts that contain at least one pair. For obtaining PubMed and PMCOA texts, we use the resource provided by Hakala et al. (2016), because they have already pre-processed the data (e.g., the sentences are tokenized and parsed).

For pre-training embeddings for named entities and named-entity pairs, we use the skip-gram training of the `word2vecf` toolkit (Levy and Goldberg, 2014), whereby the named entities, or entity pairs are given as the focus terms, and elements from their contexts are predicted. We investigate two ways to define the context: a simple linear context of words as in the base `word2vec` (see Figure 11), and as an alternative, a rich set of features extracted from the sentence. These features have previously been shown to be useful in supervised relation extraction and one might therefore expect they result in embeddings informative for relation extraction. More specifically, we will rely on the Turku Event Extraction System (TEES) (Björne, 2014) to generate these features, a system that has achieved numerous top ranks in biomedical relation extraction tasks.

Sentence: $W_1, W_2, W_3, \dots, W_{t-2}, W_{t-1}, e_1, W_{t+1}, W_{t+2}, \dots, W_n$

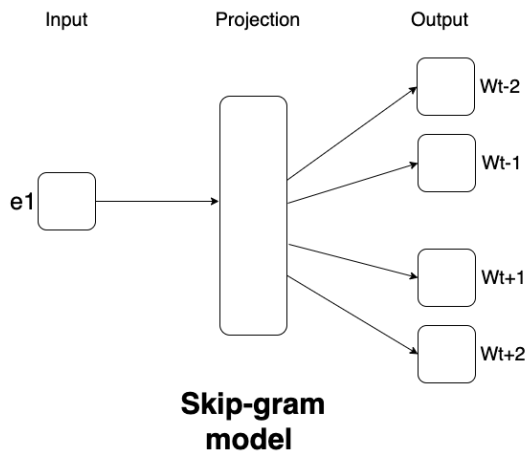


Figure 11. The skip-gram model of `word2vec`: the words surrounding entity e_1 are used as the context.

Since simultaneous training of embeddings for entities and entity pairs can impact the final model (due to the shared output layer in the `word2vec` model), we train separate embedding models that include only entity pairs, only entities, or both pairs and entities, resulting in six models (see Table 10).

Relation extraction with entity and pair embeddings

We incorporate pre-trained entity and pair embeddings to the following neural network architectures.

1. `BERT_MASK` (baseline #1): This neural network architecture was proposed by Lee et al. (2019) and achieved state-of-the-art, with an F-score of 76.46 on

| Model | Content | Context for training |
|----------|----------------------------|---|
| P_TEEES | only pair embeddings | TEES features |
| P_Words | only pair embeddings | union of the words surrounding the two entities |
| E_TEEES | only entity embeddings | union of TEEES features for every pair that includes the entity |
| E_Words | only entity embeddings | words surrounding the entity |
| PE_TEEES | pair and entity embeddings | TEES features |
| PE_Words | pair and entity embeddings | surrounding words |

Table 10. Description of the different embedding models.

ChemProt test set. This is basically a BERT encoder (Devlin et al., 2018) that has been pre-trained on PubMed abstracts and fine-tuned on the ChemProt training set for chemical-protein relation extraction. The sentence is input to the BERT encoder, and BERT sentence representation vector ($[CLS]$ token) is fetched and provided to the decision layer which has a softmax activation to decide the label. Since a sentence may include more than one chemical and/or protein mention, and the relation extraction task is performed as a sentence classification task, we need to inform the classifier which chemical and protein entities in the sentence should be focused in a given example. In BERT_MASK approach, the chemical and protein entities of focus are *masked*. More specifically, the chemical mention is replaced with @CHEMICAL\$ and the protein mention is replaced with @PROTEIN\$ (see Figure 12.1).

2. BERT_MARK (baseline #2): Since the entity and pair embeddings we pre-train provide some information about them, it would be unrealistic to obtain performance improvements by adding them to the BERT_MASK model. Therefore, we introduce BERT_MARK model, which is identical to the previous one, except instead of masking the entities, we mark them with BERT unused tokens. This should provide a fairer baseline¹³ (see Figure 12.2).
3. BERT+Entities: This architecture is similar to the BERT_MARK model, except the pre-trained chemical and protein embeddings are concatenated to the $[CLS]$ token and transformed through a 1024-dimensional dense layer with *tanh* activation and then presented to the decision layer. The dense layer with the non-linear activation function learns to combine BERT features with the entity embedding features (see Figure 12.3).

¹³In BERT_MASK model, entities are hidden, i.e., the BERT encoder and the classifier have no information about them. By feeding the embeddings of the missing entities to such model, we will obtain a performance improvement for relation extraction, but that improvement is not realistic, because the entities had been masked in the first place. Our aim is to show and conclude that feeding entity and pair embeddings to neural network models can in fact provide additional information to the classifiers, but to the classifiers that had the chance of observing chemical and protein mentions in their input. Therefore, we propose BERT_MARK model as a fairer baseline for the task.

4. **BERT+Pairs**: This architecture is similar to the **BERT+Entities** model, except we concatenate the pair embedding to the *[CLS]* token (see Figure 12.4).
5. **BERT+Pairs+Entities**: This model is similar to previous models, except we concatenate chemical, protein, and pair vectors (see Figure 12.5).

3.5.3 Results

Model selection

We evaluate our approaches on the ChemProt development set. Since the initial random weights of a neural model can affect the final F-score, we repeat each experiment (training on the ChemProt training set and evaluate the model by predicting the development set) for 10 times, resulting in 10 F-scores for each approach. We report the average and standard deviations of the F-scores and use two-tailed two-sample independent t-test (Welch’s t-test) to compare the results and establish statistical significance. Table 11 summarizes the results on the development set.

| # | Neural model | Embeddings model | F-score (mean) | F-score (std) | G1 | G2 |
|---|---------------------|----------------------|----------------|---------------|-----|-----|
| 1 | BERT_MASK | - | 78.41 | 0.53 | - | Yes |
| 2 | BERT_MARK | - | 78.96 | 0.41 | Yes | - |
| 3 | BERT+Entities | E.Words | 79.23 | 0.43 | Yes | No |
| 4 | BERT+Entities | E.TEES | 79.15 | 0.42 | Yes | No |
| 5 | BERT+Pairs | P.Words | 79.27 | 0.43 | Yes | No |
| 6 | BERT+Pairs | P.TEES | 79.36 | 0.32 | Yes | Yes |
| 7 | BERT+Pairs+Entities | PE.Words | 79.47 | 0.37 | Yes | Yes |
| 8 | BERT+Pairs+Entities | PE.TEES | 79.55 | 0.40 | Yes | Yes |
| 9 | BERT+Pairs+Entities | Randomly initialized | 79.05 | 0.46 | Yes | No |

Table 11. Results on ChemProt development set. Columns G1 and G2 show if based on the statistical test, the F-score mean is significantly different from the F-score mean of **BERT_MASK** and **BERT_MARK** models respectively.

As column G1 shows, all models that use the marking approach outperformed the masking approach (**BERT_MASK** baseline), and based on column G2, only three approaches were able to outperform the **BERT_MARK** approach (rows 6, 7, and 8). Therefore, we select these approaches for the final evaluation on the test set. We further conduct another experiment and test the effect of randomly initializing entity and pair embeddings instead of using pre-trained embeddings to check if the neural model can efficiently learn these embeddings from scratch. As can be seen in row 9, however, this approach is not able to outperform the baseline. Thus we conclude pre-training embeddings on the literature is indeed useful.

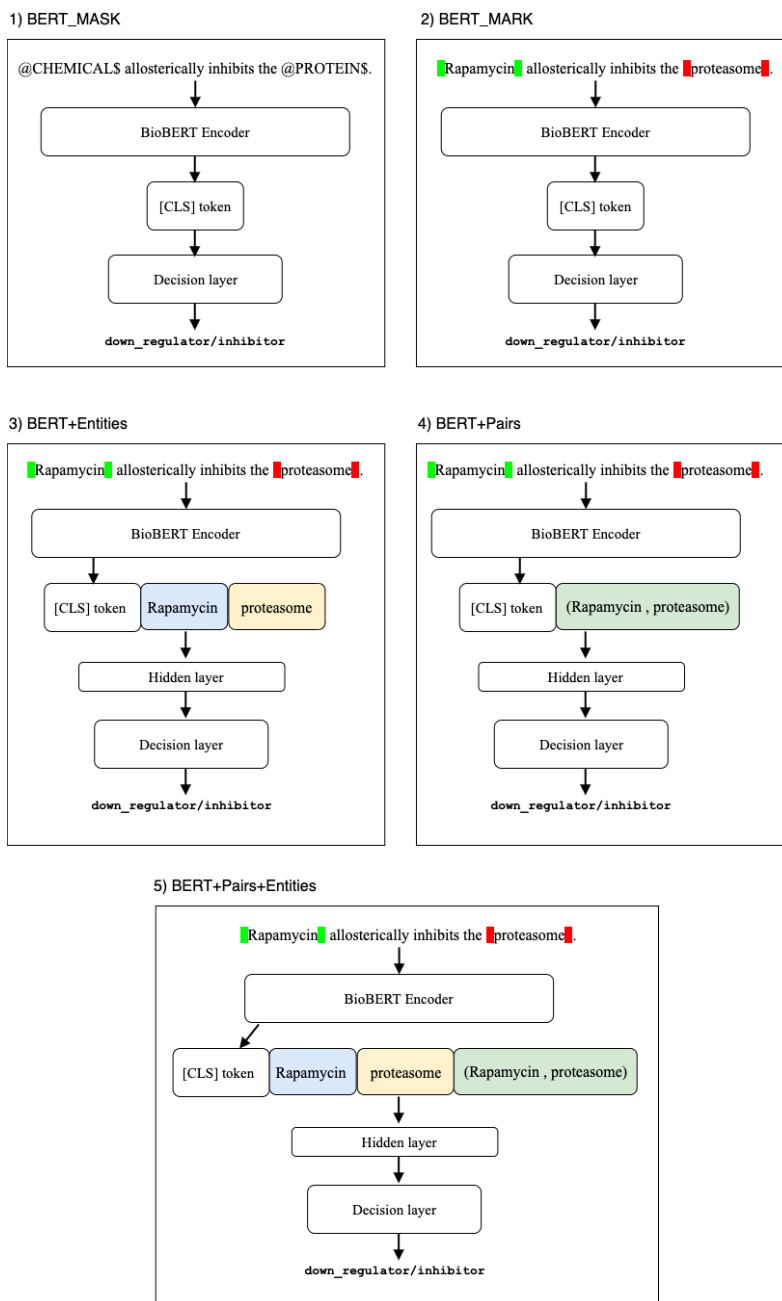


Figure 12. Relation extraction neural network architectures.

Final evaluation

We compare our best models selected on the development set (rows 6-8 in Table 11) with the best previous result of Lee et al. (2019) (an F-score of 76.46%) on the test

set. We also evaluate the BERT_MASK model to check how well we have been able to replicate the method of Lee et al. (2019). We repeat each experiment 10 times and obtain 10 F-scores and we use the one-sample t-test ($p=0.05$) to assess statistical significance.

| # | Neural model | Embeddings model | F-score (mean) | F-score (std) | G1 |
|---|---------------------|------------------|----------------|---------------|-----|
| | Lee et al. (2019) | - | 76.46 | - | - |
| 1 | BERT_MASK | - | 76.41 | 0.72 | No |
| 2 | BERT+Pairs | P_TEES | 77.13 | 0.53 | Yes |
| 3 | BERT+Pairs+Entities | PE_Words | 76.71 | 0.77 | No |
| 4 | BERT+Pairs+Entities | PE_TEES | 77.19 | 0.49 | Yes |

Table 12. Results on ChemProt test set. Column G1 shows if based on the statistical test, F-score mean is significantly different from the F-score of Lee et al. (2019)

Row 1 in Table 12 validates our replication of the Lee et al. (2019) method. Among the three approaches that used pre-trained embeddings, the model that uses surrounding words as the context (row 3) does not outperform the baseline, however the models that use TEES features (rows 2,4), outperform the best previous result, suggesting that pair-embeddings with rich feature-based context can improve upon a strong BERT-based baseline. Our best model (row 4) sets a new state-of-the-art for the task, improving the best previous score by 0.73 percentage points.

3.5.4 Discussion and conclusions

In this work, we compared different approaches for unsupervised pre-training of embeddings for biomedical named entities and named-entity pairs to improve relation extraction performance in the biomedical domain. We have shown that: (1) incorporation of entity and pair embeddings into neural models helps to achieve better performance, (2) using rich features as context (instead of using the surrounding words) leads to better results; (3) using pair embeddings with/without entity embeddings leads to better results compared to using entity embeddings alone. Our best model achieves an F-score of 77.19, improving the best previous result by +0.73 percentage point over a strong baseline, and setting a new state-of-the-art for the task.

As an additional experiment (not discussed in **Paper VI**), we investigated the effect of incorporating entity and pair embeddings to other neural network architectures. Specifically, we added entity and pair embeddings to the ST-ANN and I-ANN neural models (discussed in **Paper V**) which utilize bidirectional LSTM networks to process the words, POS-tags and dependency types in the sentence and/or along the shortest dependency path that connects the two entities in the sentence parse graph. To our surprise, entity and pair embeddings could improve the performance of those models up to 4 percentage points. However, even with such improvements, the per-

formance of LSTM models stay much behind modern neural models that utilize pre-trained transformer encoders such as BERT. Hence, I conclude that in comparison with LSTM and CNN -based architectures, the pre-trained transformers are much more powerful tools for relation extraction tasks and I predict achieving significant performance improvements on top of such architectures is going to be challenging.

4 Discussion and conclusion

4.1 Contributions of the thesis

In this thesis, I introduced novel methods to both improve the performance of biomedical relation extraction systems, and improve the quality of extracted data, after biomedical events have been detected and extracted from the biomedical literature.

Paper I introduces novel methods which can automatically and efficiently target and remove incorrect events from the output of large-scale event databases, thus improving the precision of such resources without sacrificing the recall. We showed that application of our best method on the EVEX event database results in identification of 1,338,075 incorrect events, which constitutes 3.3% of all events in EVEX. Introduction of different supervised and unsupervised methods enables other groups (with different expertise and skills) to adapt and use one of our methods to increase the perceived credibility of other large-scale event collections similar to EVEX.

Having a limited number of training and validation examples is known to represent significant challenges for leveraging the full potential of deep neural networks. When the number of weights in a neural network architecture is high and the training set is very small, the initial random state of the model can have a significant impact on the final model and its generalization performance. To measure the extent of this effect, one can repeat the development cycles (training on the training set with a fixed set of hyper-parameters and evaluating the model by predicting the validation set) a number of times and measure the standard deviation of the primary performance metric. Unfortunately, having minimal training and development data is a common case in the biomedical domain, hindering the development of robust relation extraction systems. **Paper II, III and V** introduce a few simple methods that can be used to stabilize the output, overcome the variance and build robust and efficient relation extraction systems for the biomedical domain. Our ensemble voting approach ranked the second among the entries to the BB3-event shared task, being the best performing deep learning-based method in this task (**Paper II**). Similarly, on BB3-event+ner task, the joint evaluation of our CNN-based relation extraction system and our NER system, showed 19 percentage points improvement over the winning entry this task, setting a new state-of-the-art score for the task (**Paper III**).

Focusing on the entities that occur in the same sentence, relation extraction approaches can be broadly divided into three main categories based on the type of context that they use: (1) methods that only rely on SDP features, (2) methods that

rely on full sentence tokens, and (3) methods that utilize SDP features as well as features that are extracted from the whole sentence. In this thesis I explored the effect of using different types of contexts for relation extraction and showed that utilizing full sentence tokens alongside with the SDP features can actually improve the performance of relation extraction (**Paper V**). This suggests there are clues outside of the SDP which can help in characterizing the semantic relationships between the candidate entity pairs, therefore, I encourage NLP practitioners to not solely rely on SDP features for building state-of-the-art relation extraction systems.

Many successful systems are built based on combining feature-based and deep learning-based methods to improve the performance of relation extraction. This combination can be achieved in two levels: (1) system-level combination, i.e., combining the predictions of various systems, and (2) feature-level combination, e.g., incorporating engineered features into neural network models. In this thesis, I explored existing hybrid relation extraction methods and discussed two hybrid methods that we developed to improving the performance. **Paper V** introduces a simple method to combine the predictions of feature-based systems with neural network models. During our participation in the BioCreative VI ChemProt track, our best result (the fifth rank in the task with 60.99 F-score), was achieved by our feature-based system. After the shared task, and by improving our deep learning-based system and combining its prediction with the feature-based system, we improved our previous results by 2.11 percentage points. **Paper IV** proposes to utilize dimensionality reduction techniques and use the lower-dimensional resulting vectors as additional inputs to neural network models to improve the performance of the text mining task at hand. Based on this idea, we improved the performance of our CNN-based system by 2 percentage points on the development and the test set and achieved the 5th place in the 2nd Social Media Mining for Health Applications Shared Task - Task 2. I also discussed and estimated that, we would have increased our system performance by 0.7pp in F-score (ranking our system higher in the shared task), if we had access to the full training data.

Finally, **Paper VI** introduces a novel unsupervised method to pre-train embeddings for biomedical entity and entity-pairs to improve the performance of relation extraction. We showed that by incorporating such embeddings into neural network models, higher performances can be achieved. In particular, our proposed neural network model that utilizes a BERT encoder alongside with pre-trained entity and entity-pair embeddings achieved an F-score of 77.19 on the ChemProt corpus, setting a new state-of-the-art for the task.

4.2 Future directions for biomedical relation extraction

Two decades ago, feature-based text mining methods were dominant and extremely popular. Therefore, the majority of BioNLP researchers had focused on heavy feature

engineering, designing and extracting various creative features to improve the performance of text mining systems. The significant advancements in the hardware technology during the last decade, as well as several breakthroughs in the algorithms and methods, enabled machine learning practitioners to seriously utilize artificial neural networks for real-world applications. The deep learning approach allowed computers to automatically learn efficient representations of their inputs that are suitable for the machine learning task at hand, hence, the focus of the BioNLP community shifted from feature engineering to neural network design and unsupervised pre-training. The field gradually underwent a significant progress, with deep learning-based methods showing superior performance on many NLP and text mining tasks. Recently, the introduction of transformer-based language representation models such as BERT impacted the field and resulted in unprecedented performance improvements on many data sets. I believe deep learning-based methods are to stay, at least for the upcoming decade, and I predict future developments and progress in biomedical text mining will be along this direction. For achieving higher performance in relation and event extraction tasks, I can enumerate the followings for possible future work.

1. Designing better language representation models and/or better neural network architectures.
2. Designing better unsupervised methods to pre-train neural network components (e.g. encoders) on biomedical literature.
3. Finding suitable auxiliary tasks for multi-task learning setups to improve relation extraction performance, as well as designing better loss-functions for training neural network models in such setups.
4. Unification of different relation extraction corpora to obtain more training data.
5. Addressing cross-sentence relation extraction by natural extension of existing sentence encoder-based methods to receive a full abstracts or a paragraph instead of processing a single sentence at a time.

Bibliography

- Achahanuparp, P., Hu, X., and Shen, X. (2008). The evaluation of sentence similarity measures. In Song, I.-Y., Eder, J., and Nguyen, T. M., editors, *Data Warehousing and Knowledge Discovery*, pages 305–316, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Airola, A., Pyysalo, S., Björne, J., Pahikkala, T., Ginter, F., and Salakoski, T. (2008). All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, 9(11):S2.
- Akkasi, A., Varoglu, E., and Dimililer, N. (2016). Chemtok: A new rule based tokenizer for chemical named entity recognition. *BioMed research international*, 2016:4248026–4248026.
- Arighi, C. N., Wu, C. H., Cohen, K. B., Hirschman, L., Krallinger, M., Valencia, A., Lu, Z., Wilbur, J. W., and Wieggers, T. C. (2014). Biocreative-iv virtual issue. *Database : the journal of biological databases and curation*, 2014:bau039.
- Barrett, N. and Weber-Jahnke, J. (2011). Building a biomedical tokenizer using the token lattice design pattern and the adapted viterbi algorithm. *BMC bioinformatics*, 12 Suppl 3(Suppl 3):S1–S1.
- Baumgartner, W., Bada, M., Pyysalo, S., Ciosici, M. R., Hailu, N., Pielke-Lombardo, H., Regan, M., and Hunter, L. (2019). CRAFT shared tasks 2019 overview — integrated structure, semantics, and coreference. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 174–184, Hong Kong, China. Association for Computational Linguistics.
- Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.
- Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. (2013). Genbank. *Nucleic acids research*, 41(Database issue):D36–D42.
- Björne, J. (2014). *Biomedical Event Extraction with Machine Learning*. PhD thesis, TUCS Dissertations.
- Björne, J., Ginter, F., and Salakoski, T. (2012). University of turku in the bionlp’11 shared task. *BMC bioinformatics*, 13 Suppl 11(Suppl 11):S4–S4. 22759458[pmid].
- Björne, J. and Salakoski, T. (2013). TEES 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25. Association for Computational Linguistics.
- Björne, J. and Salakoski, T. (2018). Biomedical event extraction using convolutional neural networks and dependency parsing. In *Proceedings of the BioNLP 2018 workshop*, pages 98–108, Melbourne, Australia. Association for Computational Linguistics.
- Bossy, R., Deléger, L., Chaix, E., Ba, M., and Nédellec, C. (2019). Bacteria biotope at BioNLP open shared tasks 2019. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 121–131, Hong Kong, China. Association for Computational Linguistics.
- Bossy, R., Golik, W., Ratkovic, Z., Bessières, P., and Nédellec, C. (2013). BioNLP shared task 2013 – an overview of the bacteria biotope task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 161–169, Sofia, Bulgaria. Association for Computational Linguistics.
- Bossy, R., Jourde, J., Bessières, P., van de Guchte, M., and Nédellec, C. (2011). BioNLP shared task 2011 - bacteria biotope. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 56–64, Portland, Oregon, USA. Association for Computational Linguistics.
- Boudjellal, N., Zhang, H., Khan, A., and Ahmad, A. (2020). Biomedical relation extraction using distant supervision. *Scientific Programming*, 2020:8893749.

- Brown, P. F., Lai, J. C., and Mercer, R. L. (1991). Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, ACL '91, pages 169–176, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Bunescu, R. and Mooney, R. (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, British Columbia, Canada. Association for Computational Linguistics.
- Can, D.-C., Le, H.-Q., Ha, Q.-T., and Collier, N. (2019). A richer-but-smarter shortest dependency path with attentive augmentation for relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2902–2912, Minneapolis, Minnesota. Association for Computational Linguistics.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.
- Choi, J. D. (2016). Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 271–281, San Diego, California. Association for Computational Linguistics.
- Chowdhury, F. M., Lavelli, A., and Moschitti, A. (2011). A study on dependency tree kernels for automatic extraction of protein-protein interaction. In *Proceedings of BioNLP 2011 Workshop*, pages 124–133, Portland, Oregon, USA. Association for Computational Linguistics.
- Cohen, K. B., Lanfranchi, A., Choi, M. J.-Y., Bada, M., Baumgartner, W. A. J., Panteleyeva, N., Verpoor, K., Palmer, M., and Hunter, L. E. (2017). Coreference annotation and resolution in the colorado richly annotated full text (craft) corpus of biomedical journal articles. *BMC bioinformatics*, 18(1):372–372.
- Crawshaw, M. (2020). Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*.
- Crichton, G., Pyysalo, S., Chiu, B., and Korhonen, A. (2017). A neural network multi-task learning approach to biomedical named entity recognition. *BMC Bioinformatics*, 18(1):368.
- Cruz Díaz, N. P. and Maña López, M. (2015). An analysis of biomedical tokenization: Problems and strategies. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 40–49, Lisbon, Portugal. Association for Computational Linguistics.
- Culotta, A. and Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 423–429, Barcelona, Spain.
- Davis, A. P., Wieggers, T. C., Rosenstein, M. C., and Mattingly, C. J. (2012). Medic: a practical disease vocabulary used at the comparative toxicogenomics database. *Database : the journal of biological databases and curation*, 2012:bar065–bar065.
- Deléger, L., Bossy, R., Chaix, E., Ba, M., Ferré, A., Bessières, P., and Nédellec, C. (2016). Overview of the bacteria biotope task at BioNLP shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.
- Deng, P., Chen, H., Huang, M., Ruan, X., and Xu, L. (2019). An ensemble CNN method for biomedical entity normalization. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 143–149, Hong Kong, China. Association for Computational Linguistics.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- dos Santos, C. and Guimarães, V. (2015). Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33, Beijing, China. Association for Computational Linguistics.

- Emadzadeh, E., Nikfarjam, A., and Gonzalez, G. (2011). Double layered learning for biological event extraction from text. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 153–154, Portland, Oregon, USA. Association for Computational Linguistics.
- Federhen, S. (2012). The ncbi taxonomy database. *Nucleic acids research*, 40(Database issue):D136–D143.
- Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.
- Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.
- Fu, L., Nguyen, T. H., Min, B., and Grishman, R. (2017). Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–429, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Gerner, M., Sarafraz, F., Bergman, C. M., and Nenadic, G. (2012). Biocontext: an integrated text mining system for large-scale extraction and contextualization of biomolecular events. *Bioinformatics (Oxford, England)*, 28(16):2154–2161.
- Gonzalez-Agirre, A., Marimon, M., Intxaurreondo, A., Rabal, O., Villegas, M., and Krallinger, M. (2019). PharmaCoNER: Pharmacological substances, compounds and proteins named entity recognition track. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Gridach, M. (2017). Character-level neural network for biomedical named entity recognition. *Journal of Biomedical Informatics*, 70:85–91.
- Grouin, C. (2016). Identification of mentions and relations between bacteria and biotope from PubMed abstracts. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 64–72, Berlin, Germany. Association for Computational Linguistics.
- Hakala, K., Kaewphan, S., Bjorne, J., Mehryary, F., Moen, H., Tolvanen, M., Salakoski, T., and Ginter, F. (2020). Neural Network and Random Forest Models in Protein Function Prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1.
- Hakala, K., Kaewphan, S., Salakoski, T., and Ginter, F. (2016). Syntactic analyses and named entity recognition for PubMed and PubMed Central –up-to-the-minute. In *Proceedings of the 2016 Workshop on Biomedical Natural Language Processing*, pages 102–107. Association for Computational Linguistics.
- Hakala, K., Mehryary, F., Kaewphan, S., and Ginter, F. (2013a). Hypothesis generation in large-scale event networks. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine (LBM'13)*, pages 19–28.
- Hakala, K., Mehryary, F., Moen, H., Kaewphan, S., Salakoski, T., and Ginter, F. (2017). Ensemble of convolutional neural networks for medicine intake recognition in twitter. In *Proceedings of the 2nd Social Media Mining for Health Research and Applications (SMM4H 2017) Workshop*, pages 59–63. CEUR-WS.org. (equal contribution between Hakala and Mehryary).
- Hakala, K., Van Landeghem, S., Salakoski, T., Van de Peer, Y., and Ginter, F. (2013b). EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In *Proceedings of the BioNLP Shared Task 2013 Workshop (BioNLP-ST'13)*, pages 26–34.
- Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288.
- Hastings, J., Owen, G., Dekker, A., Ennis, M., Kale, N., Muthukrishnan, V., Turner, S., Swainston, N., Mendes, P., and Steinbeck, C. (2016). Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic acids research*, 44(D1):D1214–9.
- He, Y. and Kayaalp, M. (2006). A comparison of 13 tokenizers on medline. Technical Report LHCNBC-TR-2006-003, The Lister Hill National Center for Biomedical Communications.

- Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2010). SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Ji, Z., Wei, Q., and Xu, H. (2019). Bert-based ranking for biomedical entity normalization. *arXiv preprint arXiv:1908.03548*.
- Jiang, Y., Oron, R. T., Clark, T. W., Bankapur, R. A., D’Andrea, D., Lepore, R., Funk, S. C., Kahanda, I., Verspoor, M. K., Ben-Hur, A., Koo, E. D. C., Penfold-Brown, D., Shasha, D., Youngs, N., Bonneau, R., Lin, A., Sahraeian, E. S. M., Martelli, L. P., Profiti, G., Casadio, R., Cao, R., Zhong, Z., Cheng, J., Altenhoff, A., Skunca, N., Dessimoz, C., Dogan, T., Hakala, K., Kaewphan, S., Mehryary, F., Salakoski, T., Ginter, F., Fang, H., Smithers, B., Oates, M., Gough, J., Törönen, P., Koskinen, P., Holm, L., Chen, C.-T., Hsu, W.-L., Bryson, K., Cozzetto, D., Minneci, F., Jones, T. D., Chapman, S., BKC, D., Khan, K. I., Kihara, D., Ofer, D., Rappoport, N., Stern, A., Cibrian-Uhalte, E., Denny, P., Foulger, E. R., Hieta, R., Legge, D., Lovering, C. R., Magrane, M., Melidoni, N. A., Mutowo-Meullenet, P., Pichler, K., Shypitsyna, A., Li, B., Zakeri, P., ElShal, S., Tranchevent, L.-C., Das, S., Dawson, L. N., Lee, D., Lees, G. J., Sillitoe, I., Bhat, P., Nepusz, T., Romero, E. A., Sasidharan, R., Yang, H., Paccanaro, A., Gillis, J., Sedeño-Cortés, E. A., Pavlidis, P., Feng, S., Cejuela, M. J., Goldberg, T., Hamp, T., Richter, L., Salamov, A., Gabaldon, T., Marcet-Houben, M., Supek, F., Gong, Q., Ning, W., Zhou, Y., Tian, W., Falda, M., Fontana, P., Lavezzo, E., Toppo, S., Ferrari, C., Giollo, M., Piovesan, D., Tosatto, C. S., del Pozo, A., Fernández, M. J., Maietta, P., Valencia, A., Tress, L. M., Benso, A., Di Carlo, S., Politano, G., Savino, A., Rehman, U. H., Re, M., Mesiti, M., Valentini, G., Bargsten, W. J., van Dijk, J. A. D., Gemovic, B., Glisic, S., Perovic, V., Veljkovic, V., Veljkovic, N., Almeida-e Silva, C. D., Vencio, N. R. Z., Sharan, M., Vogel, J., Kansakar, L., Zhang, S., Vucetic, S., Wang, Z., Sternberg, E. M. J., Wass, N. M., Huntley, P. R., Martin, J. M., O’Donovan, C., Robinson, N. P., Moreau, Y., Tramontano, A., Babbitt, C. P., Brenner, E. S., Linial, M., Orengo, A. C., Rost, B., Greene, S. C., Mooney, D. S., Friedberg, I., and Radivojac, P. (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(1):1–19.
- Jin-Dong, K., Claire, N., Robert, B., and Louise, D., editors (2019). *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, Hong Kong, China. Association for Computational Linguistics.
- Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google’s multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.
- Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Kaewphan, S., Mehryary, F., Hakala, K., Salakoski, T., and Ginter, F. (2017). TurkuNLP entry for interactive Bio-ID assignment. In *Proceedings of the BioCreative VI Workshop*, pages 32–35.
- Kanerva, J., Ginter, F., Miekka, N., Leino, A., and Salakoski, T. (2018). Turku neural parser pipeline: An end-to-end system for the CoNLL 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 133–142, Brussels, Belgium. Association for Computational Linguistics.
- Kanerva, J., Ginter, F., and Salakoski, T. (2020). Universal lemmatizer: A sequence-to-sequence model for lemmatizing universal dependencies treebanks. *Natural Language Engineering*, page 1–30.
- Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2009). Overview of BioNLP’09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado. Association for Computational Linguistics.

- Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. (2011a). EXTRACTING BIO-MOLECULAR EVENTS FROM LITERATURE—THE BIONLP’09 SHARED TASK. *Computational Intelligence*, 27(4):513–540.
- Kim, J.-D., Ohta, T., and Tsujii, J. (2008). Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(1):10.
- Kim, J.-D., Pyysalo, S., Ohta, T., Bossy, R., Nguyen, N., and Tsujii, J. (2011b). Overview of BioNLP shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 1–6, Portland, Oregon, USA. Association for Computational Linguistics.
- Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., Zaslavsky, L., Zhang, J., and Bolton, E. E. (2019). Pubchem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1):D1102–D1109.
- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Kondratyuk, D. and Straka, M. (2019). 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.
- Krallinger, M., Leitner, F., Rodríguez-Penagos, C., and Valencia, A. (2008). Overview of the protein-protein interaction annotation extraction task of biocreative ii. *Genome Biology*, 9(2):S4.
- Krallinger, M., Rabal, O., Akhondi, S. A., Pérez-Pérez, M., Santamaría, J., et al. (2017). Overview of the BioCreative VI chemical-protein interaction Track. In *Proceedings of the BioCreative VI Challenge Evaluation Workshop*, volume 1, pages 141–146, Bethesda, MD, USA.
- Krallinger, M., Rabal, O., Leitner, F., Vazquez, M., Salgado, D., Lu, Z., Leaman, R., Lu, Y., Ji, D., Lowe, D. M., Sayle, R. A., Batista-Navarro, R. T., Rak, R., Huber, T., Rocktäschel, T., Matos, S., Campos, D., Tang, B., Xu, H., Munkhdalai, T., Ryu, K. H., Ramanan, S. V., Nathan, S., Zitnik, S., Bajec, M., Weber, L., Irmer, M., Akhondi, S. A., Kors, J. A., Xu, S., An, X., Sikdar, U. K., Ekbal, A., Yoshioka, M., Dieb, T. M., Choi, M., Verspoor, K., Khabsa, M., Giles, C. L., Liu, H., Ravikumar, K. E., Lamurias, A., Couto, F. M., Dai, H.-J., Tsai, R. T.-H., Ata, C., Can, T., Usié, A., Alves, R., Segura-Bedmar, I., Martínez, P., Oyarzabal, J., and Valencia, A. (2015). The CHEMDNER corpus of chemicals and drugs and its annotation principles. *Journal of Cheminformatics*, 7(1):S2.
- Krallinger, M., Vazquez, M., Leitner, F., Salgado, D., Chatr-Aryamontri, A., Winter, A., Perfetto, L., Briganti, L., Licata, L., Iannuccelli, M., Castagnoli, L., Cesareni, G., Tyers, M., Schneider, G., Rinaldi, F., Leaman, R., Gonzalez, G., Matos, S., Kim, S., Wilbur, W. J., Rocha, L., Shatkay, H., Tendulkar, A. V., Agarwal, S., Liu, F., Wang, X., Rak, R., Noto, K., Elkan, C., Lu, Z., Dogan, R. I., Fontaine, J.-F., Andrade-Navarro, M. A., and Valencia, A. (2011). The protein-protein interaction tasks of biocreative iii: classification/ranking of articles and linking bio-ontology concepts to full text. *BMC bioinformatics*, 12 Suppl 8(Suppl 8):S3–S3.
- Kreuzthaler, M. and Schulz, S. (2015). Detection of sentence boundaries and abbreviations in clinical narratives. *BMC Medical Informatics and Decision Making*, 15(2):S4.
- Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.
- Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American*

- Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., et al. (2019). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.
- Lever, J. and Jones, S. J. (2016). VERSE: Event and relation extraction in the BioNLP 2016 shared task. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 42–49, Berlin, Germany. Association for Computational Linguistics.
- Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.
- Li, C., Rao, Z., Zheng, Q., and Zhang, X. (2018a). A set of domain rules and a deep network for protein coreference resolution. *Database : the journal of biological databases and curation*, 2018:bay065.
- Li, H., Chen, Q., Tang, B., Wang, X., Xu, H., Wang, B., and Huang, D. (2017). Cnn-based ranking for biomedical entity normalization. *BMC bioinformatics*, 18(Suppl 11):385–385.
- Li, J., Sun, A., Han, J., and Li, C. (2018b). A survey on deep learning for named entity recognition. *arXiv preprint arXiv:1812.09449*.
- Li, Z., Yang, Z., Shen, C., Xu, J., Zhang, Y., and Xu, H. (2019). Integrating shortest dependency path and sentence sequence into a deep learning framework for relation extraction in clinical text. *BMC Medical Informatics and Decision Making*, 19(Suppl 1):22.
- Lim, S., Lee, K., and Kang, J. (2018). Drug drug interaction extraction from the literature using a recursive neural network. *PLOS ONE*, 13(1):1–17.
- Liu, H., Christiansen, T., Baumgartner, W. A. J., and Verspoor, K. (2012). Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *Journal of biomedical semantics*, 3:3–3.
- Liu, H., Komandur, R., and Verspoor, K. (2011). From graphs to events: A subgraph matching approach for information extraction from biomedical text. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 164–172, Portland, Oregon, USA. Association for Computational Linguistics.
- Liu, S., Kai Chen, Chen, Q., and Tang, B. (2016). Dependency-based convolutional neural network for drug-drug interaction extraction. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1074–1080.
- Liu, Y., Wei, F., Li, S., Ji, H., Zhou, M., and Wang, H. (2015). A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290, Beijing, China. Association for Computational Linguistics.
- Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.
- Maglott, D., Ostell, J., Pruitt, K. D., and Tatusova, T. (2011). Entrez gene: gene-centered information at ncbi. *Nucleic acids research*, 39(Database issue):D52–D57.
- Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- McClosky, D., Surdeanu, M., and Manning, C. (2011). Event extraction as dependency parsing for BioNLP 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 41–45, Portland, Oregon, USA. Association for Computational Linguistics.
- Mehryary, F., Björne, J., Pyysalo, S., Salakoski, T., and Ginter, F. (2016a). Deep learning with minimal training data: TurkuNLP entry in the BioNLP shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 73–81, Berlin, Germany. Association for Computational Linguistics.

- Mehryary, F., Björne, J., Salakoski, T., and Ginter, F. (2017a). Combining support vector machines and LSTM networks for chemical-protein relation extraction. In *Proceedings of the BioCreative VI Workshop*, pages 175–179.
- Mehryary, F., Björne, J., Salakoski, T., and Ginter, F. (2018). Potent pairing: ensemble of long short-term memory networks and support vector machine for chemical-protein relation extraction. *Database : the journal of biological databases and curation*, 2018:bay120.
- Mehryary, F., Hakala, K., Kaewphan, S., Björne, J., Salakoski, T., and Ginter, F. (2017b). End-to-end system for bacteria habitat extraction. In *BioNLP 2017*, pages 80–90, Vancouver, Canada. Association for Computational Linguistics.
- Mehryary, F., Kaewphan, S., Hakala, K., and Ginter, F. (2014). Eliminating incorrect events from large-scale event networks by trigger word clustering and pruning. In *Proceedings of the 6th International Symposium on Semantic Mining in Biomedicine (SMBM 2014)*, pages 75–79.
- Mehryary, F., Kaewphan, S., Hakala, K., and Ginter, F. (2016b). Filtering large-scale event collections using a combination of supervised and unsupervised learning for event trigger classification. *Journal of Biomedical Semantics*, 7(1):1–13.
- Mehryary, F., Moen, H., Salakoski, T., and Ginter, F. (2020). Entity-pair embeddings for improving relation extraction in the biomedical domain. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020 (online event)*, pages 613–618. i6doc.com publication.
- Mikheev, A. (2000). Tagging sentence boundaries. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference, NAACL 2000*, pages 264–271, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.
- Minnen, G., Carroll, J., and Pearce, D. (2001). Applied morphological processing of english. *Nat. Lang. Eng.*, 7(3):207–223.
- Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.
- Miwa, M., Sætre, R., Kim, J.-D., and Tsujii, J. (2010). Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology*, 08(01):131–146.
- Moen, H., Hakala, K., Mehryary, F., Peltonen, L.-M., Salakoski, T., Ginter, F., and Salanterä, S. (2017). Detecting mentions of pain and acute confusion in Finnish clinical text. In *Proceedings of the 2017 Workshop on Biomedical Natural Language Processing*, pages 365–372, Vancouver, Canada. Association for Computational Linguistics.
- Mooney, R. J. and Bunescu, R. C. (2006). Subsequence kernels for relation extraction. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press.
- Mueller, T., Schmid, H., and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.
- Nédellec, C., Bossy, R., Kim, J.-D., Kim, J.-j., Ohta, T., Pyysalo, S., and Zweigenbaum, P. (2013). Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7, Sofia, Bulgaria. Association for Computational Linguistics.
- Ngo, T. M., Kanerva, J., Ginter, F., and Pyysalo, S. (2019). Neural dependency parsing of biomedical text: TurkuNLP entry in the CRAFT structural annotation task. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 206–215, Hong Kong, China. Association for Computational Linguistics.
- Nguyen, D. Q. and Verspoor, K. (2019). From pos tagging to dependency parsing for biomedical event extraction. *BMC Bioinformatics*, 20(1):72.

- Nguyen, N., Kim, J.-D., and Tsujii, J. (2011). Overview of BioNLP 2011 protein coreference shared task. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 74–82, Portland, Oregon, USA. Association for Computational Linguistics.
- Nguyen, T. H. and Grishman, R. (2015). Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado. Association for Computational Linguistics.
- Nguyen, T.-V. T., Moschitti, A., and Ricciardi, G. (2009). Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Singapore. Association for Computational Linguistics.
- Ohta, T., Tateisi, Y., Kim, J.-D., Yakushiji, A., and Tsujii, J.-i. (2006). Linguistic and biological annotations of biological interaction events. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).
- Peng, N., Poon, H., Quirk, C., Toutanova, K., and Yih, W.-t. (2017). Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.
- Peng, Y. and Lu, Z. (2017). Deep learning for extracting protein-protein interactions from biomedical literature. In *BioNLP 2017*, pages 29–38, Vancouver, Canada. Association for Computational Linguistics.
- Peng, Y., Rios, A., Kavuluru, R., and Lu, Z. (2018). Extracting chemical–protein relations with ensembles of SVM and deep learning models. *Database*, 2018. bay073.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Pyysalo, S., Ginter, F., Heimonen, J., Björne, J., Boberg, J., Järvinen, J., and Salakoski, T. (2007). BioInfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(1):50.
- Pyysalo, S., Ginter, F., Moen, H., Salakoski, T., and Ananiadou, S. (2013). Distributional semantic resources for biomedical text mining. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine*, pages 39–44.
- Quan, C., Hua, L., Sun, X., and Bai, W. (2016). Multichannel convolutional neural network for biological relation extraction. *BioMed Research International*, 2016:1850404.
- Quirk, C. and Poon, H. (2017). Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182, Valencia, Spain. Association for Computational Linguistics.
- Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *Technical report, OpenAI Blog*.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *Technical report, OpenAI Blog*.
- Raihani, A. and Laachfoubi, N. (2016). Extracting drug-drug interactions from biomedical text using a feature-based kernel approach. *Journal of Theoretical and Applied Information Technology*, 92:109–120.
- Reichartz, F., Korte, H., and Paass, G. (2009). Dependency tree kernels for relation extraction from natural language text. In *Proceedings of the 2009th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II, ECMLPKDD'09*, page 270–285, Berlin, Heidelberg. Springer-Verlag.
- Riedel, S., Yao, L., and McCallum, A. (2010). Modeling relations and their mentions without labeled text. In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg. Springer Berlin Heidelberg.

- Sarker, A., Belousov, M., Friedrichs, J., Hakala, K., Kiritchenko, S., Mehryary, F., Han, S., Tran, T., Rios, A., Kavuluru, R., de Bruijn, B., Ginter, F., Mahata, D., Mohammad, S. M., Nenadic, G., and Gonzalez-Hernandez, G. (2018). Data and systems for medication-related text classification and concept normalization from Twitter: insights from the Social Media Mining for Health (SMM4H)-2017 shared task. *Journal of the American Medical Informatics Association*, 25(10):1274–1283.
- Sarker, A. and Gonzalez, G. (2017). A corpus for mining drug-related knowledge from twitter chatter: Language models and their utilities. *Data in Brief*, 10:122 – 131.
- Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.
- Segura-Bedmar, I., Martínez, P., and Herrero-Zazo, M. (2013). SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA. Association for Computational Linguistics.
- Segura-Bedmar, I., Martínez, P., and Sánchez-Cisneros, D. (2011). The 1st DDIExtraction-2011 challenge task: Extraction of Drug-Drug Interactions from biomedical texts. In *Proceedings of the 1st Challenge Task on Drug-Drug Interaction Extraction 2011*, volume 2011, pages 1–9, Huelva, Spain.
- Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Shatkay, H. and Craven, M. (2012). *Mining the Biomedical Literature*. Mit Press.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.
- Straka, M., Hajič, J., and Straková, J. (2016). UDPipe: Trainable pipeline for processing CoNLL-u files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).
- Straka, M. and Straková, J. (2017). Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.
- Sun, X., Dong, K., Ma, L., Sutcliffe, R., He, F., Chen, S., and Feng, J. (2019). Drug-drug interaction extraction via recurrent hybrid convolutional neural networks with an improved focal loss. *Entropy*, 21:37.
- Surdeanu, M., Tibshirani, J., Nallapati, R., and Manning, C. D. (2012). Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea. Association for Computational Linguistics.
- Tang, C., Chen, W., Wang, T., Sun, C., Jiang, J., and Guan, Y. (2021). Normcg: A novel deep learning model for medical entity linking. In Satapathy, S. C., Zhang, Y.-D., Bhateja, V., and Majhi, R., editors, *Intelligent Data Engineering and Analytics*, pages 565–573, Singapore. Springer Singapore.
- Tateisi, Y., Yakushiji, A., Ohta, T., and Tsujii, J. (2005). Syntax annotation for the GENIA corpus. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*.
- Trieu, H.-L., Tran, T. T., Duong, K. N. A., Nguyen, A., Miwa, M., and Ananiadou, S. (2020). Deep-EventMine: end-to-end neural nested event extraction from biomedical texts. *Bioinformatics*, 36(19):4910–4917.

- Tsuruoka, Y., Tateishi, Y., Kim, J.-D., Ohta, T., McNaught, J., Ananiadou, S., and Tsujii, J. (2005). Developing a robust part-of-speech tagger for biomedical text. In Bozanis, P. and Houstis, E. N., editors, *Advances in Informatics*, pages 382–392, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Van Landeghem, S., Björne, J., Wei, C.-H., Hakala, K., Pyysalo, S., Ananiadou, S., Kao, H.-Y., Lu, Z., Salakoski, T., Van de Peer, Y., and Ginter, F. (2013a). Large-scale event extraction from literature with multi-level gene normalization. *PLoS one*, 8(4):e55814–e55814.
- Van Landeghem, S., Kaewphan, S., Ginter, F., and Van de Peer, Y. (2013b). Evaluating large-scale text mining applications beyond the traditional numeric performance measures. In *Proceedings of the 2013 Workshop on Biomedical Natural Language Processing*, pages 63–71, Sofia, Bulgaria. Association for Computational Linguistics.
- Verspoor, K., Cohen, K. B., Lanfranchi, A., Warner, C., Johnson, H. L., Roeder, C., Choi, J. D., Funk, C., Malenkiy, Y., Eckert, M., Xue, N., Baumgartner, W. A., Bada, M., Palmer, M., and Hunter, L. E. (2012). A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools. *BMC Bioinformatics*, 13(1):207.
- Vlachos, A. and Craven, M. (2011). Biomedical event extraction from abstracts and full papers using search-based structured prediction. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 36–40, Portland, Oregon, USA. Association for Computational Linguistics.
- Wei, C.-H., Peng, Y., Leaman, R., Davis, A. P., Mattingly, C. J., Li, J., Wieggers, T. C., and Lu, Z. (2016). Assessing the state of the art in biomedical relation extraction: overview of the BioCreative V chemical-disease relation (CDR) task. *Database*, 2016. baw032.
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Xiong, Y., Shen, Y., Huang, Y., Chen, S., Tang, B., Wang, X., Chen, Q., Yan, J., and Zhou, Y. (2019). A deep learning-based system for PharmaCoNER. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 33–37, Hong Kong, China. Association for Computational Linguistics.
- Xuan, W., Watson, S. J., and Meng, F. (2007). Tagging sentence boundaries in biomedical literature. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, pages 186–195, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Yadav, V. and Bethard, S. (2018). A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.
- Zelenko, D., Aone, C., and Richardella, A. (2002). Kernel methods for relation extraction. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 71–78. Association for Computational Linguistics.
- Zeng, D., Liu, K., Lai, S., Zhou, G., and Zhao, J. (2014). Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.
- Zhang, Q., Chen, M., and Liu, L. (2017). A review on entity relation extraction. In *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 178–183.
- Zhang, Y., Zheng, W., Lin, H., Wang, J., Yang, Z., and Dumontier, M. (2018). Drug-drug interaction extraction via hierarchical RNNs on sequence and shortest dependency paths. *Bioinformatics*, 34(5):828–835.

- Zhao, Z., Yang, Z., Luo, L., Lin, H., and Wang, J. (2016). Drug drug interaction extraction from biomedical literature using syntax convolutional neural network. *Bioinformatics*, 32(22):3444–3453.
- Zhou, D., Miao, L., and He, Y. (2018). Position-aware deep multi-task learning for drug–drug interaction extraction. *Artificial Intelligence in Medicine*, 87:1–8.
- Zhou, N., Jiang, Y., Bergquist, T., Lee, A., Kacsóh, B., Crocker, A., Lewis, K., Georghiou, G., Nguyen, H., Hamid, M., Davis, L., Dogan, T., Atalay, V., Rifaioğlu, A., Dalkran, A., Cetin Atalay, R., Zhang, C., Hurto, R., Freddolino, P., Zhang, Y., Bhat, P., Supek, F., Fernández, J., Gemovic, B., Perovic, V., Davidović, R., Sumonja, N., Veljkovic, N., Asgari, E., Mofrad, M., Profiti, G., Savojardo, C., Martelli, P., Casadio, R., Boecker, F., Schoof, H., Kahanda, I., Thurlby, N., McHardy, A., Renaux, A., Saidi, R., Gough, J., Freitas, A., Antczak, M., Fabris, F., Wass, M., Hou, J., Cheng, J., Wang, Z., Romero, A., Paccanaro, A., Yang, H., Goldberg, T., Zhao, C., Holm, L., Törönen, P., Medlar, A., Zosa, E., Borukhov, I., Novikov, I., Wilkins, A., Lichtarge, O., Chi, P., Tseng, W., Linial, M., Rose, P., Dessimoz, C., Vidulin, V., Dzeroski, S., Sillitoe, I., Das, S., Lees, J., Jones, D., Wan, C., Cozzetto, D., Fa, R., Torres, M., Warwick Vesztröcy, A., Rodriguez, J., Tress, M., Frasca, M., Notaro, M., Grossi, G., Petrini, A., Re, M., Valentini, G., Mesiti, M., Roche, D., Reeb, J., Ritchie, D., Aridhi, S., Alborzi, S., Devignes, M., Koo, D., Bonneau, R., Gligorijević, V., Barot, M., Fang, H., Toppo, S., Lavezzo, E., Falda, M., Berselli, M., Tosatto, S., Carraro, M., Piovesan, D., Ur Rehman, H., Mao, Q., Zhang, S., Vucetic, S., Black, G., Jo, D., Suh, E., Dayton, J., Larsen, D., Omdahl, A., McGuffin, L., Brackenridge, D., Babbitt, P., Yunes, J., Fontana, P., Zhang, F., Zhu, S., You, R., Zhang, Z., Dai, S., Yao, S., Tian, W., Cao, R., Chandler, C., Amezola, M., Johnson, D., Chang, J., Liao, W., Liu, Y., Pascarelli, S., Frank, Y., Hoehndorf, R., Kulmanov, M., Boudellioua, I., Politano, G., Di Carlo, S., Benso, A., Hakala, K., Ginter, F., Mehryary, F., Kaewphan, S., Björne, J., Moen, H., Tolvanen, M., Salakoski, T., Kihara, D., Jain, A., Šmuc, T., Altenhoff, A., Ben-Hur, A., Rost, B., Brenner, S., Orengo, C., Jeffery, C., Bosco, G., Hogan, D., Martin, M., O’Donovan, C., Mooney, S., Greene, C., Radivojac, P., and Friedberg, I. (2019). The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biology*, 20(1).
- Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley, New York.

Original Publications

**Farrokh Mehryary, Suwisa Kaewphan, Kai Hakala, Filip
Ginter**
**Filtering large-scale event collections using a combination
of supervised and unsupervised learning for event trigger
classification**

Journal of Biomedical Semantics, 7(1), 2016, 1-13

RESEARCH

Open Access



Filtering large-scale event collections using a combination of supervised and unsupervised learning for event trigger classification

Farrokh Mehryary^{1,2*}, Suwisa Kaewphan^{1,2,3}, Kai Hakala^{1,2} and Filip Ginter¹

Abstract

Background: Biomedical event extraction is one of the key tasks in biomedical text mining, supporting various applications such as database curation and hypothesis generation. Several systems, some of which have been applied at a large scale, have been introduced to solve this task.

Past studies have shown that the identification of the phrases describing biological processes, also known as trigger detection, is a crucial part of event extraction, and notable overall performance gains can be obtained by solely focusing on this sub-task. In this paper we propose a novel approach for filtering falsely identified triggers from large-scale event databases, thus improving the quality of knowledge extraction.

Methods: Our method relies on state-of-the-art word embeddings, event statistics gathered from the whole biomedical literature, and both supervised and unsupervised machine learning techniques. We focus on EVEX, an event database covering the whole PubMed and PubMed Central Open Access literature containing more than 40 million extracted events. The top most frequent EVEX trigger words are hierarchically clustered, and the resulting cluster tree is pruned to identify words that can never act as triggers regardless of their context. For rarely occurring trigger words we introduce a supervised approach trained on the combination of trigger word classification produced by the unsupervised clustering method and manual annotation.

Results: The method is evaluated on the official test set of BioNLP Shared Task on Event Extraction. The evaluation shows that the method can be used to improve the performance of the state-of-the-art event extraction systems. This successful effort also translates into removing 1,338,075 of potentially incorrect events from EVEX, thus greatly improving the quality of the data. The method is not solely bound to the EVEX resource and can be thus used to improve the quality of any event extraction system or database.

Availability: The data and source code for this work are available at: <http://bionlp-www.utu.fi/trigger-clustering/>.

Keywords: BioNLP, Event extraction, Trigger detection, Word embeddings

*Correspondence: farmeh@utu.fi

¹Department of Information Technology, University of Turku, Turku, Finland

²The University of Turku Graduate School (UTUGS), University of Turku, Turku, Finland

Full list of author information is available at the end of the article



Background

The overwhelming amount of biomedical literature published annually makes it difficult for life science researchers to acquire and maintain a broad view of the field, crossing the boundaries of organism-centered research communities and gathering all of the information that would be relevant for their research. Modern natural language processing (NLP) techniques strive to assist the researchers with scanning the available literature and aggregating the information found within, automatically normalizing the variability of natural language statements. The applications of NLP in life sciences range from automated database curation and content visualization to hypothesis generation and offer intriguing challenges for both NLP and life science communities [1–3].

As a response to the need for advanced literature mining techniques for the biomedical domain, the BioNLP (Biomedical Natural Language Processing) community of researchers has emerged. The primary focus of the majority of research within the BioNLP community is to improve information retrieval (IR) and information extraction (IE) in the domain.

In this paper we focus on the task of *event extraction*, a task that has received much attention in BioNLP recently. Event extraction constitutes the identification of biological processes and interactions described in biomedical literature, and their representation as a set of recursive event structures. In its original form, introduced in the 2009 BioNLP Shared Task on Event Extraction (ST) [4], the task focused on gene and protein interactions, such as RNA transcription, regulatory control and post-translational modifications. In subsequent Shared Tasks, while the overall setting remained unchanged, the task has been broadened to cover a large number of additional biological domains and event types [5, 6].

More specifically, event extraction involves detecting mentions of the relevant named entities which are typically genes and gene products (GGPs), the type of their interaction from a small vocabulary of possible types, the *trigger* expression in the text which states the event, and the roles of the participants in the event, e.g. *regulator* or *regulatee*. One of the distinguishing features of events is that they can recursively act as participants of other events, forming recursive tree structures which precisely encode the factual statements in the text, but are a challenging extraction target. An example of an event is shown in Fig. 1.

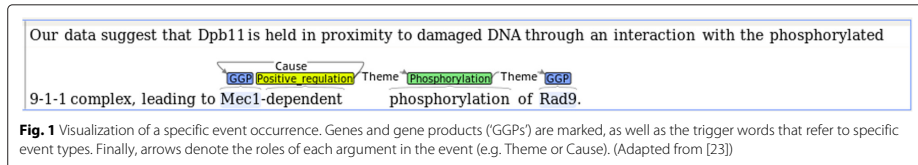
A number of event extraction systems have been introduced as the result of the series of BioNLP Shared Tasks. Most of these systems focus solely on the immediate textual context of the event candidates, but recently approaches benefiting from bibliome-wide data, either through self-training or post-processing steps, have been introduced as well [7–9]. Unfortunately the recent

advancements in this field have been modest, reflecting the complexity of the task. As an example, the best performing system in ST'09, TEES (Turku Event Extraction System) [10], has remained a state-of-the-art approach, winning also several categories in the later Shared Tasks, although the performance of the system has not increased substantially during these years.

Several event extraction systems have been applied at a large scale, extracting millions of events from massive text corpora [11, 12]. These large corpora, typically the totality of PubMed abstracts and PubMed Central full-text articles, contain a number of documents which are partly or entirely out-of-domain for these systems, being unlike the carefully selected data from narrow biological domains on which the systems have been trained. Facing documents from such previously unseen domains, the systems often produce suboptimal output, making what seems to a human like trivial mistakes. Tuning the performance of these systems in the general domain requires further effort.

Here we focus on EVEX [11], an event database covering the whole PubMed and PubMed Central Open Access (PMC-OA) literature, produced using the aforementioned TEES system. Already a casual inspection of EVEX reveals occasional occurrences of obviously incorrect events especially in out-of-domain documents. Previously, Van Landeghem et al. [13] have studied the output of the event extraction systems on general domain data in further detail. Their analysis resulted in a set of rules that can be used to remove or correct erroneous events. Although applying this method produced only an increase of 0.02pp in F-score when evaluated on the official Shared Task data, the consequences on large-scale resources such as EVEX are significant: hundreds of thousands of false events can be excluded, thus greatly improving the quality of the extracted data. This is because the official Shared Task test data does not contain the out-of-domain documents found in the corpora used to build EVEX and many of the error types made by the system will not be seen in the test set output.

Van Landeghem et al. point out that a large portion of the false event predictions originate from the trigger detection phase, i.e. false positive identification of the textual spans expressing the biological processes underlying the events. These, in turn, lead to the generation of false positive events by the system. Here it is important to take into consideration that the top-ranking event extraction systems are based on machine learning and do not uniquely rely on a list of possible “safe” trigger words, which would result in an excessively low recall. Instead, any word can become a trigger word, which occasionally leads to wildly incorrect predictions. These, in turn, are easily spotted by the users of the event databases and decrease the perceived credibility of the resources.



In this paper, we thus focus specifically on the event triggers in the EVEX event database, with the objective of automatically identifying and removing those that are obviously incorrect. To solve this problem, we introduce a novel approach based on word embeddings, bibliome-wide statistics and both supervised and unsupervised machine learning techniques.

Since our method relies on bibliome-wide statistics that should be gathered from a large-scale biomedical event database, it serves as a post-processing step in event extraction pipeline to filter out incorrect events from that database, after the events are extracted.

Method

In this section, we first introduce the data that is used in this study and then propose a 6-step method to achieve the aforementioned objectives.

In the first five steps of our method, we focus on the top most frequent trigger words which account for 97.1 % of all events in EVEX. In steps 1, 2, 3 and 4 we perform hierarchical clustering of these trigger words and build, analyze and prune the resulting binary tree to categorize these triggers as correct/incorrect. In step 5, we refine these two sets using manual annotation. Finally in step 6, we build a predictive model based on support vector machines (SVM) to classify the triggers as correct or incorrect.

Data

This study is based on the EVEX resource [11] containing 40,190,858 events of 24 different types such as *binding*, *positive-regulation*, *negative-regulation*, and *phosphorylation*. These events are extracted using the TEES system [14] from 6,392,824 PubMed abstracts and 383,808 PMC-OA full-text articles that were published up to 2012 and which contain at least one gene/gene-product mention. The EVEX resource can be downloaded and browsed online at www.evexdb.org.

Trained on the ST-data sets, TEES extracts events based on the recognition of an occurrence of a trigger word in the underlying sentence. An event is thus representing the link between the event trigger word and participating argument GGPs. However, one textual span can act as a trigger for multiple events with varying arguments as illustrated in Fig. 2.

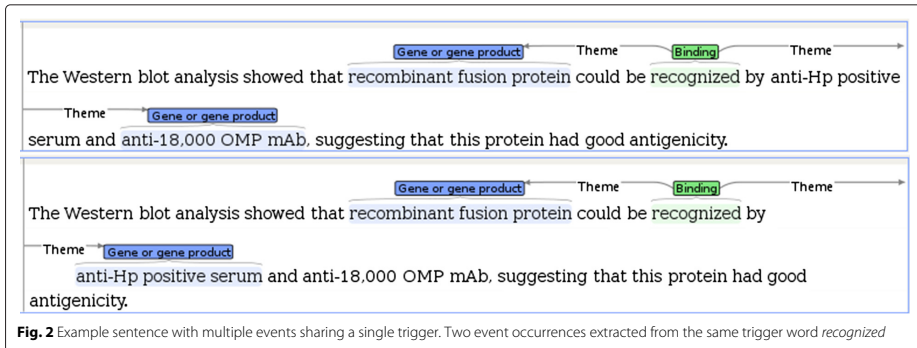
In addition, a single unique trigger word, such as *modify*, may have a number of occurrences in the data, acting as a trigger for many events. It is important to note that these events may be of different types. For instance the trigger word *expression* acts as a trigger for both *gene-expression* and *transcription* events, depending on the context.

Throughout this paper, we refer to the total number of extracted events from a trigger as “trigger frequency” and to the actual occurrence count of the trigger in the corpus as “trigger occurrence count”. Clearly, trigger frequency is greater or equal to trigger occurrence count since one trigger occurrence can be associated with multiple events. For example, the frequency of *expression* is 3,909,759, while its occurrence count is 2,736,782. It should be highlighted that the aim of this study is to increase the precision of extracted events, thus the focus is on the trigger frequency, i.e. the number of incorrect events that are finally removed from EVEX, when a particular trigger is identified as incorrect.

In total, there are 137,146 unique event triggers (excluding obviously incorrect trigger words that are purely numbers and those which contain unicode special characters). Different trigger words have different frequency in the system ranging from 1 to 3,909,759.

As expected, the vast majority of events in EVEX correspond to a small number of highly frequent trigger words, as shown in Table 1. For example, there are only 3,391 trigger words with frequency above 300 (i.e. corresponding to at least 300 event occurrences), but these words account for 97.1 % of all events in EVEX. Consequently, when the aim is to increase the precision of the events in EVEX by recognizing *incorrect* trigger words and eliminating them, the focus should be centered on highly frequent trigger words instead of the rare ones. Accordingly, we decided to concentrate on these 3,391 top most frequent trigger words. Limiting ourselves to the top most frequent trigger words allows manual inspection of the hierarchical clustering tree discussed in the following sections.

Among the trigger words, we will target those which are obviously incorrect, regardless of their context. These could be for example, gene/protein/chemical names, author names or any other words such as *hospital*, *university*, *research*, *diagram*, *box*, *clarify*, *investigate*, *visualization*, *knowledge*, *one* or *please*. The main objective of this study is thus to develop a method that can categorize the



trigger words so as to eliminate the obviously incorrect trigger words, thus increasing the precision of the event extraction systems without impacting their recall.

Another interesting aspect when studying the trigger words is to build a general overview of all of the trigger words according to the 24 different event types and to study whether there exist groups/sub-groups of related trigger words which would allow us to define subtypes of the 24 event types. Of specific interest will be studying the groups/sub-groups before and after eliminating incorrect trigger words.

Hierarchical clustering of top most frequent trigger words

In the first step, we induce a vector space representation for the trigger words, and hierarchically cluster the triggers based on this representation. *Cosine similarity* is used as the clustering metric with the *Ward's variance minimization algorithm* defining the distances between newly formed clusters. To build the vector space representations, we use the *word2vec* method of distributional semantics introduced by Mikolov et al. [15] and previously applied in the biomedical domain by Pyysalo et al. [16]. The *word2vec* method comprises a simplified neural network model with a linear projection layer and a hierarchical soft-max output prediction layer. The input

layer has the width of the vocabulary, while the projection layer has the desired dimensionality of the vector space representation. Upon training, the weight matrix between the input and the projection layer constitutes the word vector space embeddings. The network can be trained in several different regimes, but in this work we use the skip-gram architecture, whereby the network is trained to predict nearby context words, given a single focus word at the center of a sliding window context.

We train the *word2vec* model on the lower-cased texts from the EVEX resource, i.e. all abstracts and full-text articles in which at least one GGP was identified. All GGP mentions in the texts are replaced with the "ggp" placeholder and all numbers with the "num" placeholder to densify the text.

An initial experiment in hierarchical clustering of the top 100 most frequent trigger words revealed that on one hand many coarse/fine grained sub-clusters were formed in a way that each sub-cluster contained trigger words with biologically similar meaning. Many sub-clusters could be clearly associated with a unique event type. On the other hand, many trigger words were clustered together incorrectly, especially for the common *positive-regulation* and *negative-regulation* types (e.g. *increase* and *decrease*) because they have a high similarity in the vector space representation.

To address this issue, we add trigger/event type association information as additional dimensions to the word vectors, thereby affecting the clustering to more closely conform to the event types. To obtain reliable event type distribution for the trigger words, we use the BioNLP Shared Task 2011 (ST'11) *training* and *development* sets [5]. Out of the 1,447 unique trigger words in this data, 995 are single-token trigger words and of these, 828 are actually among the top 3,391 most frequent EVEX trigger words. For these 828 triggers, we append a normalized event type distribution vector to their *word2vec*-based

Table 1 Distribution of triggers and their associated event percentages in the EVEX database

| Trigger word frequency (at least) | EVEX events coverage percentage | Number of trigger words |
|-----------------------------------|---------------------------------|-------------------------|
| 100 | 98.4 | 6339 |
| 200 | 97.6 | 4263 |
| 300 | 97.1 | 3391 |
| 400 | 96.6 | 2880 |
| 500 | 96.3 | 2538 |

vectors (the vectors for the remaining 2,563 triggers for which a reliable event type information could not be obtained are simply padded with 24 zeroes). Re-clustering with the modified vectors, we notice that *positive-regulation* and *negative-regulation* trigger words are no longer clustered together, obtaining more meaningful clusters with regard to the task at hand.

Event type vectors of sub-clusters

In this step, event type distribution vectors for all nodes of the binary cluster tree are calculated. For each leaf of the tree (i.e., a trigger word), its corresponding trigger/event type vector is calculated based on the occurrence counts of its respective events in EVEX, and for each intermediate node of the tree (i.e., a sub-cluster), its respective event type vector is calculated by adding trigger/event type vectors of all triggers that belong to it.

Using this information, it is possible to inspect how the tree is organized and whether and how its different branches represent different event types. For example, by checking which element in a sub-cluster's event type vector has the maximum value, we can tell what is the event type that this sub-cluster is mostly associated with and the level of purity of that cluster. For example, while one sub-cluster can be 98% *binding* and is thus to a large extent *pure*, another cluster can be 43% *gene_expression* and cannot be assigned a single predominant type.

Identifying possibly incorrect trigger words

Focusing on 3,391 top most frequent trigger words, in this step we prepare a list of *safe* or *supposedly correct* trigger words and regard the remaining triggers as *possibly incorrect*. This is necessary for pruning the tree and finding the list of *incorrect* trigger words in the next step.

As stated in Section Hierarchical clustering of top most frequent trigger words, by analyzing the ST'11 training and development sets, we obtain a list of 995 unique single-token trigger words. Some of these triggers are overlapping with EVEX triggers. However, our list contains many other trigger words that can not be directly found in the ST'11 sets, but variations of them or variations of their parts can. For instance, *processing* and *co-regulation* are in the EVEX-based list, while *processed* and *regulation* are in the ST'11 sets.

We therefore process BioNLP ST'09 [4], ST'11 [5], and ST'13 [6], *training* and *development* sets, to obtain a set of all single-token ST-trigger words. This trigger set, hereafter *ST-set*, contains 1,092 trigger words. Then we perform the following preprocessing steps on every trigger word in both EVEX and *ST-set*.

1. Remove any punctuation or special characters from the beginning of the trigger word, retaining the rest of the word as the trigger word. For example, *-stimulated* is transformed into *stimulated*.

2. We split each trigger word based on occurrences of the following characters: {'-', ':', '_', '/'}. For example, *co-express* is divided into *co* and *express*, and similarly *cross-reacts* is divided into *cross* and *reacts*.
3. For every trigger word, each of its split parts is saved if *all* of the following conditions are met:
 - (1) The part is longer than one character.
 - (2) The part is not a number.
 - (3) The part is not in the following stop list: {32p, auto, beta, cis, co, cross, de, double, down, mono, non, out, poly, post, re, self, trans, under}. We obtained this list experimentally by careful examination of the *ST-set*.
4. Finally, we lemmatize all the trigger words, and all of their parts, using the *BioLemmatizer* tool [17] which is specifically developed for the biomedical domain, and record all the produced lemmas for each trigger word.

After the preprocessing, 977 EVEX trigger words that can directly be found in the *ST-set* are regarded as *safe*. The rest of the triggers are regarded as *safe* if their exact form, or one of their parts, or one of the lemmas of their parts can be found in the *ST-set*, or *ST-set* words' parts or part lemmas. Otherwise, the trigger word is regarded as *possibly incorrect*.

Performing the aforementioned approach resulted in identification of 506 trigger words which were added to the list of *safe* triggers, totaling to a list of 1,483 *safe* trigger words. The 1,908 remaining triggers are regarded as *possibly incorrect*. Table 2 shows some example words from EVEX triggers in our list that are matched against *ST-set* trigger words, parts, or lemmas.

As discussed earlier, we do not save parts of the EVEX trigger words if they belong to our stop list. The stop list comprises the prefix parts obtained by splitting *ST-set* trigger words, which are *not* themselves *ST-set* trigger words. For example, *cross-link* is a *ST-set* trigger word, but *cross* itself is not a stand-alone *ST-set* trigger, therefore

Table 2 Examples of matching EVEX trigger words against Shared Task *exact trigger words* or their corresponding *parts/lemmas*

| EVEX trigger word | ST'11-trigger word/Part/Lemma |
|-----------------------|-------------------------------|
| co-transcribed | transcribed |
| calcium-induced | induced |
| co-immunoprecipitates | immunoprecipitate |
| downregulating | downregulate |
| recognise | recognize |
| preceding | precede |
| analyzing | analyse |

cross is included in the stop list. As a contrary example, *up-regulation* is a *ST-set* trigger word, however we did not include *up* in the stop list because *up* itself is a *ST-set* trigger word. We perform such approach because we do not want any EVEX trigger word like *re-X* or *cross-X* (*X* can be any word) to be matched against *ST-set* words, parts and lemmas, just because it has a *re* or *cross* as a prefix.

Pruning the tree

Pruning the tree is done using the list of *possibly incorrect* trigger words in four steps:

1. If a trigger word exists in the list of *possibly incorrect* trigger words, its corresponding leaf is marked as *unsafe*, otherwise it is marked as *safe*.
2. If *all* of the children of an intermediate node are marked as *unsafe*, this node (sub-cluster as a whole) is marked as *unsafe* as well, otherwise it is marked as *safe*.
3. All of the descendants of the intermediate nodes that are marked as *unsafe*, are deleted from the tree. The respective trigger words of the deleted leaves are subsequently added to the list of *incorrect* trigger words.
4. After tree pruning, the trigger words of all leaves that remain in the tree, are marked as *safe* and regarded *correct*.

After applying the aforementioned tree pruning algorithm, we obtain a set of *correct* and a set of *incorrect* top most frequent EVEX trigger words.

There is one important aspect in the pruning algorithm. Since the tree is binary, not all of the trigger words that are in the list of *possibly incorrect* triggers were finally regarded as *incorrect* trigger words, because if such a trigger word was clustered near a *safe* trigger word (i.e., had a very small cosine distance to a *safe* trigger word in the feature space), it was not considered as an *incorrect* trigger word and remained in the tree. This helps us to identify more *correct* trigger words.

For example, *co-localization* which is an EVEX trigger word is also a Shared Task trigger word, so it had been marked as *safe* in the matching step, however *colocalization* (another EVEX trigger word) originally had been regarded as *possibly incorrect*, because our matching procedure could not have matched this trigger word (or its lemma) against any *ST-set* trigger word or part or lemma. However, because these two words are extremely similar in the vector space representation, they clustered together in the binary cluster tree. Consequently, since an *unsafe* trigger was clustered with a *safe* trigger, that whole sub-cluster was regarded as *safe* and remained in the tree, so *colocalization* finally is regarded as a *correct* trigger word. To summarize, the tree pruning algorithm causes deletions to be propagated to the upper level nodes of the

tree only if *all* of the participating leaves are recognized as *incorrect*.

After pruning, event type vectors for all intermediate nodes of the tree are recalculated so that we can compare the tree before and after pruning.

Refining correct and incorrect trigger sets

The output of the tree pruning step are the *correct* and *incorrect* trigger words sets, into which the top most frequent EVEX trigger words are assigned. As discussed in Section Results, our unsupervised method (steps 1-4) increases the precision and F-score of event extraction systems, however it causes a comparatively small drop of recall. This means that some of the *correct* trigger words are erroneously included in the *incorrect* trigger set, thus deleting their corresponding events from EVEX consequently decreases the recall of that event extraction system. As our objective is to increase the precision without decreasing the recall, i.e. we try to avoid removing correct events from EVEX at any cost, we address the issue using manual annotation to refine the results.

Manual annotation of triggers

A list of 3,391 trigger words was prepared by extracting the trigger words with *frequency* of at least 300 from EVEX. As discussed in Section Identifying *possibly incorrect* trigger words, 977 of EVEX top most frequent triggers overlap with the *ST-set*. We assume these triggers to be correct and provided the 2,414 remaining trigger words to an annotator with prior experience in biomedical domain annotation.

The annotator performed the manual annotation by deciding for each trigger whether it is *correct* or *incorrect*. On one hand, a trigger is *correct* if its occurrence can lead to the extraction of one or more of the 24 Shared Task event types, i.e. the given trigger word can represent at least one of the ST event types in some context, although in another context they might still be invalid. On the other hand, an *incorrect* trigger cannot express Shared Task events in any context. The annotator was allowed to use any available resources, such as NCBI [18], Gene Ontology [19] and KEGG [20] databases, to support the annotation.

The annotation of the top most frequent EVEX triggers resulted in three categories:

- 2083 triggers were annotated as *correct*.
- 577 triggers were annotated as *incorrect*.
- 731 triggers were not annotated and remained *undecided*.

In a closer look at the annotation data, the most common *correct* triggers are the words specifically used in biomedical domains such as “gene expression”, “regulation” and “transcription” to state the events. The

incorrect triggers are mostly biomedical entities such as genes, proteins and chemicals. While the majority of the triggers (2660/3391, 78.44%) can be annotated, the annotator was unable to make a decision for 21.56% of the triggers. Most of these *undecided* triggers are multiple-meaning words used in both biomedical and generic domains such as “conserved”, “deletion”, and “development”. Thus it is possible to construct hypothetical sentences where these words are valid triggers, but the annotator was not able to find any evidence supporting the use of these words as triggers from the existing literature. While going through all the sentences would be an ideal solution to resolve this issue, it is impossible in practice due to the vast amount of the data.

As this evaluation was conducted by a single annotator, we have not assessed the inter-annotator agreement (IAA) for this task. To our knowledge, the organizers of the BioNLP Shared Task have not reported the IAA for the GE data set either. For the EPI data set (Epigenetics and Post-translational Modifications) the organizers report agreement level of 82% measured in F-score [21]. This evaluation, however, measures the annotation of the full event structures and no direct conclusions can be made for the trigger annotations.

Aggregating unsupervised method results with manual annotation results

In this step, we aggregate the results from tree pruning (Section Pruning the tree) and manual annotation. We naturally prioritize the manual annotation, i.e., in the aggregated data a trigger remains *correct* or *incorrect* if labeled as such in the manual annotation. The *undecided* triggers are assigned using the tree pruning method. As a result, the *final* set is comprised of 2,242 *correct* triggers and 1,149 *incorrect* triggers.

Classification of low-frequency event triggers

In the previous steps, the focus was on assigning a label for top most frequent trigger words (those with frequency of at least 300) which account for 97.1% of all EVEX events. However, this demanding manual annotation method can not be applied to the huge number of triggers with lower frequency that exist in EVEX. To address this problem, we use support vector machines (SVM) to classify the low-frequency triggers (i.e., triggers with frequency below 300). As the training data, we use the aggregated trigger set from the previous section, assigning *correct* and *incorrect* triggers as *positive* and *negative* examples, respectively. Our training set totals 3,391 training examples, consisting of 2,242 positive and 1,149 negative examples. We optimize the model using grid-search combined with cross-validation.

Prior to building the model, we considered two important aspects which should be highlighted here. First, we

prefer a conservative predictive model which tends to have a very high positive recall, because if the classifier mispredicts a *correct* trigger as *incorrect*, all of its respective events mistakenly will be deleted from the output of event extraction systems which is very undesirable and that can also have a huge adverse effect on the recall of events. Conversely, if the classifier mistakenly predicts an *incorrect* trigger as *correct*, its respective events will remain in the output of event extraction systems, and in general we prefer to tolerate false events instead of deleting correctly extracted events. Because of this reason, and because our training set is imbalanced, we give weight 10 to the positive class and weight 1 to the negative class during classifier training. These weights are set experimentally during the grid search and classifier optimization. In addition, during optimization, instead of optimizing against F1-score we optimize against F2-score, because F2-score weights recall higher than precision.

Second, from the point of view of event extraction systems, the respective events of the triggers are more important than the trigger words themselves. For instance, misclassifying a correct trigger with frequency of 200 will translate into removing 200 correct events, comparing it with the removal of a correct trigger with frequency of only 1. Consequently, we consider the precision and recall of respective events (not the triggers) and adjust the parameter optimization and training accordingly:

- During optimization, instead of optimizing against the F2-score of triggers, i.e., calculating F2-score based on the counts of true-positives (TP), true-negatives (TN), false-positives (FP) and false-negatives (FN), we optimize against F2-score of trigger frequency, i.e., calculating F2-score based on the sum of frequencies of TP, TN, FP and FN.
- We give a weight to each training example by calculating the logarithm of its frequency. Thus the training examples with higher weights, i.e. higher event frequency, will be regarded more important than lower weight examples, those with lower event frequency. In other words, classifier will be penalized more on misclassifying the frequent trigger words than lesser ones during training and k-fold cross-validations. As a result, the classifier is trained towards better performance on more frequent triggers while we intentionally do not give the trigger frequency as a feature to the classifier.

Below is the set of features used by the classifier.

1. *word2vec*-based vector for each trigger, which is exactly the same vector discussed in Section Hierarchical clustering of top most frequent trigger words.

2. If the trigger word (or its lemma or its parts or lemmas of its parts) can be matched against *ST-set* words/parts/lemmas (according to Section Identifying possibly incorrect trigger words) this feature is 1, otherwise it is zero.
3. The value for this feature is calculated as following: $feature_value = occurrence\ count\ of\ trigger / X$ where X is the sum of occurrences of the word in all PubMed abstracts and PMC-OA full text articles published up to 2013, regardless of being recognized as a trigger word in the underlying sentences or not. For many incorrect trigger words, this feature will have a very low value. For example, for an incorrect trigger word like *hospital* which is in EVEX (not in training set), the value will be $(928 / 1,266,408) = 0.0007$.
4. For this feature, we first extract the set of single-token triggers with length of more than 6 characters in the *ST-set*, introduced in Section Identifying possibly incorrect trigger words. Then, for each training example we calculate the feature value as following: $feature_value = length(trigger\ word) / (Y + 1)$ where Y is the minimum edit distance (Levenshtein distance) of the trigger word to the words in the previously created set. For all training examples that belong to the *ST-set*, we assume Y to be zero. The longer the trigger word, and the smaller its minimum edit distance, the higher will be the value of this feature. This feature is beneficial for example in the case of a misspelled trigger (e.g., *phosphorylation* instead of *phosphorylation*), which is not recognized correctly by our matching protocol discussed in Section Identifying possibly incorrect trigger words.
5. Number of alphabetic characters divided by the length of the trigger word.

We perform a grid-search combined with 5-fold cross-validation to optimize the classifier and find the best hyper-parameters for the model (*kernel type*, *C value*, and the *gamma parameter* for RBF-kernel) against the F2-score of trigger event frequency. Subsequently, we train the classifier using the best parameter values on all available training examples.

Results

In this section, we discuss the results in four parts. First, in Section Evaluation of event filtering, we evaluate the impact of trigger pruning on event extraction systems. We then evaluate our predictive model and investigate the effect of event filtering on the EVEX resource in Sections Evaluation of low-frequency trigger classification and Evaluation of event removal on the EVEX resource. Finally, in Section Tree organization before/after pruning

we examine the trigger cluster tree organization before and after the pruning.

Evaluation of event filtering

Evaluation method

We evaluate the impact of trigger pruning on event extraction using the official test sets of the BioNLP ST'11 and GENIA Event Extraction (GE) Shared Tasks (ST'13). As the basis we consider the outputs of the *TEES system* entry [10, 14] in 2011 (3rd place) and in 2013 (2nd place) GE tasks and, for the 2013 Shared Task, also the winning *EVEX* entry [7]. We prune the outputs of these systems by removing events whose trigger words are identified as incorrect using the aforementioned algorithm and evaluate the resulting pruned set of events using the official evaluation services of the respective Shared Task on the held-out test sets. The results are shown in Table 3.

It should be highlighted that naturally the magnitude of the F-score improvements is modest, as the top-ranking systems are well optimized and major improvements have been difficult to achieve regardless of the approach. Note also that a filtering approach such as the one proposed in this paper cannot increase recall because it is unable to produce new events. Our main focus thus is on *improving the precision while trying to retain the recall*, aiming to increase the credibility of large-scale event extraction systems in general.

Evaluation of unsupervised method

In this section, we investigate the effect of removing triggers from event extraction systems using the set of *incorrect* trigger words obtained from our unsupervised method in Section Pruning the tree.

As shown in Table 3, in all three instances (comparing our *unsupervised method* against the *TEES* system's predictions on tasks 2011 and 2013, and the *EVEX* system's predictions on task 2013), we see an improvement in both precision and F-score with a relatively small drop in recall. Especially for the ST'13, the pruned *TEES* system (+0.23pp F-score over *TEES*) matches in performance with the winning 2013 *EVEX* system. Since the *EVEX* system was also based on *TEES*, it is interesting to note that we have matched these improvements using a different approach. Finally, the pruned *EVEX* system (+0.18pp F-score over the *EVEX* entry) establishes a new top score on the task.

Evaluation of manual annotation method

In this section we investigate the effects on event extraction if we rely our method solely on the manual annotation results. We remove events from those three aforementioned event extraction system outputs, using the set of trigger words that were annotated as *incorrect* by the human annotator.

Table 3 Performance comparison of the different pruning approaches and the baseline methods (TEES/EVEX) on the official BioNLP Shared Task GE data sets

| | Predictions | Precision | Recall | F1-score |
|------------------------------|--|-----------|--------|----------|
| TEES-2011 (Shared Task 2011) | Original TEES | 61.76 | 48.78 | 54.51 |
| | Pruned-TEES (Unsupervised Method) | 62.39 | 48.75 | 54.74 |
| | Pruned-TEES (Manual Annotation Method) | 62.04 | 48.78 | 54.62 |
| | Pruned-TEES (Aggregation Method) | 62.26 | 48.78 | 54.70 |
| | Pruned-TEES (Aggregation Method + SVM) | 62.27 | 48.78 | 54.71 |
| TEES-2013 (Shared Task 2013) | Original TEES | 56.32 | 46.17 | 50.74 |
| | Pruned-TEES (Unsupervised Method) | 57.13 | 46.02 | 50.97 |
| | Pruned-TEES (Manual Annotation Method) | 56.63 | 46.17 | 50.87 |
| | Pruned-TEES (Aggregation Method) | 56.97 | 46.17 | 51.00 |
| | Pruned-TEES (Aggregation Method + SVM) | 57.01 | 46.17 | 51.02 |
| EVEX-2013 (Shared Task 2013) | Original EVEX | 58.03 | 45.44 | 50.97 |
| | Pruned-EVEX (Unsupervised Method) | 58.77 | 45.29 | 51.15 |
| | Pruned-EVEX (Manual Annotation Method) | 58.32 | 45.44 | 51.08 |
| | Pruned-EVEX (Aggregation Method) | 58.66 | 45.44 | 51.21 |
| | Pruned-EVEX (Aggregation Method + SVM) | 58.71 | 45.44 | 51.23 |

As shown in Table 3, in all three instances (comparing *manual annotation method* against the *TEES* system's predictions on tasks 2011 and 2013, and the *EVEX* system's predictions on task 2013), manual annotation retains the recall, which is obviously a better result than our unsupervised method. However in all three instances, its precision and F-score is less than the precision and F-score of our unsupervised method.

The preserved recall suggest that our annotation strongly agrees with the ST annotation guidelines. However, the higher precision of the unsupervised pruning strategy shows that some cases not clear for a human annotator, can be classified with this method.

This is exactly what we had anticipated. As precise annotation was not possible for many trigger words, we have 731 *undecided* top most frequent triggers, and many *incorrect* trigger words might actually be among them.

To summarize, the manual annotation has produced an *almost pure but incomplete* set of *incorrect* trigger words. In comparison to original event extraction system performances, our manual annotation method does increase the precision and F-score while retaining the recall, but its precision and F-score are not as high as our unsupervised method.

Evaluation of aggregation method

As shown in the previous sections, our unsupervised method increases the precision and F-score, but slightly drops the recall, whereas the manual annotation alone retains recall with lesser increase in precision. In this

section, we investigate the effect of event filtering using the set of *incorrect* triggers obtained from the aggregation method discussed in Section Aggregating unsupervised method results with manual annotation results.

As shown in Table 3, in comparison with the *TEES* performance on ST'11 and ST'13, and the *EVEX* performance on ST'13, the aggregation method retains the recall and increases the precision and F-score. Interestingly, in all three cases, in comparison with manual annotation method it has a higher precision and F-score. Consequently, we conclude that our unsupervised method is indeed able to find *incorrect* trigger words elusive to the human annotator.

If we compare the aggregation method performance with our unsupervised method performance, we notice that in all three instances, it does have a higher recall and in two cases also higher F-score. In one case the unsupervised method alone reaches the highest F-score. This might be due to trigger words that we have annotated as *correct*, but are used in wrong event types by the underlying event extraction system, thus resulting in lower precision.

As a conclusion, while all of our methods establish new top scores on 2013 tasks, the aggregation method is the best among them. It retains the recall, increases the precision and has the best F-score in two cases out of three.

Evaluation of low-frequency trigger classification

As stated in Section Classification of low-frequency event triggers, we use all 3,391 top most EVEX frequent triggers

to train the classifier and aim to apply it on those triggers with *frequency* below 300.

Similar to the previous evaluations, we first evaluate the classifier performance against the Shared Task test sets as an end-to-end system together with the aggregation method. For this aim, we apply the trained classifier to predict labels for EVEX triggers with frequency below 300. This results in identification of 16,674 negative (supposed to be incorrect) triggers with total frequency of 232,748 respective events in EVEX. The rest of the triggers were predicted as correct. Then, we prune the output of event extraction systems using these recognized incorrect triggers and incorrect triggers obtained by aggregation method.

Results for this experiment are shown in Table 3. Comparison of this method (*Aggregation Method + SVM* entries in the table) against our aggregation method, the previously best approach, shows slight increase in both precision and F-score in all three cases while retaining the same recall. Thus, the classifier is able to recognize some previously undetected *incorrect* trigger words, giving us the most complete set of *incorrect* trigger words.

As this processing step focuses specifically on low-frequency (rare) triggers, unlikely to be found in the carefully selected Shared Task data sets, the performance improvement is small, as anticipated. However, we expect the outcome to be more significant in large-scale event extraction and to show this we conduct another evaluation based on the EVEX resource.

In the second evaluation we form an evaluation set by randomly selecting 700 words from the triggers with *frequency* less than 300 in EVEX and use the same manual annotation procedure discussed in Section Manual annotation of triggers to divide them into positive (*correct*) and negative (*incorrect*) sets.

The annotation resulted in 363 *correct* and 233 *incorrect* triggers. For 104 triggers our annotator was unable to assign a label. Even though, in terms of our annotation protocol, the triggers are divided into three independent classes, for simplicity we exclude the 104 *undecidable* trigger words from our test set and use only the 596 remaining words.

The performance evaluation results against the test set are shown in three different tables.

- Table 4 shows the counts and respective event frequencies of true-positives, true-negatives, false-positives and false-negatives.
- Table 5 shows the performance in terms of classification of triggers. Precision, recall and F2-score in this table are calculated based on the counts of the predicted triggers.
- Table 6 shows the performance in terms of classification of events. Precision, recall and F2-score

Table 4 Trigger/event classification performance, measured on the EVEX test set: The first column (Count) shows prediction results based on the counts of trigger words (test set examples). The second column (Sum of frequency) shows the number of respective events of those triggers in the EVEX database. For instance, the first row (True-Positive) shows that the classifier has correctly predicted 352 test set trigger words to be *correct triggers*, while these words account for 4,602 extracted events in the EVEX resource

| | Count | Sum of frequency (Number of events) |
|----------------|-------|--|
| True-Positive | 352 | 4602 |
| True-Negative | 99 | 679 |
| False-Positive | 134 | 850 |
| False-Negative | 11 | 115 |
| Total | 596 | 6246 |

in this table are calculated based on the event frequencies of the predicted triggers (i.e., based on the sum of frequencies of TP, TN, FP and FN).

As mentioned in Section Classification of low-frequency event triggers, from the event extraction point of view, the event frequencies are more important than the unique trigger words themselves. Thus, results listed in Table 6 are the most relevant for examining the performance of the classifier. As this is also the evaluation metric the classifier hyper-parameters were optimized against, the numbers in Table 6 are generally higher than in Table 5.

We can see that the classifier achieves recall of 0.98 for the positive class, i.e. the *correct* triggers as shown in Table 6. This result suggests that we have succeeded in our goal of preserving as much of the true events as possible. Besides, the classifier also reaches recall of 0.44 for the *incorrect* triggers, i.e. we are able to detect and exclude almost half of the events with false triggers in this evaluation set.

Table 5 Trigger classification performance on the EVEX resource based on trigger counts (test set examples). The prediction measures in this table are calculated based on the values in the first column of Table 4. This table shows how well the classifier is able to classify and distinguish between *correct* and *incorrect* trigger words. The last column (Support) shows that there are 363 *correct* and 233 *incorrect* trigger words in the test set, i.e. 596 in total

| | Precision | Recall | F2-score | Support |
|--------------------------|-----------|--------|----------|---------|
| Negative (incorrect) | 0.90 | 0.42 | 0.48 | 233 |
| Positive (correct) | 0.72 | 0.97 | 0.91 | 363 |
| Weighted averages, total | 0.79 | 0.76 | 0.74 | 596 |

Table 6 Classification performance on the EVEX resource based on the respective event counts in the EVEX database. This table shows how well the classifier will perform the prediction, preserving *correct* and eliminating *incorrect* respective events from the EVEX database. The prediction measures in this table are calculated based on the values in the second column of Table 4. The last column (Support) shows that there are 1,529 *incorrect* and 4,717 *correct* corresponding events in the EVEX database (6,246 in total) which are extracted based on those 596 trigger words in the test set

| | Precision | Recall | F2-score | Support |
|--------------------------|-----------|--------|----------|---------|
| Negative (incorrect) | 0.86 | 0.44 | 0.49 | 1529 |
| Positive (correct) | 0.84 | 0.98 | 0.95 | 4717 |
| Weighted averages, total | 0.85 | 0.77 | 0.77 | 6246 |

Evaluation of event removal on the EVEX resource

In this section we investigate the impact of removing events from the EVEX resource based on all trigger words recognized as *incorrect*.

Even though our manual annotation or aggregation methods are able to preserve the recall when evaluated against official predictions of Shared Task test sets, it is not guaranteed that the same performance will be achieved when applying them on a large-scale resource such as EVEX. In fact there might be *correct* triggers which are not present in ST'11 or ST'13 test sets, but are mistakenly labeled as *incorrect* by the human annotator, our unsupervised method or the classifier. Consequently, in the evaluation against official Shared Task test sets, we do not delete these triggers and do not detect any drop in recall. However based on our evaluation results, we are optimistic that most of the correct events will be preserved if the method is applied on the EVEX resource.

To investigate the impact of event removal on EVEX, for top most frequent triggers (accounting for 97.1% of all EVEX events), we rely on our aggregation method which had the best performance. The aggregation method resulted in labeling 1,149 triggers as *incorrect* and these account for 1,105,327 events in EVEX.

For the rest of EVEX triggers (*low frequency* triggers accounting for 2.9% of all EVEX events), we use the classifier. However, the classifier could not be applied to 48,960 triggers with 122,344 respective events (0.3% of all EVEX events). These words have less than 5 occurrences in the corpus used for training the *word2vec* model, and thus do not have a corresponding vector representation, required by the classifier. Applying the classifier on the rest of low frequency triggers (accounting for 2.6% of all EVEX events) resulted in identification of 16,674 *incorrect* triggers with 232,748 events in EVEX.

Consequently, in total we have been able to identify 17,823 expected to be *incorrect* triggers in the whole EVEX resource with 1,338,075 events which constitutes 3.3% of all events in EVEX.

Tree organization before/after pruning

In this section we address two questions. First, how the resulting binary cluster tree differs before and after the pruning, and second, whether we can define new event subtypes based on the organization of sub-clusters in different branches of the tree. For these aims, we visualize the tree before and after pruning up to the depth of 9 using the *Dendroscope* software [22]. We label every intermediate node of the tree with its *mostly associated event type* and the *level of purity* of that sub-cluster (see Additional file 1 for diagrams of the tree).

As expected, in both trees we notice that trigger words of same event types are clustered together, to some extent. By considering the length of the shortest path in tree as a basic distant measure, we observe that sub-clusters of similar/related event types are closer in the tree, while sub-clusters of different event types are located far. For instance, triggers for expressing different types of post-translational modifications events (e.g., "phosphorylation", "DNA-methylation", "glycosylation", "acetylation") are clustered together, far from trigger words for expressing "positive/negative regulation" or "binding". Similarly, sub-clusters of "gene-expression", "transcription" and "localization" trigger event types are close in the tree. We observe that before pruning the tree, sub-clusters are not pure. For example, many trigger words for "positive regulation" events are often clustered together with the ones for "negative regulation" events. By removing the sub-clusters of purely *incorrect* triggers, i.e. pruning, the sub-clusters in the middle levels of the tree become purer and are for the most part, associated with the same event type which signifies the possibility of identifying some of the event subtypes.

We thus continue the analysis on the associated events in the sub-tree anticipating to recognize the patterns. However, to our surprise, there is no clear signal in the sub-clusters that would signify any of the subtypes. As a result, we thus do not pursue further analysis on the trees. To conclude, our pruning algorithm yields a meaningful tree which can distinguish different event types into sub-clusters, however, the resulting clusters could not be used to identify event subtypes.

Conclusions

In this paper, we propose a method which can be used for identification of *incorrect* trigger words and removing *incorrect* events from the output of large-scale event extraction systems.

Our unsupervised method achieves a modest improvement over the winning system of the BioNLP 2013 Shared Task on GENIA event extraction and establishes a new top score on the task. The aggregation of manual annotation results with our unsupervised method results, further increases the precision and F-score of the unsupervised

method. Besides, the unsupervised method decreases the original recall when evaluated against official predictions of Shared Task test sets, while our aggregation method retains it.

Because the highly demanding manual annotation is not possible for all EVEX trigger words, we build a SVM classifier for predicting *incorrect* triggers among low-frequency EVEX triggers. While having 0.98 positive recall which translates to preserving a huge proportion of correct events, the classifier has 0.44 negative recall, meaning that it is able to identify about half of the *incorrect* events.

Combining the results of our aggregation method with *incorrect* trigger words identified by applying the classifier on all low frequency EVEX triggers, results in recognition of 17,823 expected to be incorrect triggers with 1,338,075 respective events which constitutes about 3.3% of all events in EVEX resource.

In this paper we have only discussed the identification of the incorrect triggers and the outcome of removing these triggers from a large-scale event resource. Although our evaluation shows only minimal drop in recall, bluntly removing the corresponding events might have unwanted effects. As the EVEX resource ranks the events shown to the users based on a scoring system derived from the TEES classification confidence, we would thus as a future work like to investigate how to incorporate these new findings in the ranking. This would let us, instead of completely abolishing the likely incorrect events, only to decrease their scoring and conserve them for those use cases that demand extremely high recall, but can overcome the noise in the data.

Another direction is to investigate the different event types in more detail. We hope this study will give us a better insight of whether the method can be adapted to also correct mistyped events, thus increasing the precision even further. For instance, it is possible that a detected *regulation* trigger should in fact be classified as *positive-regulation*, a subtype of *regulation*, but the used trigger detector has not been able to make this distinction. By observing how the given trigger word is located in the hierarchical cluster tree, these errors could be possibly corrected.

As the distributional semantics research is progressing towards better representations of phrases and larger text sections in addition to word-level embeddings, it might be possible in the future to instead of judging the trigger words globally, to focus only on certain types of contexts giving us the ability to make more precise decisions.

Availability of supporting data

The source code and data sets supporting the results of this article are available at the Turku BioNLP group website at: <http://bionlp-www.utu.fi/trigger-clustering/>.

Additional file

Additional file 1: This tar file contains images of the binary cluster tree, before and after the pruning. The HowToInterpretTreeDiagrams.txt file describes how the diagrams should be interpreted. (TAR 1290 kb)

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

The method is designed, fine-tuned, and implemented by FM, with input from all authors. SK did all manual annotations, while KH prepared an annotation framework. All authors equally contributed in analyzing the results and drafting the manuscript. The work is carried out under the supervision of FG. All authors read and approved the final manuscript.

Acknowledgments

We would like to thank Sofie Van Landeghem, Ghent University, for initiating the ideas for this project and her valuable suggestions. We also kindly thank Jari Björne, University of Turku, for his help during the project.

Author details

¹Department of Information Technology, University of Turku, Turku, Finland. ²The University of Turku Graduate School (UTUGS), University of Turku, Turku, Finland. ³Turku Centre for Computer Science (TUCS), Turku, Finland.

Received: 28 February 2015 Accepted: 1 May 2016

Published online: 11 May 2016

References

- Wei CH, Kao HY, Lu Z. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Res.* 2013;41:W518,W522.
- Szklarczyk D, Franceschini A, Wyder S, Forslund K, Heller D, Huerta-Cepas J, Simonovic M, Roth A, Santos A, Tsafou KP, Kuhn M, Bork P, Jensen LJ, von Mering C. STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res.* 2015;43(D1):447–52.
- Hakala K, Mehryary F, Kaewphan S, Ginter F. Hypothesis generation in large-scale event networks. In: Proceedings of the 5th International Symposium on Languages in Biology and Medicine (LBM'13). Tokyo: Database Center for Life Science; 2013.
- Kim JD, Ohta T, Pyysalo S, Kano Y, Tsujii J. Overview of BioNLP'09 Shared Task on Event Extraction. In: Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task. Boulder, Colorado: Association for Computational Linguistics; 2009. p. 1–9.
- Kim JD, Pyysalo S, Ohta T, Bossy R, Nguyen N, Tsujii J. Overview of BioNLP Shared Task 2011. In: Proceedings of the BioNLP Shared Task 2011 Workshop. Portland, Oregon, USA: Association for Computational Linguistics; 2011. p. 1–6.
- Nédellec C, Bossy R, Kim JD, Kim J-j, Ohta T, Pyysalo S, Zweigenbaum P. Overview of BioNLP Shared Task 2013. In: Proceedings of the BioNLP Shared Task 2013 Workshop. Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 1–7.
- Hakala K, Van Landeghem S, Salakoski T, Van de Peer Y, Ginter F. EVEX in ST'13: Application of a large-scale text mining resource to event extraction and network construction. In: Proceedings of the BioNLP Shared Task 2013 Workshop (BioNLP-ST'13). Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 26–34.
- Björne J, Ginter F, Salakoski T. University of Turku in the BioNLP'11 Shared Task. *BMC Bioinformatics.* 2012;13(Suppl 11):4.
- MacKinlay A, Martinez D, Jimeno Yepes A, Liu H, Wilbur WJ, Verspoor K. Extracting biomedical events and modifications using subgraph matching with noisy training data. In: Proceedings of the BioNLP Shared Task 2013 Workshop. Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 35–44.
- Björne J, Salakoski T. TEES 2.1: Automated annotation scheme learning in the BioNLP 2013 Shared Task. In: Proceedings of the BioNLP Shared Task 2013 Workshop. Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 16–25.

11. Van Landeghem S, Björne J, Wei CH, Hakala K, Pyysalo S, Ananiadou S, Kao HY, Lu Z, Salakoski T, Van de Peer Y, Ginter F. Large-scale event extraction from literature with multi-level gene normalization. *PLoS ONE*. 2013;8(4):e55814.
12. Gerner M, Sarafraz F, Bergman CM, Nenadic G. BioContext: an integrated text mining system for large-scale extraction and contextualization of biomolecular events. *Bioinformatics*. 2012;28(16):2154–61.
13. Van Landeghem S, Kaewphan S, Ginter F, Van de Peer Y. Evaluating large-scale text mining applications beyond the traditional numeric performance measures. In: *Proceedings of the 2013 Workshop on Biomedical Natural Language Processing (BioNLP'13)*. Sofia, Bulgaria: Association for Computational Linguistics; 2013. p. 63–71.
14. Björne J, Ginter F, Salakoski T. University of Turku in the BioNLP'11 Shared Task. *BMC Bioinformatics*. 2012;13(Suppl 11):4.
15. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. 2013;abs/1301.3781:1–12. *CoRR abs/1301.3781*.
16. Pyysalo S, Ginter F, Moen H, Salakoski T, Ananiadou S. Distributional semantics resources for biomedical text processing. In: *Proceedings of the 5th International Symposium on Languages in Biology and Medicine (LBM 2013)*. Tokyo: Database Center for Life Science; 2013. p. 39–43.
17. Liu H, Christiansen T, Jr WAB, Verspoor K. BioLemmatizer: a lemmatization tool for morphological processing of biomedical text. *J Biomed Semant*. 2012;3(3):1–29.
18. Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, et al. Database resources of the national center for biotechnology information. *Nucleic Acids Res*. 2007;35(suppl 1):5–12.
19. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, et al. Gene Ontology: tool for the unification of biology. *Nat Genet*. 2000;25(1):25–9.
20. Kanehisa M, Goto S. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*. 2000;28(1):27–30.
21. Pyysalo S, Ohta T, Rak R, Sullivan D, Mao C, Wang C, Sobral B, Tsujii J, Ananiadou S. Overview of the ID, EPI and REL tasks of BioNLP Shared Task 2011. *BMC Bioinformatics*. 2012;13(Suppl 11):2.
22. Huson DH, Scornavacca C. Dendroscope 3: An interactive tool for rooted phylogenetic trees and networks. *Syst Biol*. 2012;61:1061–1067.
23. Van Landeghem S, Hakala K, Rönqvist S, Salakoski T, Van de Peer Y, Ginter F. Exploring biomolecular literature with evex: Connecting genes through events, homology, and indirect associations. *Adv Bioinformatics*. 2012;2012:1–12.

Submit your next manuscript to BioMed Central
and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at
www.biomedcentral.com/submit



**Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio
Salakoski, Filip Ginter**
**Deep Learning with Minimal Training Data: TurkuNLP Entry
in the BioNLP Shared Task 2016**

Proceedings of the 4th BioNLP Shared Task Workshop. Association for
Computational Linguistics, Berlin, Germany,
2016, 73-81

Deep Learning with Minimal Training Data: TurkuNLP Entry in the BioNLP Shared Task 2016

Farrokh Mehryary^{1,3}, Jari Björne^{2,3}, Sampo Pyysalo⁴, Tapio Salakoski^{2,3} and Filip Ginter^{2,3}

¹University of Turku Graduate School (UTUGS)

²Turku Centre for Computer Science (TUCS)

³Department of Information Technology, University of Turku
Faculty of Mathematics and Natural Sciences, FI-20014, Turku, Finland

⁴Language Technology Lab, DTAL, University of Cambridge
firstname.lastname@utu.fi, sampo@pyysalo.net

Abstract

We present the TurkuNLP entry to the BioNLP Shared Task 2016 Bacteria Biotopes event extraction (BB3-event) subtask. We propose a deep learning-based approach to event extraction using a combination of several Long Short-Term Memory (LSTM) networks over syntactic dependency graphs. Features for the proposed neural network are generated based on the shortest path connecting the two candidate entities in the dependency graph. We further detail how this network can be efficiently trained to have good generalization performance even when only a very limited number of training examples are available and part-of-speech (POS) and dependency type feature representations must be learned from scratch. Our method ranked second among the entries to the shared task, achieving an F-score of 52.1% with 62.3% precision and 44.8% recall.

1 Introduction

The BioNLP Shared Task 2016 was the fourth in the series to focus on event extraction, an information extraction task targeting structured associations of biomedical entities (Kim et al., 2009; Ananiadou et al., 2010). The 2016 task was also the third to include a Bacteria Biotopes (BB) subtask focusing on microorganisms and their habitats (Bossy et al., 2011). Here, we present the TurkuNLP entry to the BioNLP Shared Task 2016 Bacteria Biotope event extraction (BB3-event) subtask. Our approach builds on proven tools and ideas from previous tasks and is novel in its application of deep learning methods to biomedical event extraction.

The BB task was first organized in 2011, then consisting of named entity recognition (NER) targeting mentions of bacteria and locations, followed by the detection of two types of relations involving these entities (Bossy et al., 2011). Three teams participated in this task, with the best F-score of 45% achieved by the INRA Bibliome group with the Alvis system, which used dictionary mapping, ontology inference and semantic analysis for NER, and co-occurrence-based rules for detecting relations between the entities (Ratkovic et al., 2011). The 2013 BB task defined three subtasks (Nédellec et al., 2013), the first one concerning NER, targeting bacteria habitat entities and their normalization, and the other two subtasks involving relation extraction, the task targeted also by the system presented here. Similarly to the current BB3-event subtask, the 2013 subtask 2 concerned only relation extraction, and subtask 3 extended this with NER. Four teams participated in these tasks, with the UTurku TEES system achieving the first places with F-scores of 42% and 14% (Björne and Salakoski, 2013).

We next present the 2016 BB3-event subtask and its data and then proceed to detail our method, its results and analysis. We conclude with a discussion of considered alternative approaches and future work.

2 Task and Data

In this section, we briefly present the BB3-event task and the statistics of the data that has been used for method development and optimization, as well as for test set prediction.

Although the BioNLP Shared Task has introduced an event representation that can capture associations of arbitrary numbers of participants in complex, recursive relationships, the BB3-event task follows previous BB series subtasks in ex-

| | Train | Devel | Test |
|------------------------|-------|-------|------|
| Total sentences | 527 | 319 | 508 |
| Sentences w/examples | 158 | 117 | 158 |
| Sentences w/o examples | 369 | 202 | 350 |
| Total examples | 524 | 506 | 534 |
| Positive examples | 251 | 177 | - |
| Negative examples | 273 | 329 | - |

Table 1: BB3-event data statistics. (The relation annotations of the test set have not been released.)

clusively marking directed binary associations of exactly two entities. For the purposes of machine learning, we thus cast the BB3-event task as binary classification taking either a (BACTERIA, HABITAT) or a (BACTERIA, GEOGRAPHICAL) entity pair as input and predicting whether or not a *Lives-in* relation holds between the BACTERIA and the location (HABITAT or GEOGRAPHICAL).

Our approach builds on the shortest dependency path between each pair of entities. However, while dependency parse graphs connect words to others in the same sentence, a number of annotated relations in the data involve entities appearing in different sentences, where no connecting path exists. Such cross-sentence associations are known to represent particular challenges for event extraction systems, which rarely attempt their extraction (Kim et al., 2011). In this work, we simply exclude cross-sentence examples from the data. This elimination procedure resulted in the removal of 106 annotated relations from the training set and 62 annotated relations from the development set.

The examples that we use for the training, optimization and development evaluation of our method are thus a subset of those in the original data.¹ When discussing the training, development and test data, we refer to these filtered sets throughout this manuscript. The statistics of the task data after this elimination procedure are summarized in Table 1. Note that since there are various ways of converting the shared task annotations into examples for classification, the numbers we report here may differ from those reported by other participating teams.

¹Official evaluation results on the test data are of course comparable to those of other systems: any cross-sentence relations in the test data count against our submission as false negatives.

3 Method

We next present our method in detail. Preprocessing is first discussed in Section 3.1. Section 3.2 then explains how the shortest dependency path is used, and the architecture of the proposed deep neural network is presented in Section 3.3. Section 3.4 defines the classification features and embeddings for this network. Finally, in Section 3.5 we discuss the training and regularization of the network.

3.1 Preprocessing

We use the TEES system, previously developed by members of the TurkuNLP group (Björne and Salakoski, 2013), to run a basic preprocessing pipeline of tokenization, POS tagging, and parsing, as well as to remove cross-sentence relations. Like our approach, TEES targets the extraction of associations between entities that occur in the same sentence. To support this functionality, it can detect and eliminate relations that cross sentence boundaries in its input. We use this feature of TEES as an initial preprocessing step to remove such relations from the data.

To obtain tokens, POS tags and parse graphs, TEES uses the BLLIP parser (Charniak and Johnson, 2005) with the biomedical domain model created by McClosky (2010). The phrase structure trees produced by the parser are further processed with the Stanford conversion tool (de Marneffe et al., 2006) to create dependency graphs. The Stanford system can produce several variants of the Stanford Dependencies (SD) representation. Here, we use the *collapsed* variant, which is designed to be useful for information extraction and language understanding tasks (de Marneffe and Manning, 2008).

3.2 Shortest Dependency Path

The syntactic structure connecting two entities e_1 and e_2 in various forms of syntactic analysis is known to contain most of the words relevant to characterizing the relationship $R(e_1, e_2)$, while excluding less relevant and uninformative words.

This observation has served as the basis for many successful relation extraction approaches in both general and biomedical domain NLP (Bunescu and Mooney, 2005; Airola et al., 2008; Nguyen et al., 2009; Chowdhury et al., 2011). The TEES system also heavily relies on the shortest dependency path for defining and ex-

tracting features (Björne et al., 2012; Björne and Salakoski, 2013). Recently, this idea was applied in an LSTM-based relation extraction system by Xu et al. (2015). Since the dependency parse is directed (i.e. the path from e_1 to e_2 differs from that from e_2 to e_1), they separate the shortest dependency path into two sub-paths, each from an entity to the common ancestor of the two entities, generate features along the two sub-paths, and feed them into different LSTM networks, to process the information in a direction sensitive manner.

To avoid doubling the number of LSTM chains (and hence the number of weights), we convert the dependency parse to an undirected graph, find the shortest path between the two entities (BACTERIA and HABITAT/GEOGRAPHICAL), and always proceed from the BACTERIA entity to the HABITAT/GEOGRAPHICAL entity when generating features along the shortest path, regardless of the order of the entity mentions in the sentence. With this approach, there is a single LSTM chain (and set of LSTM weights) for every feature set, which is more effective when the number of training examples is limited.

There is a subtle and important point to be addressed here: as individual entity mentions can consist of several (potentially discontinuous) tokens, the method must be able to select which word (i.e. single token) serves as the starting/ending point for paths through the dependency graph. For example, in the following training set sentence, “*biotic surfaces*” is annotated as a HABITAT entity:

“*We concluded that S. marcescens MG1 utilizes different regulatory systems and adhesins in attachment to biotic and abiotic surfaces [...]*”

As this mention consists of two (discontinuous) tokens, it is necessary to decide whether the paths connecting this entity to BACTERIA mentions (e.g., “*S. marcescens MG1*”) should end at “*biotic*” or “*surfaces*”. This problem has fortunately been addressed in detail in previous work, allowing us to adopt the proven solution proposed by Björne et al. (2012) and implemented in the TEES system, which selects the *syntactic head*, i.e. the root token of the dependency parse sub-tree covering the entity, for any given multi-token entity. Hence, in the example above, the token “*surfaces*” is selected and used for finding the shortest dependency paths.

3.3 Neural Network Architecture

While recurrent neural networks (RNNs) are inherently suitable for modeling sequential data, standard RNNs suffer from the *vanishing or exploding gradients* problem: if the network is deep, during the back-propagation phase the gradients may either decay exponentially, causing learning to become very slow or stop altogether (*vanishing gradients*); or become excessively large, causing the learning to diverge (*exploding gradients*) (Bengio et al., 1994). To avoid this issue, we make use of Long Short-Term Memory (LSTM) units, which were proposed to address this problem (Hochreiter and Schmidhuber, 1997).

We propose an architecture centered around three RNNs (chains of LSTM units): one representing *words*, the second *POS tags*, and the third *dependency types* (Figure 1). For a given example, the sequences of words, POS tags and dependency types on the shortest dependency path from the BACTERIA mention to the HABITAT/GEOGRAPHICAL mention are first mapped into vector sequences by three separate embedding lookup layers. These word, POS tag and dependency type vector sequences are then input into the three RNNs. The outputs of the last LSTM unit of each of the three chains are then concatenated and the resulting higher-dimensional vector input to a fully connected hidden layer. The hidden layer finally connects to a single-node binary classification layer.

Based on experiments on the development set, we have set the dimensionality of all LSTM units and the hidden layer to 128. The sigmoid activation function is applied on the output of all LSTM units, the hidden layer and the output layer.

3.4 Features and Embeddings

We next present the different embeddings defining the primary features of our model. In addition to the embeddings, we use a binary feature which has the value 0 if the corresponding location is a GEOGRAPHICAL entity and 1 if it is a HABITAT entity. This input is directly concatenated with the LSTM outputs and fed into the hidden layer. We noticed this signal slightly improves classification performance, resulting in a less than 1 percentage point increase of the F-score.

3.4.1 Word embeddings

We initialize our word embeddings with vectors induced using six billion words of biomedical

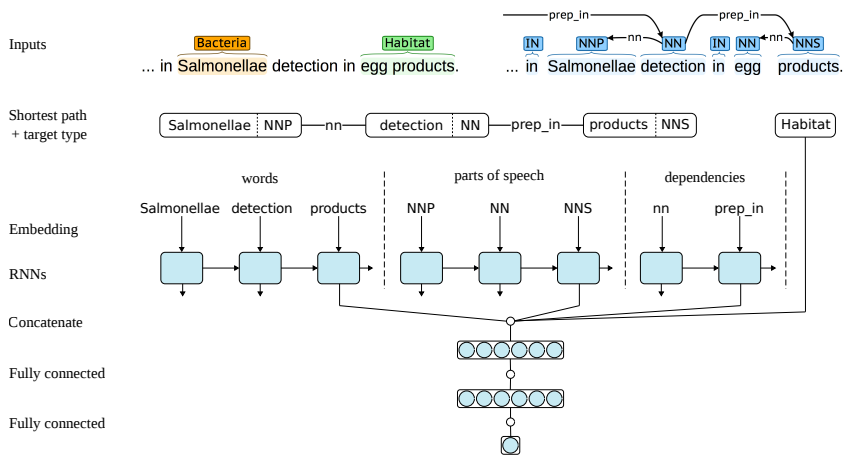


Figure 1: Proposed network architecture.

scientific text, namely the combined texts of all PubMed titles and abstracts and PubMed Central Open Access (PMC OA) full text articles available as of the end of September 2013.² These 200-dimensional vectors were created by Pyysalo et al. (2013) using the *word2vec* implementation of the *skip-gram* model (Mikolov et al., 2013).

To reduce the memory requirements of our method, we only use the vectors of the 100,000 most frequent words to construct the embedding matrix. Words not included in this vocabulary are by default mapped to a shared, randomly initialized unknown word vector. As an exception, out of vocabulary BACTERIA mentions are instead mapped to the vector of the word “*bacteria*”. Based on development set experiments we estimate that this special-case processing improved the F-score by approximately 1% point.

3.4.2 POS embeddings

Our POS embedding matrix consists of a 100-dimensional vector for each of the POS tags in the Penn Treebank scheme used by the applied tagger. We do not use pre-trained POS vectors but instead initialize the embeddings randomly at the beginning of the training phase.

3.4.3 Dependency type embeddings

Typed dependencies – the edges of the parse graph – represent directed grammatical relations between the words of a sentence. The sequence of dependencies on the shortest path between two entities thus conveys highly valuable information about the nature of their relation.

We map each dependency type in the collapsed SD representation into a randomly initialized 350-dimensional vector (size set experimentally). Note that in the applied SD variant, prepositions and conjunctions become part of collapsed dependency types (de Marneffe et al., 2006), as illustrated in Figure 2.

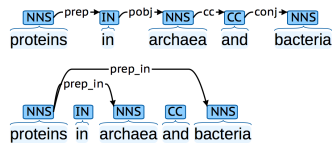


Figure 2: Basic (top) and collapsed (bottom) Stanford Dependency representations

As the collapsed dependencies thus incorporate preposition and conjunction words into the grammatical relations themselves, the set of dependency types is somewhat open-ended. To account for this, all preposition/conjunction dependency types not observed in the training and develop-

²Available from <http://bio.niplab.org/>

ment sets are mapped to the vectors for the general preposition and conjunction types `prep` and `conj`, respectively.

3.5 Training and Regularization

We use *binary cross-entropy* as the objective function and the *Adam* optimization algorithm with the parameters suggested by Kingma and Ba (2014) for training the network. We found that this algorithm yields considerably better results than the conventional stochastic gradient descent in terms of classification performance.

During training, the randomly initialized POS and dependency type embeddings are trained and the pre-trained word embeddings fine-tuned by back-propagation using the supervised signal from the classification task at hand.

Determining how long to train a neural network model for is critically important for its generalization performance. If the network is *under-trained*, model parameters will not have converged to good values. Conversely, *over-training* leads to overfitting on the training set. A conventional solution is early stopping, where performance is evaluated on the development set after each set period of training (e.g. one pass through the training set, or *epoch*) to decide whether to continue or stop the training process. A simple rule is to continue while the performance on the development set is improving. By repeating this approach for 15 different runs with different initial random initializations of the model, we experimentally concluded that the optimal length of training is four epochs. Overfitting is a serious problem in deep neural networks with a large number of parameters. To reduce overfitting, we experimented with several regularization methods including the l_1 weight regularization penalty (*LASSO*) and the l_2 weight decay (*ridge*) penalty on the hidden layer weights. We also tried the dropout method (Srivastava et al., 2014) on the output of LSTM chains as well as on the output of the hidden layer, with a dropout rate of 0.5. Out of the different combinations, we found the best results when applying dropout after the hidden layer. This is the only regularization method used in the final method.

4 Results

4.1 Overcoming Variance

At the beginning of training, the weights of the neural network are initialized randomly. As we are

| Run | Recall | Precision | F-score |
|-----------|-------------|-------------|-------------|
| 12 | 76.3 | 60.3 | 67.3 |
| 14 | 71.2 | 63.0 | 66.8 |
| 13 | 75.7 | 59.3 | 66.5 |
| 10 | 78.0 | 56.3 | 65.4 |
| 3 | 80.8 | 54.0 | 64.7 |
| 15 | 79.1 | 54.3 | 64.4 |
| 1 | 66.1 | 62.2 | 64.1 |
| 11 | 65.0 | 62.8 | 63.9 |
| 2 | 67.8 | 59.4 | 63.3 |
| 5 | 55.9 | 69.7 | 62.1 |
| 7 | 57.6 | 66.7 | 61.8 |
| 9 | 53.1 | 70.2 | 60.5 |
| 8 | 50.9 | 74.4 | 60.4 |
| 6 | 50.3 | 73.6 | 59.7 |
| 4 | 46.9 | 78.3 | 58.7 |
| \bar{x} | 65.0 | 64.3 | 63.3 |
| σ | 11.3 | 7.3 | 2.6 |

Table 2: Development set results for 15 repetitions with different initial random initializations with mean (\bar{x}) and standard deviation (σ). Results are sorted by F-score.

only using pre-trained embeddings for words, this random initialization applies also to the POS and dependency type embeddings. Since the number of weights is high and the training set is very small (only 524 examples), the initial random state of the model can have a significant impact on the final model and its generalization performance. Limited numbers of training examples are known to represent significant challenges for leveraging the full power of deep neural networks, and we found this to be the case also in this task.

To study the influence of random effects on our model, we evaluate it with 15 different random initializations, training each model for four epochs on the training data and evaluating on the development set using the standard precision, recall and F-score metrics. Table 2 shows the obtained results. We find that the primary evaluation metric, the F-score, varies considerably, ranging from 58.7% to 67.3%. This clearly illustrates the extent to which the random initialization can impact the performance of the model on unseen data. While the method is shown to obtain on average an F-score of 63.3% on the development set, it must be kept in mind that given the standard deviation of 2.6, individual trained models may perform substantially better (or worse). It is also important to note that due to the small size of the development

| Threshold (t) | Recall | Precision | F-score |
|-------------------|-------------|-------------|-------------|
| 1 | 83.6 | 53.2 | 65.1 |
| 2 | 79.7 | 54.0 | 64.4 |
| 3 | 78.5 | 57.0 | 66.0 |
| 4 | 78.0 | 59.0 | 67.2 |
| 5 | 75.7 | 60.1 | 67.0 |
| 6 | 70.6 | 60.7 | 65.3 |
| 7 | 67.8 | 61.5 | 64.5 |
| 8 | 65.5 | 62.0 | 63.7 |
| 9 | 62.2 | 65.5 | 63.8 |
| 10 | 58.2 | 66.5 | 62.1 |
| 11 | 57.1 | 69.7 | 62.7 |
| 12 | 52.5 | 70.5 | 60.2 |
| 13 | 51.4 | 72.8 | 60.3 |
| 14 | 48.6 | 74.8 | 58.9 |
| 15 | 45.2 | 80.0 | 57.8 |

Table 3: Development set results for voting based on the predictions of the 15 different classifiers. Best results for each metric shown in bold.

set, individual models that achieved high performance in this experiment will not necessarily generalize well to unseen data.

To deal with these issues, we introduce a straightforward voting procedure that aggregates the prediction outputs of the 15 classifiers based on a given threshold value $t \in \{1, \dots, 15\}$:

1. For each example, predict outputs with the 15 models;
2. If at least t outputs are positive, label the example positive, otherwise label it negative.

Clearly, the most conservative threshold is $t = 15$, where a relation is voted to exist only if *all* the 15 classifiers have predicted it. Conversely, the least conservative threshold is $t = 1$, where a relation is voted to hold if *any* classifier has predicted it.

The development set results for the voting algorithm with different threshold values are given in Table 3. As expected, the threshold $t = 1$ produces the highest recall (83.6%) with the lowest precision (53.2%). With increasing values of t , precision increases while recall drops, and the highest precision (80.0%) is achieved together the lowest recall of (45.2%) with $t = 15$. The best F-score is obtained with $t = 4$, where an example is labeled positive if at least four classifiers have predicted it to be positive and negative otherwise. Figure 3 shows the precision-recall curve for these 15 threshold values.

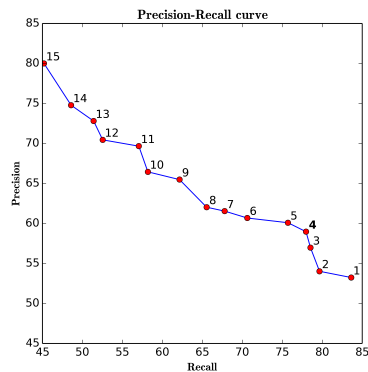


Figure 3: Precision-recall curve for different values of the threshold t (shown as labels on the curve).

As is evident from these results, the voting algorithm can be used for different purposes. If the aim is to obtain the best overall performance, we can investigate which threshold produces the highest F-score (here $t = 4$) and select that value when making predictions for unseen data (e.g., the test set). Alternatively, for applications that specifically require high recall or high precision, a different threshold value can be selected to optimize the desired metric.

To assess the performance of the method on the full, *unfiltered* development set that includes also cross-sentence relations, we selected the threshold value $t = 4$ and submitted the aggregated prediction results to the official Shared Task evaluation server. The method achieved an F-score of 60.0% (60.9% precision and 59.3% recall), 7.2% points below the result for our internal evaluation with filtered data (Table 3).

4.2 Test Set Evaluation

For evaluation of the test set, we applied the proposed model with the voting approach presented above): 15 neural network models with different random initializations were trained for 4 epochs on the combination of the training and the development sets. Each trained model was then used to produce one set of predictions for the test set. To obtain the final test set predictions, the outputs of the 15 classifiers were aggregated using the voting algorithm with a threshold $t = 4$.

Our method achieved an F-score of 52.1% with a recall of 44.8% and precision of 62.3%, ranking second among the entries to the shared task. We again emphasize that our approach ignored all potential relations between entities belonging to different sentences, which may in part explain the comparatively low recall.

4.3 Runtime Performance and Technical Details

We implemented the model using the Python programming language (v2.7) with Keras, a model-level deep learning library (Chollet, 2015). All network parameters not explicitly discussed above were left to their defaults in Keras. The Theano tensor manipulation library (Bastien et al., 2012) was used as the backend engine for Keras. Computations were run on a single server computer equipped with a GPU.³ All basic python processing, including e.g. file manipulation, the TEES pipeline and our voting algorithm, was run on a single CPU core, while all neural network related calculations (training, optimization, predictions) were run on the GPU, using the CUDA toolkit version 5.0.

The training process takes about 10 minutes, including model building and 4 epochs of training the network on the training set, but excluding pre-processing and the creation and loading of the input network. Prediction of the development set using a trained model with fully prepared inputs is very fast, taking only about 10 seconds. Finally, the voting algorithm executes in less than a minute for all 15 thresholds.

We note that even though the proposed approach involving 15 rounds of training, prediction and result aggregation might seem to be impractical for large-scale real-word applications (e.g., extracting bacteria-location relations from all PubMed abstracts), it is quite feasible in practice, as the time-consuming training process only needs to be done once, and prediction on unseen data is quite fast.

4.4 Other Architectures

In this section, we discuss alternative approaches that we considered and the reasons why they were rejected in favor of that described above.

³In detail: two 6-core Intel® Xeon® E5-2620 processors, 32 gigabytes of main memory, and one NVIDIA® TESLA™ C2075 companion processor with 448 CUDA cores and 6 gigabytes of memory.

One popular and proven method for relation extraction is to use three groups of features, based on the observation that the words preceding the first entity, the words between the entities, and those after the second entity serve different roles in deciding whether or not the entities are related (Bunescu and Mooney, 2006). Given a sentence $S = w_1, \dots, e_1, \dots, w_i, \dots, e_2, \dots, w_n$ with entities e_1 and e_2 , one can represent the sentence with three groups of words: $\{before\}_{e_1}\{middle\}_{e_2}\{after\}$ (e_1 and e_2 can also be included in the groups). The similarity of two examples represented in this way can be compared using e.g. *sub-sequence kernels* at word level (Bach and Badaskar, 2007). Bunescu and Mooney (2006) utilize three subkernels matching combinations of the before, middle and after sequences of words, with a combined kernel that is simply the sum of the subkernels. This kernel is then used with support vector machines for relation extraction. Besides the words, other features such as the corresponding POS tags and entity types can also be incorporated into such kernel functions to further improve the representation.

We adapted this idea to deep neural networks. We started with the simplest architecture, which contains 3 LSTM networks. Instead of generating features based on the shortest path, each LSTM receives inputs based on the sequence of the words seen in each of the *before*, *middle*, and *after* groups, where the word embeddings are the only features used for classification. Similar to the architecture discussed in Section 3.3, the outputs of the last LSTM units in each chain are concatenated, and the resulting higher-dimensional vector is then fed into a fully connected hidden layer and then to the output layer. This approach has a major advantage over the shortest dependency path, in particular for large-scale applications: parsing, *the most time-consuming part* in the relation extraction pipeline, is no longer required.

Unfortunately, our internal evaluation on the development set showed that this model failed to achieve results comparable to those of the shortest dependency path model, only reaching an F-score of about 57%. Hence, we attempted to use more features by adding 3 or 6 additional LSTM chains to the model, for POS or/and dependency type embeddings. Even in these cases, the F-scores only varied in the range of 57% to about 63% (for different random initializations). We conclude

that even though not requiring parsing is a benefit in these approaches, our experiments suggest that they are not capable of reaching performance comparable to methods that use the syntactic structure of sentences.

5 Conclusions and Future work

We have presented the entry of the TurkuNLP team to the Bacteria Biotope event extraction (BB3-event) sub-task of the BioNLP Shared Task 2016. Our method is based on a combination of LSTM networks over syntactic dependency graphs. The features for the network are derived from the POS tags, dependency types, and word forms occurring on the shortest dependency path connecting the two candidate entities (BACTERIA and HABITAT/GEOGRAPHICAL) in the collapsed Stanford Dependency graph.

We initialize word representations using pre-trained vectors created using six billion words of biomedical text (PubMed and PMC documents). During training, the pre-trained word embeddings are fine-tuned while randomly initialized POS and dependency type representations are trained from scratch. We showed that as the number of training examples is very limited, the random initialization of the network can considerably impact the quality of the learned model. To address this issue, we introduced a voting approach that aggregates the outputs of differently initialized neural network models. Different aggregation thresholds can be used to select different precision-recall trade-offs. Using this method, we showed that our proposed deep neural network can be efficiently trained to have good generalization for unseen data even with minimal training data. Our method ranked second among the entries to the shared task, achieving an F-score of 52.1% with 62.3% precision and 44.8% recall.

There are a number of open questions regarding our model that we hope to address in future work. First, we observed how the initial random state of the model can impact its final performance on unseen data. It is interesting to investigate whether (and to what extent) pre-training the POS and dependency type embeddings can address this issue. One possible approach would be to apply the method to similar biomedical relation extraction tasks that include larger corpora than the BB3-event task (Pyysalo et al., 2008) and use the learned POS and dependency embeddings for ini-

tialization for this task. This could also establish to what extent pre-training these representations can boost the F-score.

Second, it will be interesting to study how the method performs with different amounts of training data. On one hand, we can examine to what extent the training corpus size can be reduced without compromising the ability of the proposed network to learn the classification task; on the other, we can explore how this deep learning method compares with previously proposed state-of-the-art biomedical relation extraction methods on larger relation extraction corpora.

Third, the method and task represent an opportunity to study how the word embeddings used for initialization impact relation extraction performance and in this way assess the benefits of different methods for creating word embeddings in an extrinsic task with real-world applications.

Finally, it is interesting to investigate different methods to deal with cross-sentence relations. Here we ignored all potential relations where the entities are mentioned in different sentences as there is no path connecting tokens across sentences in the dependency graph. One simple method that could be considered is to create an artificial “paragraph” node connected to all sentence roots to create such paths (cf. e.g. Melli et al. (2007)).

We aim to address these open questions and further extensions of our model in future work.

Acknowledgments

We would like to thank Kai Hakala, University of Turku, for his technical suggestions for the pipeline development and the anonymous reviewers for their insightful comments and suggestions. We would also like to thank the CSC IT Center for Science Ltd. for computational resources. This work has been supported in part by Medical Research Council grant MR/M013049/1.

References

- Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):1.
- Sophia Ananiadou, Sampo Pyysalo, Junichi Tsujii, and Douglas B Kell. 2010. Event extraction for sys-

- tems biology by text mining the literature. *Trends in biotechnology*, 28(7):381–390.
- Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Language Technologies Institute, Carnegie Mellon University*.
- Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*.
- Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.
- Jari Björne and Tapio Salakoski. 2013. TEES 2.1: Automated annotation scheme learning in the bionlp 2013 shared task. In *Proc. BioNLP Shared Task*, pages 16–25.
- Jari Björne, Filip Ginter, and Tapio Salakoski. 2012. University of Turku in the BioNLP’11 shared task. *BMC Bioinformatics*, 13(S-11):S4.
- Robert Bossy, Julien Jourde, Philippe Bessières, Maarten van de Guchte, and Claire Nédellec. 2011. Bionlp shared task 2011: Bacteria biotope. In *Proc. BioNLP Shared Task*, pages 56–64.
- Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. HLT-EMNLP*, pages 724–731.
- Razvan Bunescu and Raymond J. Mooney. 2006. Sub-sequence kernels for relation extraction. In *Proc. NIPS*.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best parsing and maxent discriminative reranking. In *Proc. ACL*, pages 173–180.
- Franois Chollet. 2015. Keras. <https://github.com/fchollet/keras>.
- Faisal Mahbub Chowdhury, Alberto Lavelli, and Alessandro Moschitti. 2011. A study on dependency tree kernels for automatic extraction of protein-protein interaction. In *Proc. BioNLP 2011*, pages 124–133.
- Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Proc. CrossParser*, pages 1–8.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. LREC-2006*, pages 449–454.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.
- Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun’ichi Tsujii. 2009. Overview of BioNLP’09 shared task on event extraction. In *Proc. BioNLP Shared Task*, pages 1–9.
- Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Junichi Tsujii. 2011. Extracting bio-molecular events from literature - the BioNLP’09 shared task. *Computational Intelligence*, 27(4):513–540.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- David McClosky. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis.
- Gabor Melli, Martin Ester, and Anoop Sarkar. 2007. Recognition of multi-sentence n-ary subcellular localization mentions in biomedical abstracts. In *Proceedings of LBM*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.
- Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP shared task 2013. In *Proc. BioNLP Shared Task*, pages 1–7.
- Truc-Vien T Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proc. EMNLP*, pages 1378–1387.
- Sampo Pyysalo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. *BMC bioinformatics*, 9(3):1.
- Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantic resources for biomedical text mining. In *Proc. LBM*, pages 39–44.
- Zorana Ratkovic, Wiktoria Golik, Pierre Warnier, Philippe Veber, and Claire Nédellec. 2011. BioNLP 2011 task Bacteria Biotope: The Alvis system. In *Proc. BioNLP Shared Task*, pages 102–111.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. EMNLP*, pages 1785–1794.

**Farrokh Mehryary, Kai Hakala, Suwisa Kaewphan, Jari
Björne, Tapio Salakoski, Filip Ginter
End-to-End System for Bacteria Habitat Extraction**

Proceedings of the BioNLP 2017, Association for Computational Linguistics,
Vancouver, Canada,
2017, 80–90



End-to-End System for Bacteria Habitat Extraction

Farrokh Mehryary^{1,2*}, Kai Hakala^{1,2*}, Suwisa Kaewphan^{1,2,3*},
Jari Björne¹, Tapio Salakoski^{1,3} and Filip Ginter¹

1. Turku NLP Group, Department of FT, University of Turku, Finland

2. The University of Turku Graduate School (UTUGS), University of Turku, Finland

3. Turku Centre for Computer Science (TUCS), Finland

firstname.lastname@utu.fi

Abstract

We introduce an end-to-end system capable of named-entity detection, normalization and relation extraction for extracting information about bacteria and their habitats from biomedical literature. Our system is based on deep learning, CRF classifiers and vector space models. We train and evaluate the system on the BioNLP 2016 Shared Task Bacteria Biotope data. The official evaluation shows that the joint performance of our entity detection and relation extraction models outperforms the winning team of the Shared Task by 19pp on F-score, establishing a new top score for the task. We also achieve state-of-the-art results in the normalization task. Our system is open source and freely available at <https://github.com/TurkuNLP/BHE>.

1 Introduction

Knowledge about habitats of bacteria is crucial for the study of microbial communities, e.g. metagenomics, as well as for various applications such as food processing and health sciences. Although this type of information is available in the biomedical literature, comprehensive resources accumulating the knowledge do not exist (Deléger et al., 2016).

The BioNLP Bacteria Biotope (BB) Shared Tasks are organized to provide a common evaluation platform for language technology researchers interested in developing information extraction methods adapted for the detection of bacteria and their physical locations mentioned in the literature. So far three BB shared tasks have been organized, the latest in 2016 (BB3) consisting of three main

subtasks: named entity recognition and categorization (BB3-cat and BB3-cat+ner), event extraction (BB3-event and BB3-event+ner) and knowledge base extraction. The NER task includes three relevant entity types: HABITAT, BACTERIA and GEOGRAPHICAL, the categorization task focuses on normalizing the mentions to established ontology concepts, although GEOGRAPHICAL entities are excluded from this task, whereas the event extraction aims at finding the relations between these entities, i.e. extracting in which locations certain bacteria live in. The knowledge base extraction task is centered upon aggregating this type of information from a large text corpus.

In this paper we revisit the BB3 subtasks of NER, categorization and event extraction, all of which are essential for building a real-world information extraction pipeline. As a result, we present a text mining pipeline which achieves state-of-the-art results for the joint evaluation of NER and event extraction as well as for the categorization task using the official BB3 shared task datasets and evaluation tools. Building such end-to-end system is important for bringing the results from the shared tasks to the actual intended users. To our best knowledge, no such system is openly available for bacteria habitat extraction.

The pipeline utilizes deep neural networks, conditional random field classifiers and vector space models to solve the various subtasks and the code is freely available at <https://github.com/TurkuNLP/BHE>. In the following sections we discuss our system, divided into three modules: entity recognition, categorization and event extraction. We then analyze the results and finally discuss the potential future research directions.

*These authors contributed equally.

2 Method

2.1 Named entity detection

Detecting the BB3 HABITAT, BACTERIA and GEOGRAPHICAL mentions is a standard named entity recognition task, evaluated based on the correctness of the type and character offsets of the discovered text spans. In our NER pipeline, all documents are preprocessed following the approach of Hakala et al. (2016). In brief, we first convert all documents and annotation files from UTF-8 to ASCII encoding using a modified version of publicly available tool designed for parsing PubMed documents (Pyysalo et al., 2013)¹. Next we split documents into sentences using the Genia Sentence Splitter (Sætre et al., 2007) and the sentences are subsequently tokenized and part-of-speech tagged using the tokenization and POS-tagging modules in NERSuite², respectively.

To detect the entity mentions we use NERSuite, a named entity recognition toolkit, as it is relatively easy to train on new corpora, yet supports adding novel user-defined features. In biomedical NER, NERSuite has been a versatile tool achieving excellent performance for various entity types (Ohta et al., 2012; Kaewphan et al., 2014, 2016), however, it is not capable of dealing with overlapping entities. Therefore, we only use the longest spans of overlapping annotated entities as our training data, ignoring embedded entities which are substrings of the longest spans.

In biomedical NER, domain knowledge such as controlled vocabularies has been crucial for achieving high performance. In this work we prepare 3 dictionaries, specific for each entity type. For BACTERIA, we compile a dictionary of names exclusively from the NCBI Taxonomy database³ by including all names under bacteria superkingdom (NCBI taxonomy identifier 2). The *scientific names* are expanded to include abbreviations whose genus names are conventionally abbreviated with the first and/or second alphabet, whereas the rest of the names, such as species epithet and strains, remains unchanged. For HABITAT, we combine all symbols from the OntoBiotope ontology⁴ and use them without any further modifications. Similar to HABITAT, we also prepare dictionary for GEOGRAPHICAL by taking all

strings under the semantic type *geographical area* from UMLS database (version 2016AA) (Bodenreider, 2004). All dictionaries prepared in this step are directly provided to NERSuite through the dictionary-tagging module without any normalization. The tagging provides additional features describing whether the tokens are present in some semantic categories, such as bacteria names or geographical places. For GEOGRAPHICAL model, we also add token-level tagging results for *location* from Stanford NER (SNER) (Finkel et al., 2005) as binary values to NERSuite; 1 and 0 for location and non-location, respectively.

Although utilizing dictionary features is beneficial for NER, strict string matching tends to lead to low coverage, an issue which is also common in the categorization task. To remedy this problem, we also generate fuzzy matching features based on our categorization system (see Section 2.2) by measuring the maximum similarity of each token against the NCBI Taxonomy and OntoBiotope ontologies for BACTERIA and HABITAT respectively. Thus, instead of a binary feature denoting whether a token is present in the ontology or not, a similarity score ranging from 0 to 1 is assigned for each token. This approach is similar to (Kaewphan et al., 2014), but instead of using word embedding similarities, our fuzzy matching relies on character ngrams. We do not use these features for the GEOGRAPHICAL entities, which are not categorized by our system.

In the official BB3 evaluation, NER is jointly evaluated with either categorization or event extraction system. In BB3-cat+ner task, SER (Slot Error Rate) is used as the main scoring metric, whereas in BB3-event+ner, participating teams are ranked based on F-score of extracted relations. Due to the lack of an official evaluation on NER for all entities in BB3-event+ner and for GEOGRAPHICAL in BB3-cat+ner, we use our own implementation by calculating the F-score using exact string matching criteria as our main scoring metric. In this study, we consider BB3-event+ner as our primary subtask and thus all hyper-parameters in model selection are optimized against F-score instead of SER.

2.2 Named entity categorization

In the BB3 categorization subtask each BACTERIA and HABITAT mention has to be assigned to the corresponding ontology concepts, specifically

¹<https://github.com/spyysalo/nxml2txt>

²<http://nersuite.nlplab.org/>

³<https://www.ncbi.nlm.nih.gov/taxonomy>

⁴<http://agroportal.lirmm.fr/ontologies/ONTOBIOTOPE>

to NCBI Taxonomy and OntoBiotope identifiers respectively. This task is commonly known as named entity normalization or entity linking and various approaches ranging from Levenshtein edit distances to recurrent neural networks have been suggested as the plausible solutions (Tiftikci et al., 2016; Limsopatham and Collier, 2016).

Our categorization method is based on the common approach of TFIDF weighted sparse vector space representations (Salton and Buckley, 1988; Leaman et al., 2013; Hakala, 2015), i.e. the problem is seen as an information retrieval task where each concept name in the ontology is considered a document and the IDF weights are based on these names. Consequently, each concept name and each entity mention is represented with a TFIDF weighted vector and the concept with the highest cosine similarity is assigned for a given entity. Whereas these representations are commonly formed in a bag-of-words fashion, in our experiments using character-level ngrams resulted in better outcome. In the final system we use ngrams of length 1, 2 and 3 characters. These ngram lengths produced the highest accuracy on the official development set for both BACTERIA and HABITAT entities, each entity type evaluated separately. The TFIDF vectorization was implemented using the scikit-learn library (Pedregosa et al., 2011) and default parameter values except for using the character level ngrams instead of words.

For both included ontologies we use the preferred names as well as the listed synonyms to represent the concepts. Since the task is restricted to bacteria mentions instead of all organisms, we also narrow down the NCBI Taxonomy ontology to cover only the Bacteria superkingdom, i.e. the categorization system is not allowed to assign taxonomy identifiers which do not belong to this superkingdom. Otherwise all concepts from the used ontologies are included.

As preprocessing steps we use three main approaches: abbreviation expansion, acronym expansion and stemming. For stemming we use the Porter stemmer (Porter, 1980) and stem each token in the entities and concept names. According to our evaluation this is not beneficial for the BACTERIA entities and is thus included only for the HABITAT entities.

In biomedical literature the genus names in BACTERIA mentions are commonly shortened af-

ter the first mention, e.g. *Staphylococcus aureus* is abbreviated as *S. aureus*, but the NCBI Taxonomy ontology does not include these abbreviated forms as synonyms for the corresponding concepts. Thus, if an entity mention includes a token with a period in it, we expand the given token by finding the most common token with the same initial from all previously mentioned entities of the same type within the same document.

Another commonly used naming convention for BACTERIA mentions is forming acronyms, e.g. *lactic acid bacteria* is often referred to as *LAB*. Consequently, when we detect a BACTERIA mention with less than five characters or written in uppercase, we try to find the corresponding full form by generating acronyms for all previously mentioned BACTERIA entities by simply concatenating their initials. However, many BACTERIA acronyms do not follow this format strictly, e.g. *Lactobacillus casei strain Shirota* should be shortened to *LcS* instead of *LCSS* and *Francisella tularensis Live Vaccine Strain* as *LVS* instead of *FTLVS*. Thus, instead of using strict matching to find the corresponding full form, we utilize the same character-level TFIDF representations as used for the actual categorization for these acronyms to find the most similar full form. In our evaluation, using the same approach for HABITAT entities dramatically decreased the performance hence was thus not used for this entity type (see Section 3.2).

Both of these expansion methods have similar intentions as the preprocessing steps utilized by the winning system in BB3 (BOUN) by Tiftikci et al. (2016), but our system uses more relaxed criteria for finding the full forms and should thus result in better recall at the expense of precision.

2.3 Event extraction

The BB3-event and BB3-event+ner tasks demand extraction of undirected binary associations of two named entities: a BACTERIA entity and either a HABITAT or a GEOGRAPHICAL entity; and these relations represent the locations in which bacteria live. We thus formulate this task as a binary classification task and assign the label *positive* if such relation holds for a given entity pair and *negative* otherwise.

To address this task, we present a deep learning-based relation extraction system that generates features along the *shortest dependency path (SDP)*

| | Train | Devel | Test |
|------------------------|-------|-------|------|
| Total sentences | 527 | 319 | 508 |
| Sentences w/examples | 158 | 117 | 158 |
| Sentences w/o examples | 369 | 202 | 350 |
| Total examples | 524 | 506 | 534 |
| Positive examples | 251 | 177 | - |
| Negative examples | 273 | 329 | - |

Table 1: BB3-event data statistics.

which connects the two candidate entities in the syntactic parse graph. Many successful relation extraction systems have been built utilizing SDP (Cai et al., 2016; Mehryary et al., 2016; Xu et al., 2015; Björne and Salakoski, 2013; Björne et al., 2012; Bunescu and Mooney, 2005) since it is known to contain most of the relevant words for expressing the relation between the two entities while excluding less relevant and uninformative words. Since this approach focuses on a *single* sentence parse graph at a time, it is unable to detect plausible cross-sentence relations, i.e. the cases in which the two candidate entities belong to different sentences. As discussed by Kim et al. (2011), detecting such relations is a major challenge for relation extraction systems. We simply exclude any cross-sentence relations from training, development and test sets.⁵ Table 1 summarizes the statistics of the data that is used for building our relation extraction system after removing cross-sentence relations.

2.3.1 Preprocessing

For preprocessing, we use the preprocessing pipeline of the TEES system (Björne and Salakoski, 2013) which automates tokenization, part-of-speech tagging and sentence parsing. TEES runs the BLLIP parser (Charniak and Johnson, 2005) with the biomedical domain model created by McClosky (2010). The resulting phrase structure trees are then converted to dependency graphs (*nonCollapsed* variant of Stanford Dependency) using the Stanford conversion tool (version 2.0.1) (de Marneffe et al., 2006).

2.3.2 Relation extraction system architecture

The architecture of our deep learning-based relation extraction system is centered around utilizing three separate convolutional neural networks (CNN): for the sequence of *words*, the sequence of

⁵Official evaluation results on the development and test data are of course comparable to those of other systems: any cross-sentence relations in the development/test data count against our submissions as false negatives.

POS tags, and the sequence of *dependency types* (the edges of the parse graph), along the SDP connecting the two candidate entities (see Figure 1). Even though the parse graph is directed, we regard it as an undirected graph and always traverse the SDP by starting the path from the BACTERIA entity mention to the HABITAT/GEOGRAPHICAL, regardless of the order of their occurrence in the sentence. Evaluation against the development set showed that this approach leads to better generalization in comparison with simply traversing the path from the first occurring entity mention to the second (with/without considering the direction of the edges).

The structure of each CNN is similar: the words (or POS tags or dependency types) in the sequence are mapped into their corresponding vector representations using an embedding lookup layer. The resulting sequence of vectors is then forwarded into a convolutional layer which creates a convolution kernel that is applied on the layer input over a single spatial dimension to produce a tensor of outputs. These outputs are then forwarded to a max-pooling layer that gathers information from local features of the SDP. Hence, the three CNNs produce three vector representations.

Subsequently, the output vectors of the CNNs and two 1-hot-encoded entity-type vectors are concatenated. The first entity-type vector represents the type of the first occurring entity in the sentence (BACTERIA, HABITAT or GEOGRAPHICAL), and the other is used for the second one. The resulting vector is then forwarded into a fully connected hidden layer and finally, the hidden layer connects to a single-node binary classification layer.

For the word features, we use a vector space model with 200-dimensional word embeddings pre-trained by Pyysalo et al. (2013). These are fine-tuned during the training while the POS-tag and dependency type embeddings are learned from scratch after being randomly initialized.

Based on experiments on the development set, we have set the dimensionality of the POS tag embeddings to 200, and for dependency types to 300. For all convolutional layers, the number of filters has been set to 100 and the window size (filter length) to 4. Finally, dimensionality of the hidden layer has been set to 100. The *ReLU* activation function is applied on the output of the convolutional layers while we apply *sigmoid* activation to

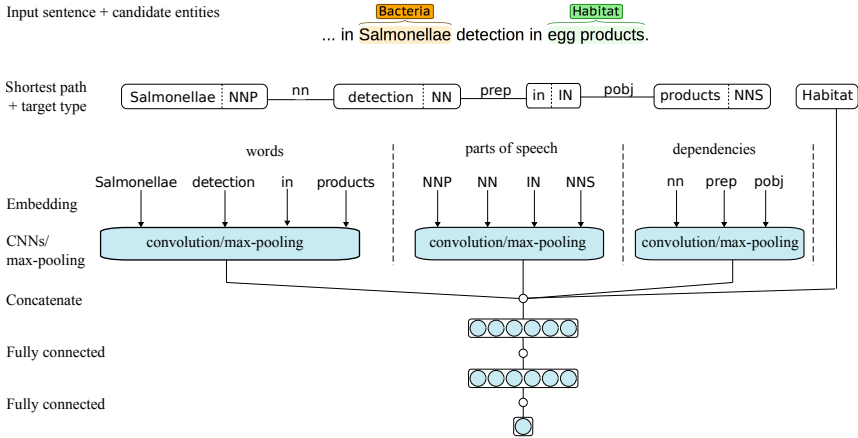


Figure 1: Proposed network architecture.

the output of the hidden layer.

2.3.3 Training and optimization

We use *binary cross-entropy* as the objective function and the *Adam* optimization algorithm (Kingma and Ba, 2014) for training the network. Applying the *dropout* (Srivastava et al., 2014) with rate of 50% on the output of the hidden layer is the only network regularization method used to avoid overfitting.

When the number of weights in a neural network is high and the training set is very small (e.g., there are only 524 examples in the BB3-event training set), the initial random state of the model can have a significant impact on the final model and its generalization performance. Mehryary et al. (2016) have reported that the F-score on the development set of BB3-event task can vary up to 9 percentage points based on the different initial random state of the network.

To overcome this problem, we implement the simple but effective strategy proposed by them, which consists of training the neural network model 15 times with different initial random states, predicting the development/test set examples and aggregating the 15 classifiers' predictions using a simple voting algorithm.

For each development/test example, the voting algorithm combines the predictions based on a given threshold parameter t : the relation is voted

to be positive if at least t classifiers have predicted it to be positive, otherwise, it will be considered as a negative. Obviously, the lowest threshold value ($t = 1$) produces the highest recall and lowest precision and the highest threshold ($t = 15$) produces the highest precision and lowest recall and the aim is to find the best threshold value which maximizes the F-score.

Our experiments on the development set (using the proposed network architecture) showed that for the BB3-event task the optimal results are achieved when we train the networks for 2 epochs and set the threshold value to 4, and for the BB3-event+ner task, when we train the networks for 2 epochs and set the threshold value to 3.

3 Results and discussion

3.1 Named entity detection

For the named entity detection task, we obtain the baseline performance by training NERsuite for each entity-type independently. As shown in Table 2, the F-scores for BACTERIA, GEOGRAPHICAL and HABITAT are 0.713, 0.516 and 0.482 respectively. The baseline performance of HABITAT and GEOGRAPHICAL models is significantly lower than BACTERIA.

For all entities, adding dictionary features improves the performance of the model. A substantial improvement in F-score is found for GEOGRAPHICAL where the performance is increased

| Entity/Experiment | Precision | Recall | F-score |
|---------------------|-----------|--------|--------------|
| Bacteria | | | |
| BB3 | 0.787 | 0.652 | 0.713 |
| BB3 + dict | 0.833 | 0.697 | 0.759 |
| BB3 + tfidf | 0.793 | 0.660 | 0.720 |
| BB3 + tfidf + dict | 0.822 | 0.717 | 0.766 |
| BB3 + BB2 + dict | 0.902 | 0.713 | 0.796 |
| BB3 + BB1 + dict | 0.893 | 0.721 | 0.798 |
| Habitat | | | |
| BB3 | 0.589 | 0.407 | 0.482 |
| BB3 + dict | 0.649 | 0.465 | 0.541 |
| BB3 + tfidf | 0.697 | 0.482 | 0.570 |
| BB3 + tfidf + dict | 0.715 | 0.520 | 0.602 |
| BB3 + BB2 + dict | 0.560 | 0.500 | 0.529 |
| Geographical | | | |
| BB3 | 0.667 | 0.421 | 0.516 |
| BB3 + dict | 0.719 | 0.605 | 0.657 |
| BB3 + SNER | 0.694 | 0.658 | 0.676 |
| BB3 + dict + SNER | 0.788 | 0.684 | 0.732 |
| BB3 + BB2 + dict | 0.903 | 0.737 | 0.812 |

Table 2: The performance of our named entity detection system on BACTERIA, HABITAT and GEOGRAPHICAL mentions using internal evaluation system. The models are evaluated on the BB3 development data.

by more than 14pp compared to 6pp and 5pp for HABITAT and BACTERIA, respectively. Adding fuzzy matching features further improves the F-score for HABITAT by more than 12pp compared to 8pp for BACTERIA. This result shows that having both domain knowledge and relaxed matching criteria can significantly enhance the model performance.

We improve equally the baseline performance for GEOGRAPHICAL by adding features from SNER tagging. The increase in F-score, 0.657 versus 0.676, is about the same as independently adding *UMLS-geographical area* dictionary features. Further increase in F-score is achieved by combining both features, likely due to the expanded coverage of geographical names.

The BB3 corpus is relatively small in terms of entity frequency and the number of unique entities. We explore the possibility of increasing model performance through adding additional training data from previously organized BB Shared Tasks (i.e., BB1 (Bossy et al., 2011) and BB2 (Bossy et al., 2013)). Annotations for BACTERIA mentions are available in both BB1 and BB2 Shared Tasks and we thus train NERsuite models by adding these annotations to the training data. The results show that the models, trained with additional datasets, achieve higher performance. BB1 provides a slightly better F-score than BB2, 0.798 vs 0.796.

For GEOGRAPHICAL and HABITAT entities, compatible annotations are only available from BB2 (Bossy et al., 2013), subtask 2. We thus train NERsuite for both HABITAT and GEOGRAPHICAL by using combined BB3 and BB2 data. The result for GEOGRAPHICAL is similar to the one observed with BACTERIA and additional data can increase the model F-score by more than 15pp. However, the result for HABITAT is different as F-score slightly drops from 0.541 to 0.529. The best NER model for HABITAT thus remains unchanged.

Finally, we train our final model by combining training and development datasets and use hyperparameters obtained from the best performing system on development dataset. The official evaluation of the NER task jointly with either categorization or event extraction system is discussed in Section 3.2 and Section 3.3, respectively.

3.2 Categorization

To analyze our categorization approaches, we evaluate their performance on the official development set. During the development we used accuracy for evaluating the effects of different hyperparameters and preprocessing steps. To get comparable results to previous systems we, however, report the results in this paper using the precision scores from the official evaluation service. As the used ontologies form hierarchical structures, the official evaluation penalizes the incorrect predictions based on the distance from the gold standard annotations, whereas our internal accuracy evaluation measures exact matches. Our accuracy scores and the official evaluation seem to correlate to the level that all improvements validated using the accuracy score also improved the performance according to the official evaluation.

The performance of our system and various preprocessing steps are shown incrementally in Table 3. As a baseline system we use TFIDF bag-of-words representations without any of our preprocessing steps. By simply switching to character level representations the precision is increased by 1.3pp for HABITAT and 14.1pp for BACTERIA mentions.

Adding the abbreviation expansion step further improves precision for BACTERIA by 14.1pp, but does not influence HABITAT entities as most likely there are no abbreviated mentions in this category. The acronym expansion has a lesser, but still no-

ticeable impact and improves precision for BACTERIA by 4.9pp. However, applying this method to HABITAT entities decreases the performance by 4.5pp and is thus left out in the final system for this entity type. This is probably due to the fact that we consider all tokens with less than 5 characters to be acronyms, which seems to hold for BACTERIA mentions, but is a bad assumption for HABITAT entities. The final preprocessing step, stemming, improves the performance on HABITAT entities by mere 1.3pp, but has a negative impact on BACTERIA and is left out for this entity type in the final system.

The results on the official test set are consistently lower than on the development set for both entity types (see Table 4), suggesting that the hyperparameters selected based on the development set might have been slightly overfit on this data. However, our system is able to outperform BOUN (Tiftikci et al., 2016), the winning system from the BioNLP’16 BB3 Shared Task, by 1pp, 1.5pp and 1.2pp on HABITAT, BACTERIA and all entities respectively.

Since the BB3 tasks do not evaluate named entity recognition independently, but only in conjunction with either categorization or event extraction, we also report the official numbers for the BB-cat+ner task in Table 5. In this combined evaluation our system is not able to reach the performance level of the state-of-the-art system TagIt (Cook et al., 2016), but does outperform the other systems which participated in the given subtask.

Our combined system is also performing clearly worse on the test set than on the development set. Unfortunately, due to the test set being blinded, we are unable to specify the exact cause for this. However, the official evaluation service does provide relaxed evaluation modes where e.g. entity boundaries are ignored, i.e. the evaluation focuses on the categorization task. Based on these evaluations our categorization system seems to perform on the same level on both development and test sets, but the performance of our NER model drops, especially for the BACTERIA mentions. This might be simply due to overfitting on the development set, but requires further investigation.

| | Habitat | Bacteria | Overall |
|-----------------|--------------|--------------|--------------|
| BOW TFIDF | 0.634 | 0.531 | 0.568 |
| Char TFIDF | 0.647 | 0.672 | 0.656 |
| + abbreviations | 0.647 | 0.813 | 0.705 |
| + acronyms | 0.602 | 0.862 | 0.693 |
| + stemming | 0.660 | 0.858 | 0.729 |
| Final system | 0.660 | 0.862 | 0.731 |

Table 3: Evaluation of our categorization system with different preprocessing steps compared to a baseline system with TFIDF weighted bag-of-words (unigrams) representations. The scoring is produced by the official evaluation service. Any added processing step, which decreases the performance is left out for the given entity type for the following experiments.

| | Habitat | Bacteria | Overall |
|------------|--------------|--------------|--------------|
| Our system | 0.630 | 0.816 | 0.691 |
| BOUN | 0.620 | 0.801 | 0.679 |

Table 4: Comparison of our entity categorization system and the best performing system in BioNLP’16 BB3 Shared Task on the test set using the official evaluation service.

| | Habitat | Bacteria | Overall |
|-----------------|--------------|--------------|--------------|
| Development set | | | |
| Our system | 0.645 | 0.377 | 0.553 |
| TagIt | 0.511 | 0.303 | 0.439 |
| Test set | | | |
| Our system | 0.804 | 0.706 | 0.766 |
| TagIt | 0.775 | 0.399 | 0.628 |

Table 5: Official results for the combined evaluation of named entity recognition and categorization compared against the state-of-the-art system. The results are evaluated in slot error rate (SER), i.e. a smaller value is better. The scores for the TagIt system are as reported in their paper.

3.3 Event extraction

As discussed earlier, there are two tasks in the BB3 which involve extracting the relations between BACTERIA and HABITAT/GEOGRAPHICAL entities: (1) The BB3-event task, for which all manually annotated entities are given (even for the test set). This task aims to assess the performance of relation extraction systems; (2) The BB3-event+ner task, for which, entities for the test set are hidden and the aim is assessing the joint performance of the NER and the relation extraction systems.

It should be highlighted that the performance of the NER system has a direct impact on the relation extraction system and subsequently on the performance of an end-to-end system for the

BB3-event+ner task. On one hand, if the NER system produces extremely low recall outputs, the relation extraction system will fail to extract some of the valid relations, simply because it only investigates the existence of possible relations among the *given* entities. On the other hand, if the NER system provides high recall but very low precision predictions, this means that words mistakenly detected as valid entities are given to the relation extraction system. For each given entity, the relation extraction system pairs it with other provided entities in the sentence and tries to classify all candidate pairs. Hence, invalid entities will lead to generation of candidate pairs in which one or even both of the entities are actually invalid. Since the relation extraction system is trained on valid entity pairs, i.e., (BACTERIA,HABITAT) or (BACTERIA,GEOGRAPHICAL), it can easily produce a plethora of false-positives and hence, its precision will dramatically drop.

To summarize, if the NER system performance is low (low precision and/or low recall), even a very high-performance relation extraction system will not be able to compensate. Thus, when building an end-to-end system, the joint performance of NER and relation extraction should be assessed since individual performances do not reflect how efficiently the system will work in real-world applications.

The official performance of our relation extraction system alone when evaluated against the test set of the BB3-event task is 0.512 measured in F-score (0.444 recall and 0.605 precision), achieving the third place among Shared Task participants for this task.

| Dataset | Overall | Habitat | Geography |
|---------------------------|---------|---------|-----------|
| Development set | | | |
| With sub-optimal entities | 0.423 | 0.390 | 0.576 |
| With optimal entities | 0.429 | 0.395 | 0.604 |
| Test set | | | |
| With sub-optimal entities | 0.372 | 0.388 | 0.207 |
| With optimal entities | 0.381 | 0.386 | 0.319 |

Table 6: Combined performance of our named entity recognition and event extraction systems on the event+ner task reported in F-score as measured by the official evaluation service.

For the BB3-event+ner task, the official results on the development and the test set are given in Ta-

ble 6. As discussed earlier, to increase the performance of the NER system, we combine the BB3 with older BB datasets. This leads to the best prediction performance (denoted as *optimal*). Thus, we report and compare the overall performance of the end-to-end system when we use these entities. To establish a fair comparison with previously published systems we also report results for models trained only on the BB3 (denoted as *sub-optimal*). As Table 6 shows, using previous BB-ST data for training the NER leads to 3pp increase in F-score of (BACTERIA,GEOGRAPHICAL) relations on the development set and about 11pp for the test set, probably due to the drastically increased performance for GEOGRAPHICAL entity detection. Unfortunately, since there are much less (BACTERIA,GEOGRAPHICAL) relations than (BACTERIA,HABITAT) relations in the data, our approach increases the overall F-score only by 1pp for the test set.

Table 7 compares the performance of our end-to-end system with the winning team in the BB3-event+ner task (LIMSI, developed by Grouin (2016)). As it can be seen in the table, our system outperforms the winning team by 19pp in F-score, achieving the new state-of-the-art score for the task. Even if we solely rely on BB3 data for the NER system, the improvement is 18pp in F-score. We emphasize that no other data than BB3 is used for training/optimization of our relation extraction system in any way.

| Teams | F-score | Recall | Precision | SER |
|------------|--------------|--------------|--------------|--------------|
| LIMSI | 0.192 | 0.191 | 0.193 | 1.558 |
| Our system | 0.381 | 0.292 | 0.548 | 0.891 |

Table 7: Official evaluation results for BB3-event+ner test data of our system compared to LIMSI, the winning team in the Shared Task.

4 Conclusions and future work

In this work, we introduced an open-source end-to-end system, capable of named-entity detection/normalization and relation extraction to extract information about bacteria and their habitats from text. Our system is trained and evaluated on the BioNLP Shared Task 2016 Bacteria Biotope data.

According to the official evaluation, our entity detection and categorization system would have achieved the second place in BB3. Compared to the best performing system on cat+ner, TagIt, we

consider that our approach on NER can still be improved, especially for HABITAT entities. First, we consider employing a *post-processing* step in order to recover embedded entities which are not currently handled by NERSuite. An effective post-processing step should have a substantial impact on our NER system as the embedded entities accounted for over 10% of all HABITAT mentions.

Our categorization system outperforms the best performing system of BB3 by 1.2pp in the official evaluation, constituting the new state-of-the-art for this task. Our system also relies less on rule-based or heuristic preprocessing steps and uses the same general approach for both BACTERIA and HABITAT mentions suggesting that it will be more adaptable for new entity types.

As 9.6% of the HABITAT entities in the official training set have more than one gold standard ontology annotation whereas our current system is only assigning a single concept for each entity, one future work direction is to assess different ways of associating entities with multiple concepts. In the simplest form this could be implemented by defining a similarity threshold instead of selecting only the best matching concept.

Since the character level ngrams resulted in significantly better performance than our word level baseline, the exploration of character level neural approaches is also warranted for the categorization task and will be tested in the future.

Official evaluation shows that the joint performance of entity detection and relation extraction of our end-to-end system outperforms the winning team by 19pp on F-score, establishing a new top score for the event+ner task. In this work we did not use previous BB Shared Task data for training the relation extraction system. However, as a future work we would like to investigate the effect of utilizing previous BB Shared Task data.

As a future work, we would like to run our system on large-scale, on all PubMed abstracts and PubMed Central Open Access full articles to form a publicly available knowledge base.

We highlight that the methods discussed and used in this work are not only applicable for BB3 tasks and can be beneficial for other entity detection/normalization and relation extraction projects as well.

5 Acknowledgements

This work was supported by ATT Tieto käyttöön grant. Computational resources were provided by CSC - IT Center For Science Ltd., Espoo, Finland.

References

- Jari Björne, Filip Ginter, and Tapio Salakoski. 2012. University of Turku in the BioNLP'11 Shared Task. *BMC bioinformatics* 13(11):S4.
- Jari Björne and Tapio Salakoski. 2013. TEES 2.1: Automated annotation scheme learning in the BioNLP 2013 Shared Task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 16–25.
- Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research* 32(suppl 1):D267–D270.
- Robert Bossy, Wiktor Golic, Zorana Ratkovic, Philippe Bessières, and Claire Nédellec. 2013. BioNLP Shared Task 2013—an overview of the Bacteria Biotope task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 161–169.
- Robert Bossy, Julien Jourde, Philippe Bessieres, Maarten Van De Guchte, and Claire Nédellec. 2011. BioNLP Shared Task 2011: Bacteria Biotope. In *Proceedings of the BioNLP Shared Task 2011 Workshop*. Association for Computational Linguistics, pages 56–64.
- Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 724–731.
- Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.
- Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 173–180.
- Helen V Cook, Evangelos Pafilis, and Lars Juhl Jensen. 2016. A dictionary-and rule-based system for identification of bacteria and habitats in text. In *Proceedings of the 4th BioNLP Shared Task 2016 Workshop*. Association for Computational Linguistics, Berlin, Germany.

- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454.
- Louise Deléger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferré, Philippe Bessieres, and Claire Nédellec. 2016. Overview of the Bacteria Biotope task at BioNLP Shared Task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop, Berlin: Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 363–370.
- Cyril Grouin. 2016. Identification of mentions and relations between bacteria and biotope from PubMed abstracts. In *Proceedings of the 4th BioNLP Shared Task Workshop*. Association for Computational Linguistics, Berlin, Germany, pages 64–72.
- Kai Hakala. 2015. UTU: Adapting biomedical event extraction system to disorder attribute detection. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, pages 375–379.
- Kai Hakala, Suwisa Kaewphan, Tapio Salakoski, and Filip Ginter. 2016. Syntactic analyses and named entity recognition for PubMed and PubMed Central-up-to-the-minute. In *Proceedings of the 4th BioNLP Shared Task Workshop*. Association for Computational Linguistics, Berlin, Germany, pages 102–107.
- Suwisa Kaewphan, Kai Hakaka, and Filip Ginter. 2014. UTU: Disease mention recognition and normalization with CRFs and vector space representations. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* pages 807–11.
- Suwisa Kaewphan, Sofie Van Landeghem, Tomoko Ohta, Yves Van de Peer, Filip Ginter, and Sampo Pyysalo. 2016. Cell line name recognition in support of the identification of synthetic lethality in cancer from text. *Bioinformatics* 32(2):276–282.
- Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun’ichi Tsujii. 2011. Extracting bio-molecular events from literature – the BioNLP’09 shared task. *Computational Intelligence* 27(4):513–540.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. <http://arxiv.org/abs/1412.6980>.
- Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. 2013. DNorm: disease name normalization with pairwise learning to rank. *Bioinformatics* page btt474.
- Nut Limsopatham and Nigel Collier. 2016. Normalising medical concepts in social media texts by learning semantic representation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1014–1023.
- David McClosky. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis.
- Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. 2016. Deep learning with minimal training data: TurkuNLP Entry in the BioNLP Shared Task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*. Association for Computational Linguistics, Berlin, Germany, pages 73–81.
- Tomoko Ohta, Sampo Pyysalo, Jun’ichi Tsujii, and Sophia Ananiadou. 2012. Open-domain anatomical entity mention detection. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*. Association for Computational Linguistics, pages 27–36.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.
- Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.
- Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of the 5th International Symposium on Languages in Biology and Medicine (LBM 2013)*. pages 39–44.
- Rune Sætre, Kazuhiro Yoshida, Akane Yakushiji, Yusuke Miyao, Yuichiro Matsubayashi, and Tomoko Ohta. 2007. AKANE system: protein-protein interaction pairs in BioCreAtIvE2 challenge, PPI-IPS subtask. In *Proceedings of the Second BioCreative Challenge Workshop*. pages 209–212.
- Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014.

Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Mert Tiftikci, Hakan Şahin, Berfu Büyükköz, Alper Yayıkçı, and Arzucan Özgür. 2016. Ontology-based categorization of bacteria and habitat entities using information retrieval techniques. In *Proceedings of the 4th BioNLP Shared Task 2016 Workshop*. Association for Computational Linguistics, Berlin, Germany.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

**Kai Hakala, Farrokh Mehryary, Hans Moen, Suwisa
Kaewphan, Tapio Salakoski, Filip Ginter
Ensemble of Convolutional Neural Networks for Medicine
Intake Recognition in Twitter**

Proceedings of the 2nd Social Media Mining for Health Research and
Applications (SMM4H 2017) Workshop, CEUR-WS.org, Washington D.C.,
United States
2017, 59–63

Ensemble of Convolutional Neural Networks for Medicine Intake Recognition in Twitter

Kai Hakala^{1,2*}, Farrokh Mehryary^{1,2*}, Hans Moen^{1,3},
Suwisa Kaewphan^{1,2,4}, Tapio Salakoski^{1,4}, Filip Ginter¹

¹Department of Future Technologies, University of Turku, Turku, Finland;

²University of Turku Graduate School, University of Turku, Turku, Finland;

³Department of Nursing Science, University of Turku, Turku, Finland;

⁴Turku Centre for Computer Science, Turku, Finland

Abstract

We present the results from our participation in the 2nd Social Media Mining for Health Applications Shared Task – Task 2. The goal of this task is to develop systems capable of recognizing mentions of medication intake in Twitter. Our best performing classification system is an ensemble of neural networks with features generated by word- and character-level convolutional neural network channels and a condensed weighted bag-of-words representation. A relatively strong performance is achieved, with an F-score of 66.3 according to the official evaluation, resulting in the 5th place in the shared task with performance close to the best systems created by other participating teams.

Introduction

Pharmacovigilance is the science of detecting, assessing and preventing drug-related adverse effects. A central focus and challenge is to detect adverse drug reactions (ADRs), which are undesired and harmful effects resulting from taking medications. Traditionally, ADRs are identified and recorded by health care professionals, and a part of their work includes weighting the risks and benefits of using medications. However, the number of documented ADRs is limited and it is believed that some of the more rare ADRs have not been revealed yet. As an alternative approach, pharmacovigilance has turned to social media. Social media represents a valuable forum for drug safety surveillance where text-mining techniques can be applied to extract potentially ADR-related events from a large population.

Our team participated in Task 2 of the 2nd Social Media Mining for Health Applications Shared Task at AMIA 2017. The goal of this shared task was to develop systems capable of classifying the mentions of medication intakes in tweets. Each of the provided tweets were to be assigned with one of the following three classes: *personal medication intake*, *possible medication intake* or *non-intake*. This is an important preliminary task for extracting ADRs from social media, since it can filter out the majority of tweets that mention drugs without any indications of personal intake. We participated with classifiers based on support vector machines (SVMs) and neural networks (NNs), as described in more detail in the Method section.

Data

The organizers provided a training dataset with manually assigned labels (*intake*, *possible intake*, *non-intake*) for each tweet. The *intake* class is defined as clearly expressing a personal intake of medication, whereas the *possible intake* is more ambiguous, yet still suggesting an intake by the tweet writer. The *non-intake* class includes the rest of the tweets, all of which include a mention of a drug, but refer to an intake by another person or discuss the drugs in general. Approximately 50% of the data belongs to the *non-intake* class, whereas the *intake* and *possible intake* classes constitute 19% and 31% of the data, respectively.

Due to the data sharing restrictions of Twitter, the organizers only provided the IDs of the tweets instead of their actual content. Since we started investigating this task much later than the data was released, we were only able to obtain the contents for 7444 tweets out of the total 8000 annotated tweets as some of the content had been already removed by the Twitter users, i.e. we had 7% less training data than teams who were involved in the shared task since the very

*: these authors have contributed equally.

beginning. The organizers also provided a separate development dataset, which consists of 2260 annotated tweets, of which we also lost roughly the same proportion. In the Results and Discussion section we evaluate the impact of the lost data in more detail.

Method

For our baseline approach we form term frequency–inverse document frequency (TF–IDF) weighted sparse bag-of-words (BOW) representations for all given tweets¹. These representations are not only constructed for single tokens, but also token bigrams, trigrams and character n-grams of length 1 to 4. These representations are then fed as features to a linear SVM classifier². The regularization parameter is selected to optimize the micro-averaged F-score of *intake* and *possible intake* classes, the official evaluation metric, on the development set. For the final submission in the shared task we merge the training and development sets and train the system on the combined dataset.

As the sparse representations are not able to generalize well to unseen vocabulary, we also test various NN approaches on this task. The final system is based on an ensemble of convolutional neural networks (CNNs)³ and utilizes word and character information.

Each tweet is represented as two separate sequences: words and characters, both of which are processed with separate convolutional channels. Each element in these sequences is represented with a latent feature vector, i.e. an embedding. The word embeddings are initialized using word2vec⁴ trained with approximately 1 billion drug related tweets as provided by Sarker and Gonzalez⁵. We also tested GloVe vectors trained on 2B general domain tweets⁶, but these experiments resulted in a decreased performance. The character embeddings are initialized randomly, but the network is allowed to backpropagate to both word and character embeddings.

The convolutional kernels are applied on the aforementioned two sequences using sliding windows. The outputs are subsequently max-pooled and concatenated. The concatenated vectors are further fed through two densely connected layers, the latter having the output dimensionality corresponding to the number of labels in the data set with softmax activation.

In addition to the convolutional layers we utilize the same TF–IDF weighted sparse vector representations as in the baseline method. As these representations have dimensionality in the order of hundreds of thousands, we first densify the representations to 4000 dimensional vectors using truncated singular-value decomposition (SVD)⁷. These vectors are concatenated alongside the CNN outputs. This dimensionality reduction is performed mainly due to computational reasons since the approach was prototyped on a consumer grade GPU with limited amount of memory. Projecting the sparse vectors to 4K dimensions preserves 74% of the variance in the data and may have caused a minor performance loss.

The network is trained on the official training data using the Nadam optimization algorithm. The network is regularized with dropout rate of 0.2 after the first dense layer, no explicit regularization is applied on the convolutional part of the network. The training is stopped once the performance on the development set is no longer improving, measured with the official evaluation metric. Table 1 shows the comprehensive list of used hyperparameters.

| Hyper-parameter | Optimal value | Tested Values |
|--|---------------|-------------------------|
| Character embedding dimensionality | 25 | [25,50,75,100] |
| Word embedding dimensionality | 400 | pre-trained |
| Character CNN, number of filters per window size | 50 | [50,100,150,200] |
| Character CNN, window sizes | [2,3,4,5] | [2,3,4,5] |
| Word CNN, number of filters per window size | 200 | [100,200,300] |
| Word CNN, window sizes | [2,4] | any subset of [2,3,4,5] |
| Dimensionality of first dense layer | 400 | [100,200,300,400,500] |
| Dropout rate | 0.2 | [0,0.2,0.5] |
| Activation functions | tanh | [ReLU, tanh, sigmoid] |

Table 1: The optimal and tested hyperparameter values of the CNN-based system.

Training the network on this dataset resulted in relatively large variance in the measured performance, caused by the random initialization of the weights. Thus we stabilize the system by training 15 networks, all identical apart from the initial (random) weights. We then select the optimal subset of these networks, as measured on the development set, for the final system where the final predictions are created by summing the confidences of all selected networks and choosing the label with the highest overall confidence. The final system included a subset of 6 neural networks out of the 15. We note that this approach may potentially overfit on the development set.

Other NN architectures experimented and tested during this shared task include various versions of BiLSTM and attention based networks^{8,9} but none of these experiments resulted in better performance than the CNN architecture described in detail. However, due to the time limits of the shared task, we cannot reject the possibility of these approaches being competitive as well.

We also experimented with a way of (pre-)tuning the utilized word embeddings to this specific classification task in an attempt to give the word-level CNN a better starting point for the training. This was done using the principles underlying the random indexing (RI)¹⁰ method. Unique index vectors are first assigned to each of the three classes (intake, possible intake and non-intake), and empty context vectors are assigned to each word in the data set. When traversing the training set, each word, in each tweet, have the index vector associated with the tweet's class added to their context vector. After training, the resulting word context vectors are normalized to unit length and summed with the corresponding word embeddings/vectors generated using word2vec⁵ (also normalized to unit length). To make the signal provided by the RI approach have a modest impact on the conjoint vectors, these vectors are first multiplied with a weight of 0.3. However, the described approach did not seem to result in a positive performance impact, compared to using the original word2vec generated embeddings.

We also tested the potential benefits of including part-of-speech (POS) tags, which were produced using the Twitter NLP toolkit¹¹. The sequences of POS tags were treated in similar fashion to the word and character sequences. Although the benefits of POS tagging are intuitive as for instance verbs in past tense are twice as common in the *intake* class as in the *possible intake*, we did not see any increase in the performance when POS tags were utilized.

Results and Discussion

We measure the performance of our systems using micro-averaged F-score of *intake* and *possible intake* classes, following the official evaluation, and conduct all our experiments on the official development set. However, the reported results are not directly comparable with other systems as we only had access to a subset of the original data (see the Data section). The results on the test set are as reported by the organizers and thus comparable to other systems.

The overall performance of our baseline (i.e. SVM) and CNN-based systems are relatively strong, resulting in F-scores of 69.6 and 72.7 on the development set respectively (see Table 2). We suspect the main advantage of the CNN approach to be the generalizability of the word embeddings, which leads to the 3.1pp improvement in F-score. We also briefly tested a nonlinear multilayer perceptron with the same BOW features as used in the SVM, which led to a slight improvement over using the linear SVM model, but was not able to outperform our CNN-based system. Thus the model complexity alone does not explain the performance difference between the SVM and CNN approaches.

An unexpected observation is that the *intake* class seems to be harder to predict than the *possible intake*, although eye-balling the data suggests otherwise and the annotation guidelines provide more precise definition for the *intake* class. Also for the CNN-based system it seems that the precision and recall are rather well balanced, thus no performance improvements could have been gained through further fine tuning of these metrics.

The test set results follow the same patterns as the development set evaluation: SVM and CNN systems reach F-scores of 64.2 and 66.3, respectively. Thus it seems that either the test set is somewhat harder than the development set or both of the systems are overfitting equally on the development set, even though the implemented ensemble system with CNNs could have caused greater overfitting. According to the official evaluation, our best system loses to the winning system by 3pp in F-score, the difference being roughly the same in both precision and recall. This places our system in the 5th position in the shared task.

By inspecting the confusion matrices it can be concluded that our classifiers tend to confuse *intake* class with both

| | | Development set | | | Test set | | |
|---------|-----------------|-----------------|--------|---------|-----------|--------|---------|
| | | Precision | Recall | F-score | Precision | Recall | F-score |
| SVM | Intake | 70.5 | 64.5 | 67.4 | | | |
| | Possible Intake | 73.3 | 68.6 | 70.9 | | | |
| | Overall | 72.3 | 67.0 | 69.6 | 69.2 | 60.1 | 64.3 |
| CNN | Intake | 70.9 | 71.3 | 71.1 | | | |
| | Possible Intake | 76.3 | 71.1 | 73.6 | | | |
| | Overall | 74.2 | 71.2 | 72.7 | 70.1 | 63.0 | 66.3 |
| InfyNLP | Overall | | | | 72.5 | 66.4 | 69.3 |

Table 2: Overall performance of our SVM and CNN-based systems. The development set results are measured with our own evaluation whereas the test set scores are as reported by the organizers. The class specific performance was not evaluated by the organizers and has been thus left out from the table. For comparison we have added the results of the best performing team: InfyNLP.

possible intake and *non-intake* classes equally often, whereas the *possible intake* is more often confused with the *non-intake* class.

As we only had access to a partial training data, we try to estimate how much the performance of the systems could have been improved with additional data. To accomplish this, we train the CNN-based system with different subsets of the training data, starting from 5K training examples and incrementally increasing the size in steps of 300 up to the whole training data available to us. After every increment we evaluate the system’s performance on the development set. To reduce the variance caused by different initial random weights, we train 5 networks with each subset of the training data, and calculate the mean performance for each subset. Fitting a linear regression on the resulting measurements shows that in this region, the learning curve is fairly linear and decent performance improvements can be gained by adding more training data. Assuming a performance increase equal to the slope of the fitted regression line, having the full training dataset would have increased our performance by 0.7pp in F-score, placing our system close to the top 3 teams in the shared task.

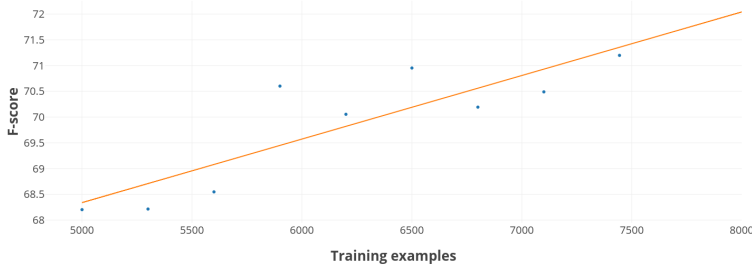


Figure 1: Influence of the number of training examples to the performance of the CNN system as evaluated on the development set.

As most approaches we tested, as well as the systems created by other teams, resulted roughly in the same performance level, we wanted to assess what would be a theoretical performance limit for this task. To this end, we manually annotated a random subset of 100 tweets from the development set and evaluated the annotations against the gold standard. Surprisingly our manual annotations reached only an F-score of 59.3, notably lower than the developed systems or what the official inter-annotator agreement would suggest¹². This indicates that the task is complex even for humans and deep understanding of the annotation guidelines is required for high quality annotations.

Conclusions and Future Work

We have shown that strong results in detecting tweets describing personal medication intake can be achieved using convolutional neural networks and word embeddings. However, more traditional methods relying on bag-of-words features and linear classifiers also result in competitive performance. Considering that such a system can be implemented in less than an hour with the existing tools and libraries, and is easily interpretable, the simpler methods may be a more practical choice in many use cases.

Since the amount of training data for this task is fairly limited, we plan to explore various approaches for pretraining NN classifiers as a future task. The goal here is to find a suitable proxy task related to the domain for initializing the network before the actual training. Such a task could be, for instance, sentiment detection as many of the tweets expressing drug intake also express a certain sentiment about the condition of the user or the effects of the drug.

Acknowledgements

This work was supported by ATT Tieto käyttöön grant and Tekes – Räättäli project (no. 644/31/2015). Computational resources were provided by CSC - IT Center For Science Ltd., Espoo, Finland.

References

1. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. *Information processing & management*. 1988;24(5):513–523.
2. Joachims T. Making large-scale SVM learning practical. In: Schölkopf B, Burges C, Smola A, editors. *Advances in Kernel Methods – Support Vector Learning*. Cambridge, MA: MIT Press; 1999. p. 169–184.
3. LeCun Y, Bengio Y, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*. 1995;3361(10):1995.
4. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems 26*; 2013. p. 3111–3119.
5. Sarker A, Gonzalez G. A corpus for mining drug-related knowledge from Twitter chatter: language models and their utilities. *Data in Brief*. 2017;10(Supplement C):122 – 131.
6. Pennington J, Socher R, Manning C. Glove: global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*; 2014. p. 1532–1543.
7. Halko N, Martinsson PG, Tropp JA. Finding structure with randomness: stochastic algorithms for constructing approximate matrix decompositions. 2009;.
8. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural computation*. 1997;9(8):1735–1780.
9. Luong MT, Pham H, Manning CD. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:150804025*. 2015;.
10. Kanerva P, Kristofersson J, Holst A. Random indexing of text samples for latent semantic analysis. In: *Proceedings of 22nd Annual Conference of the Cognitive Science Society*. Philadelphia, PA, USA; 2000. p. 1036.
11. Owoputi O, O’Connor B, Dyer C, Gimpel K, Schneider N, Smith NA. Improved part-of-speech tagging for online conversational text with word clusters. *Association for Computational Linguistics*; 2013. .
12. Klein A, Sarker A, Rouhizadeh M, O’Connor K, Gonzalez G. Detecting personal medication intake in Twitter: an annotated corpus and baseline classification system. In: *BioNLP 2017*. Vancouver, Canada.; Association for Computational Linguistics; 2017. p. 136–142.

Farrokh Mehryary, Jari Björne, Tapio Salakoski, Filip Ginter
Potent pairing: ensemble of long short-term memory
networks and support vector machine for chemical-protein
relation extraction

Database: the journal of biological databases and curation, 2018, bay120





Original article

Potent pairing: ensemble of long short-term memory networks and support vector machine for chemical-protein relation extraction

Farrokh Mehryary^{1,2,*}, Jari Björne^{1,3}, Tapio Salakoski^{1,3} and Filip Ginter¹

¹TurkuNLP group, Department of Future Technologies, University of Turku, Turku, Finland, ²University of Turku Graduate School, Turku, Finland and ³Turku Centre for Computer Science, Turku, Finland

*Corresponding author: Tel: +358 2 333 7649; Email: farmeh@utu.fi

Citation details: Mehryary, F., Björne, J., Salakoski, T. *et al.* Potent pairing: ensemble of long short-term memory networks and support vector machine for chemical-protein relation extraction. *Database* (2018) Vol. 2018: article ID bay120; doi:10.1093/database/bay120

Received 26 February 2018; Revised 5 October 2018; Accepted 7 October 2018

Abstract

Biomedical researchers regularly discover new interactions between chemical compounds/drugs and genes/proteins, and report them in research literature. Having knowledge about these interactions is crucially important in many research areas such as precision medicine and drug discovery. The BioCreative VI Task 5 (CHEMPROT) challenge promotes the development and evaluation of computer systems that can automatically recognize and extract statements of such interactions from biomedical literature. We participated in this challenge with a Support Vector Machine (SVM) system and a deep learning-based system (ST-ANN), and achieved an F-score of 60.99 for the task. After the shared task, we have significantly improved the performance of the ST-ANN system. Additionally, we have developed a new deep learning-based system (I-ANN) that considerably outperforms the ST-ANN system. Both ST-ANN and I-ANN systems are centered around training an ensemble of artificial neural networks and utilizing different bidirectional Long Short-Term Memory (LSTM) chains for representing the shortest dependency path and/or the full sentence. By combining the predictions of the SVM and the I-ANN systems, we achieved an F-score of 63.10 for the task, improving our previous F-score by 2.11 percentage points. Our systems are fully open-source and publicly available. We highlight that the systems we present in this study are not applicable only to the BioCreative VI Task 5, but can be effortlessly re-trained to extract any types of relations of interest, with no modifications of the source code required, if a manually annotated corpus is provided as training data in a specific file format.

Database URL: https://github.com/TurkuNLP/BioCreativeVI_CHEMPROT_RE

Introduction

BioCreative VI Task 5 challenge (hereinafter referred to as the ‘shared task’), focuses on extraction of relations between chemical compounds/drugs and genes/proteins, stated in biomedical texts (1). The CHEMPROT corpus that provides such annotations is used as the training and test data in this task. The aim of the task is to promote the development of systems for extracting such relations for use in precision medicine, drug discovery and basic biomedical research (<http://www.biocreative.org/tasks/biocreative-vi/track-5/>).

This shared task follows the well-established approach of pairwise relation extraction in the field of biomedical text mining. protein–protein interactions (PPI) were one of the extraction targets in a number of shared tasks and datasets. The BioCreative II and BioCreative III challenges (2, 3) focused on pure PPI extraction, while the BioCreative V CDR Task focused on chemical-induced disease relation extraction (4). The two drug–drug interaction shared tasks (DDI-2011 and DDI-2013) focused on the detection of adverse interactions between pairs of drugs (5, 6), and the Bacteria–Biotope relation extraction tasks aimed at extracting the location of bacteria from scientific web pages or PubMed abstracts (7–9). Finally, Pyysalo *et al.* (10) have preprocessed and unified five publicly available protein–protein interaction corpora (<http://mars.cs.utu.fi/PPICorpora/>), in order to facilitate seamless development and comparison of biomedical relation extraction methods. Among these tasks, DDI-2013 (6) has become popular for assessing the performance of relation extraction methods, mainly because it has a relatively large and challenging corpus.

We approach the BioCreative VI Task 5 challenge as a classification task where we classify each valid pair of entities as one of the annotated relation types or as a negative. We have developed three different systems to address the task. The first system relies on a rich set of features and a linear support vector machine (SVM) classifier (11). The two other systems are based on deep learning and require less feature engineering. Our shared task artificial neural network (ST-ANN) system utilizes an ensemble of neural networks, each having three long short-term memory (LSTM) chains (12), for representing the words, part-of-speech (POS) tags and dependency types (DTs) (i.e. edges in the sentence parse graph) along the shortest dependency path (SDP) connecting the two candidate entities. Our improved ANN (I-ANN) system is also an ensemble of neural networks, each having three LSTM chains for representing the words, POS tags and DTs along the SDP (similar to the ST-ANN) and a bidirectional LSTM (forward and backward chains) for learning a representation of the whole sentence and the two entities of interest in it. We have

also experimented with several methods for combining the predictions of these systems, with the goal of increasing the overall performance.

We participated in the shared task with the SVM and ST-ANN systems (13). On the development set, our system combination approach outperformed the two individual systems, achieving an F-score of 61.09. On the test set, our SVM system achieved the highest result of our submissions with an F-score of 60.99. After the shared task, we have significantly improved the performance of the ST-ANN system. In addition, we have developed the I-ANN system, which considerably outperforms the ST-ANN system. Finally, by combining the predictions of the SVM and I-ANN systems, we achieved an F-score of 61.46 on the development set, with a corresponding F-score of 63.10 on the test set, 2.11 percentage points (pp) higher than our best test set submission during the shared task. Here we discuss all approaches and results in detail.

Background

In all the aforementioned biomedical relation extraction tasks (including BioCreative VI Task 5), the named entities are manually annotated and given as known data to the participants, hence the aim is to build methods that are able to automatically detect statements of relations among known named entities in the given texts. In addition to the named entities, the training data for these tasks also include manually annotated relations, making these tasks ideal for the development of *supervised* relation extraction methods. These machine learning-based methods utilize the provided training data to train a classifier—e.g. an SVM, an ANN or a Naive Bayes classifier—capable of detecting statements of relations in texts.

According to Zhang *et al.* (14), supervised relation extraction methods can be broadly divided into three main groups: (i) feature-based methods, (ii) kernel-based methods and (iii) deep learning-based methods.

Feature-based methods extract a series of relevant features from the text in order to train a relation extraction classifier. In these methods, each entity pair is represented with a corresponding numerical feature vector that is further used for either training the classifier or for detection of the relation(s) (14). The list of features usually includes (but is not necessarily limited to) bags-of-words/lemmas/POS/DTs or their n-grams in the sentence or along the SDP. The Turku Event Extraction System (TEES) (15)—previously developed by members of our research group—is an example of such a system, using a rich set of features to build an SVM classifier. TEES achieved 62.99 F-score in the DDI-2011 task (5), 58.7 F-score in the DDI-2013 task (6) and the state-of-the-art performance

(42.00 F-score) in the Bacteria-Biotope 2013 relation extraction task (8). Another example is the VERSE system, developed by Lever and Jones (16), which obtained the state-of-the-art result with an F-score of 55.8 in the Bacteria-Biotope 2016 relation extraction task (9). Similarly to TEES, VERSE also extracts a rich set of features in order to train a linear SVM, but utilizes a feature selection component for optimization. Finally, Raihani *et al.* (17) achieved the impressive F-score of 71.14 on the DDI-2013 corpus with a system utilizing lexical, phrase, verb, syntactic and auxiliary features.

Kernel-based methods use kernel functions that are able to directly calculate the similarity between two instances (i.e. two machine learning examples) to train a relation classification model (14, 18). In kernel methods, examples retain their original representation (e.g. as bag-of-words in the sentence, sentence dependency parse graph or sentence shallow parse graph) and the kernel method is able to assign a label to a given novel example by computing and comparing its similarity to all labeled training set examples (14, 18, 19). An advantage of kernel methods is that they can search a feature space much larger than could be represented by a feature-based approach, because the kernel functions can explore an implicit feature space when calculating the similarity between two examples (19). Kernel functions are usually used in conjunction with classifiers like SVM and voted perceptron (20). Several kernel functions have been suggested and applied for relation extraction. In a bag-of-features kernel approach, the words in the sentence are divided into three groups: before, between and after the two entities. Each group is further represented with a bag-of-features. Bunescu *et al.* (21) used this approach to build three subsequence-kernels for each bag, with the final kernel function being simply the sum of the three kernels, which is further used with an SVM classifier for relation classification. Another popular family of kernels are tree/graph kernels. Zelenko *et al.* (18) developed kernels capable of comparing the similarity of shallow parse trees and used them with SVM and voted perceptron classifiers for relation extraction. Culotta *et al.* (19) extended the previous work by introducing the 'Dependency Tree Kernel' for relation extraction and showed that their model outperforms bag-of-words kernel approach by 20 pp. Reichartz *et al.* (22) developed the 'All-Pairs Dependency Tree Kernel', and the 'Dependency Path Tree Kernel' and showed their kernels with richer structural features significantly outperform all published approaches for kernel-based relation extraction from dependency trees. Finally, Airola *et al.* (23) developed the 'All-Paths Graph Kernel' for biomedical relation extraction and showed that their method achieves the state-of-the-art performance on five protein-protein interaction corpora.

Feature-based methods extensively rely on natural language processing (NLP) tools (e.g. tokenizers, POS taggers, lemmatizers, syntactic parsers, etc.) and require heavy feature engineering to transform the input data into a 'representation' (i.e. a feature vector) that can lead to a successful relation classification. On one hand, feature engineering is skill-dependent and time-consuming (24), on the other hand, the errors in the NLP tools are amplified in the relation extraction systems, negatively impacting their performance (14). In contrast, the aim in deep learning approach is to 'automatically learn' efficient representations, suitable for the relation classification task at hand. Deep learning achieves this by introducing representations that are expressed in terms of other, simpler representations and allowing the computer to automatically learn complex concepts out of simpler concepts (25). For example, the concept of a sentence can be expressed by phrases, while phrases are composed of words and syntactical dependencies among them. This allows a modular design and training of a hierarchy of representations, with the root as the final representation used for a prediction task. A key feature is that lower-level representations (i.e. 'embeddings') can sometimes be pre-trained in advance, in an unsupervised fashion and with training data other than the training data available for the prediction task at hand. A successful example is pre-trained 'word embeddings', the vector representations for words in a language that are trained on millions of unannotated sentences, so that words with similar meanings have similar corresponding vectors in the vector space model (26). Several studies have shown that integrating pre-trained word embeddings into deep neural networks (DNNs) can improve the performance of downstream prediction tasks.

Deep learning-based relation extraction methods have recently outperformed feature/kernel-based methods on different corpora. For example, on the DDI-2013 corpus (6) all top performing methods are based on DNNs (24). The only exception is the feature-based system of Raihani *et al.* (17) with 71.1 F-score, on par with the recent deep learning-based methods. recurrent neural networks (RNN) and convolutional neural networks (CNN) are the two main neural structures that are extensively utilized in DNNs for achieving state-of-the-art performance in various NLP and text mining tasks, such as syntactic parsing, sentence classification, sentiment analysis, text summarization, machine translation, named-entity recognition and relation extraction. CNNs are inherently efficient in learning 'local' or 'position-invariant' features through discrete convolution with different size filters (kernels), because they extract the features based on n -grams of the sentences. In contrast, RNNs can directly model sequential data, such as the sequence of

words in sentences (24). LSTM networks (12) and gated recurrent units (GRUs) (27) are variants of RNNs that utilize memory cells and/or gating mechanisms to deal with the vanishing or exploding gradients (28), a problem associated with RNNs which negatively impacts their training and prediction performance.

Yin *et al.* (29) have systematically compared the performance of CNNs with LSTMs and GRUs on various NLP tasks and have shown that the performance of CNN and LSTM/GRU networks are very close for relation extraction on the SemEval-2010 corpus (30). However, literature survey on the DDI-2013 corpus shows that at the moment, the top three methods on this corpus are based on RNNs with F-scores higher than those of CNN-based methods. Lim *et al.* (31) have achieved the state-of-the-art F-score of 73.5 with an ensemble of Tree-LSTMs; Zhou *et al.* (32) have achieved 73.0 F-score with position-aware attention-based bidirectional LSTM networks and multitask learning; and Zhang *et al.* (24) have achieved 72.9 F-score using hierarchical bidirectional LSTM networks. In contrast, the ‘dependency-based CNN’ developed by Liu *et al.* (33) has achieved 70.8 F-score, the ‘multichannel CNN’ developed by Quan *et al.* (34) has achieved 70.21 F-score and the ‘Syntax CNN’ (SCNN) developed by Zhao *et al.* (35) has achieved 68.6 F-score. According to Zhang *et al.* (24) the main reason is that the DDI-2013 corpus contains many long and complicated sentences, and compared to CNNs, RNN-based models can better learn the long-term dependence of the sentence that is crucial for capturing the lexical and syntactic features in the long and complicated sentences. Since CNNs work based on n -grams, they can encounter problems in learning from long sentences or sentences that have important clues lying far away from each other. However, we highlight that the performance of a neural relation extraction system does not boil down only to the neural network architecture it uses, but also the inputs it receives, the feature set that it uses and the training and optimization procedures that are used to train the system.

On the CHEMPROT corpus, the highest F-score (64.10) in the shared task has been achieved by Peng *et al.* (36), with a system combination approach. Their method is an ensemble of three separate systems: (i) a CNN-based relation extraction system that receives the sentence sequence and the SDP sequence as inputs, (ii) an RNN-based system that utilizes a bidirectional LSTM network to learn from the full sentence sequence and (iii) an SVM-based system that generates features based on the full sentence and SDP. This suggests that combining the power of neural models with feature-based methods is a promising approach for relation extraction. We also participated in the shared task with a system combination approach, utilizing an SVM-based system and a deep learning-based system (13). After the

shared task we have improved our neural network models, hence improving our best F-score by 2.11 pps.

Data

The CHEMPROT corpus is a pairwise relation dataset. All entities are given as known data to the participants, thus the task is to predict the relations for valid pairs of these entities. The relations are directed, always connecting a GENE-type entity (gene or protein) to a CHEMICAL-type entity. A large set of distinct types are used for annotating the relations, but these types are combined into 10 groups that are used as the actual classes for this task. Further, only five of these classes are taken into account in the task evaluation. The micro-averaged F-score of the five target classes is the official metric used for evaluation.

Cross-sentence relations constitute less than 1% of the total relations in the CHEMPROT training set. In addition, only 10 pairs in the training set have been labeled with multiple relation types. Hence, we formulate the task as a multi-class classification task where we classify each valid pair of entities as 1 of the 10 annotated relation types or as a ‘negative’, and we only focus on candidate pairs belonging to the same sentence.

Methods

We develop three different systems capable of extracting relations between CHEMICAL and GENE entities. Our first system relies on a rich set of features and a linear SVM classifier (11). The two other systems are based on deep learning and require less feature engineering. Our ST-ANN system has been developed during our participation in the shared task, whereas we have developed the I-ANN system after the shared task. We also combine predictions of the SVM classifier with either ST-ANN or I-ANN predictions to boost the F-score, using a simple algorithm that is optimized on the official development set. In this section we discuss the details of each approach.

Preprocessing

We use the TEES system (15) to run a preprocessing pipeline of tokenization, POS tagging and parsing. We convert the CHEMPROT corpus into the Interaction XML format native to the TEES preprocessing system. We test different parses generated using the TEES preprocessor wrappers for the BLLIP, Stanford converter and SyntaxNet parser software (37–39). The default parsing pipeline in our experiments consists of BLLIP constituency parsing with the biomedical domain model of McClosky (40), followed by conversion to dependencies using the Stanford conversion

tool (38). We test different variants of the Stanford dependencies (SDs) representation, with the ‘CCprocessed’ variant being the default unless otherwise stated.

The training data incorporates 10 different types of relations, five of them being evaluated in the task. We also define and add a ‘negative’ type for the cases where no relation exists between the two candidate entities. Hence, we formulate this relation extraction task as an 11-class classification problem.

SVM-based system

The SVM-based system used in this work is the TEES (15). The system is applied as is, with no task-specific modifications. The TEES system uses the SVM^{multiclass} software as the multi-class classifier implementation (41).

The TEES system has primarily been developed for the detection of ‘events’ (42), a more complex alternative to pairwise relation annotations like those used in the CHEMPROT corpus. Events consist of a trigger word (usually a verb) and 0-n directed arguments that can be named entities like proteins, but also other events. In this manner, events form a complex graph where the named entities and triggers are the nodes, and the event arguments are the edges. For detecting events, the TEES system is built as a pipeline of consecutive classification steps. In the first step (entity detection), each word token in the sentence is classified as a trigger or not. In the second phase (edge detection), directed edges are predicted between all valid pairs of entities. Since multiple, different events can use the same word token as their trigger, the third step (unmerging) is used to ‘pull apart’ such nodes by classifying all valid trigger and argument combinations as real events or not. The fourth step (modifier detection) can be used to detect binary modality modifiers annotated for some events, such as negation and speculation.

TEES can also be used for pairwise relation extraction tasks such as the DDI (drug–drug extraction) challenges (43) or in the current work, the CHEMPROT task. In such tasks, only the second step (edge detection) of the TEES pipeline is used. The set of nodes consists of the given named entities, and relations are predicted for all valid node pairs; in the case of CHEMPROT all pairs where the first entity is of type CHEMICAL and the second of type GENE.

For all the steps in the classification pipeline, TEES relies on a rich feature representation. While most features for relation detection are generated from the shortest path of dependencies between the two entities, dependency chains outside this shortest path, bags-of-words and the linear order of tokens are also used for generating features, in an attempt to capture more of the sentence context outside the direct relation between the two entities of interest.

We test several different forms of parsing and variations of predicting the CHEMPROT corpus with TEES, but find that none of these improve performance over the default approach (see the Shared task results section for detailed information). In addition we tried using the DrugBank dataset (44) as additional features. For the three CHEMPROT corpus representations, the TEES system is trained with either the default of all 10 classes, with all the non-evaluated classes merged into a single class or with the non-evaluated classes entirely removed. For parses, we try the BLLIP parser with the McClosky biomodel and with all five types of Stanford conversion, as well as the SyntaxNet parser.

ST-ANN system

ST-ANN is a deep learning-based relation extraction system that requires less feature engineering than the SVM system and is centered around three main ideas: (i) utilizing LSTM networks instead of simple RNNs, (ii) focusing on the words along the SDP and (iii) using an ensemble of neural networks (with identical architectures) instead of a single neural network, to stabilize the variance in the performance caused by the random initialization of the network weights.

The ST-ANN system has an architecture similar to the successful system we have recently developed for extracting bacteria–habitat relations from biomedical texts (45), but the predictions of the networks in this ensemble are aggregated using a different approach, more suitable for multi-class classification. Each neural network in the ST-ANN ensemble utilizes three separate LSTM chains for representing the words, representing the POS tags and representing the DTs (i.e. edges in the parse graph) along the SDP that connects the two entities. Figure 1 shows the architecture of one neural network in the ST-ANN ensemble with an example sentence from the CHEMPROT training set and its dependency parse graph as the inputs.

Even though standard RNNs are theoretically efficient sequence learning models, they usually suffer from the vanishing or exploding gradients problem (28): if the network is deep, during the back-propagation, the gradients may either decay exponentially and cause the learning to become very slow or stop altogether (‘vanishing gradients’); or become excessively large, and cause the learning to diverge (‘exploding gradients’). To address this problem, LSTM networks (12) and GRUs (27) have been proposed based on RNNs. LSTM-based networks exploit memory cells and gating mechanisms while GRU-based models are simpler and only utilize a gating mechanism. LSTM and GRU networks are shown to be much more efficient sequence modelers compared to simple RNNs. For example, Zhang *et al.* (24) have compared standard RNNs with

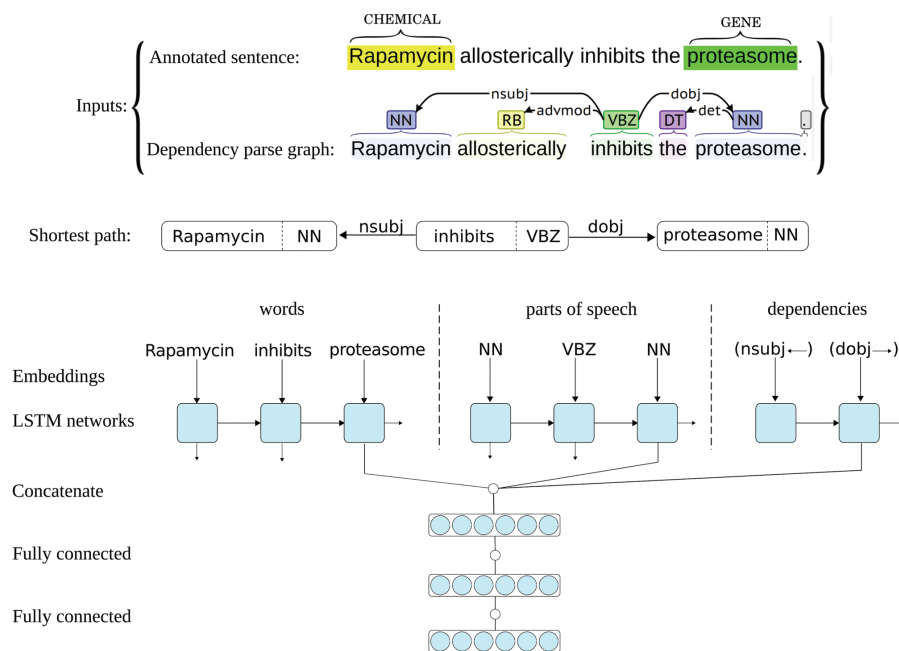


Figure 1. Architecture of one neural network in the ST-ANN ensemble. The figure illustrates the architecture of one neural network in the ST-ANN ensemble with an example sentence from the CHEMPROT training set and its dependency parse graph as the inputs. The shortest dependency path which connects the two entities ("Rapamycin" and "proteasome") in the parse graph is first discovered. The path is traversed from the CHEMICAL entity to the GENE entity, producing the sequence of words, the sequence of POS tags, and the sequence of dependency types (edges) along the path. The words, POS tags and dependency types are then mapped into their corresponding vector representations using embedding lookup layers and then input to three separate LSTM chains. The outputs of the last LSTM units of the three chains are concatenated together and the resulting higher dimensional vector (i.e. the SDP vector representation) is input to a hidden dense layer. The hidden layer finally connects to the decision (classification) layer, which has a softmax activation.

LSTMs and GRUs for relation extraction and have shown that their standard RNN-based model achieves 61.4 F-score on DDI-2013 corpus, whereas the equivalent GRU-based and LSTM-based models achieve 72.4 and 72.9 F-score, respectively, ~11 pps higher than the standard RNN-based model. In this work, we also use LSTM networks instead of standard RNNs, for capturing the information in the SDP.

The SDP that connects the two entities in the syntactic parse graph is known to contain most of the relevant words for expressing the relation between the two entities, while excluding less relevant and uninformative words (45). Figure 1 shows an example sentence from the CHEMPROT training set, its parse graph and the shortest path that connects the two entities. As can be noticed in the figure, the SDP for this particular example contains only the subject 'Rapamycin' (the CHEMICAL), the verb 'inhibits' and the direct object 'proteasome' (the GENE), whereas less

relevant words (the adverb 'allosterically' and the article 'the') are put aside. Building a relation extraction system by focusing on the most important words (e.g. words in the SDP) can lead to good generalization for unseen data. Based on this observation, many successful feature-based and deep learning-based relation extraction methods have been developed (15, 16, 45–49). The ST-ANN system also generates the features along the SDP. For this aim, we first assume the parse graph is undirected and find the shortest path between the two entities (CHEMICAL and GENE), and always traverse the path from the CHEMICAL entity to the GENE entity, regardless of the order of the entity mentions in the sentence. Based on experiments on the CHEMPROT development set, we notice this approach results in significantly better generalization for unseen data, compared to traversing the path from the first occurring entity mention in the sentence to the second. Besides the

existing DT edges in the parse graph, we also add an artificial edge between any two adjacent words of the sentence (word-adjacency edges). As shown by Quirk *et al.* (50), this approach mitigates the parsing errors and increases accuracy and robustness when the system is confronted with linguistic variation. For instance, if the parser produces a graph with more than one connected component, adding these artificial edges to the parse graph assures the existence of a path between the two entities. We assign a distance (weight) of one to DT edges and the distance assigned to word-adjacency edges is treated as a hyperparameter, set (to the value of 5) using the grid-search optimization procedure described later. We then generate the features based on the words, POS-tags and DTs in this path.

As Figure 1 shows, each neural network in the ST-ANN system utilizes three separate LSTM chains. The sequences of words, POS tags and DTs are first mapped into sequences of their corresponding vector representations, i.e. embeddings, by three separate embedding lookup layers and then used as input to the three LSTM chains. For words, we use 200-dimensional pre-trained word embeddings provided by Pyysalo *et al.* (51), which have been trained on the texts of all PubMed titles and abstracts and PubMed Central Open Access (PMC-OA) full text articles using the word2vec method (26), whereas POS tag and DT embeddings are initialized randomly at the beginning of the training. During the training of our system, word embeddings are fine-tuned while the randomly initialized POS and DT embeddings are learnt from scratch. The outputs of the last LSTM units of the three chains are concatenated together, the resulting higher-dimensional vector (i.e. the SDP vector representation) is fed to a fully connected hidden layer. The hidden layer finally connects to the decision layer, having an output dimensionality of 11 (corresponding to the number of classes in the dataset, plus one for the ‘negative’ class), with the softmax activation.

We train the aforementioned neural network on the official CHEMPROT training data and evaluate it with the official evaluation script—provided by the organizers—on the official development data. At the beginning of training, all neural network weights (except for the pre-trained word embeddings) are randomly initialized. After the training, we notice a slight variation in the measured F-score (~ 1 pp) based on different initial random weights. In other words, if we repeat training the neural network with the ‘exact’ hyperparameter values, and over and over again, the performance of the trained models on the development set vary in the range of 1 pp. To stabilize the variance in the performance caused by the random initialization of the network weights, we train an ensemble of four neural

networks (instead of a single neural network), all identical apart from the initial (random) weights, and aggregate their predictions. Each network predicts a set of confidences for each development/test set example. The final prediction for an example is generated by summing the confidences of all networks and selecting the label with the highest overall confidence. We highlight that this particular ensemble method does not automatically improve the overall F-score, but stabilizes the performance of the ensemble (regardless of the initial random weights in each network). In other words, the ensemble acts like an average neural network, but robust and indifferent to the initial random weights used to train the individual networks. Even though the 1% variation observed in the F-scores does not seem especially excessive, we use the ensemble method for the following reasons. Firstly, the ensemble method facilitates hyperparameter optimization (i.e. finding optimal values for the hyperparameters), because it ensures the same performance level can be achieved (on the development set) if the ensemble is re-trained using the same hyperparameter values. This helps to make sure the improvements with values of < 1 pp in the F-score are actually due to the chosen values for the hyperparameters and not caused by a random initialization of the weights, thus allowing us to fine-tune the hyperparameters. Secondly, as we discussed previously, our relation extraction system is not specific to the CHEMPROT corpus and can be re-trained with other training data (e.g. the DDI-2013 or the Bacteria-Biotope corpora) for other biomedical relation extraction applications. Our previous experiments with similar neural network-based relation extraction methods and different corpora indicate that when the number of weights in a neural network is high and the training set is very small, the initial random state of a model can have a significant impact on the final model and its generalization performance, thus special care is needed when dealing with such datasets. For example, the Bacteria-Biotope 2016 corpus (9) contains only 524 relations in the training set. We have previously shown that the F-score on the development set of this corpus can vary up to 9 pps based on the different initial random state of the network (45). Training an ensemble of networks—instead of a single network—helps to reduce the variance when our system is trained/optimized on different corpora for different real-world applications.

We optimize the following hyperparameters with a grid search and repeating the cycle of training the ensemble (with a set of selected hyperparameters) and evaluating it on the development data: word-adjacency edge weight, dimensionality of the POS and DT embeddings, output dimensionality of the LSTMs and the dense layer, activation functions, dropout rate, learning rate and mini-batch size.

For training we use the Nadam optimization algorithm, with a learning rate of 0.002 and mini-batch size of 32, values found to be optimal by the grid search. Similarly, we apply a dropout (52) with 0.2 rate on the output of the first dense layer. The dropout is the only explicit regularization method used. Finally, we use the early stopping technique to obtain the optimal number of training epochs: the training is stopped once the performance on the development set is no longer improving, measured using the official evaluation metric.

I-ANN system

The I-ANN system—also an ensemble of four ANNs—has been developed after our participation in the shared task. The neural networks in the I-ANN ensemble have an architecture similar to the networks in the ST-ANN ensemble (i.e. each neural model utilizes three LSTM chains for representing the sequence of words, POS tags and DTs along the SDP), but a bidirectional LSTM (forward and backward chains) is also added to the architecture for learning a representation of the full sentence and the two entities of interest in it.

A literature survey shows that in one comparison perspective, relation extraction methods can be divided into three groups: (i) the methods that only rely on the SDP, (ii) the methods that process either full sentence tokens or the entire parse graph (e.g. graph kernels or Tree-LSTMs), not explicitly targeting and extracting features from the SDP, and (iii) mixed methods that simultaneously process full sentence tokens and the SDP and generate two distinct sets of features (two vector representations) from them. Recently, the mixed methods have become popular, performing efficiently on various relation extraction datasets. For example, on the Bacteria-Biotope 2016 relation extraction corpus (9), the state-of-the-art performance has been achieved by the VERSE system, a feature-based relation extraction system developed by Lever and Jones (16) that generates features based on the word/POS tag/DT n -grams in the full sentence and also in the SDP, as well as features that are generated from the two candidate entities and paths around them in the parse graph. On the DDI-2013 corpus (6), an impressive F-score of 72.9 has been achieved by Zhang *et al.* (24) with attention-based bidirectional LSTM networks that process the sequence of words in the sentence, as well as the sequence of words in the SDP. They first divide the sentence into three sub-sequences: the words before the first entity mention, the words between the two entity mentions and the words after the second entity mention. The three sub-sequences and the SDP sequence are processed by four attention-based bidirectional LSTM chains that learn the representation for each sequence. The

resulting representations are then processed with an upper-bidirectional LSTM network that learns the representation of the full sentence and the SDP. On the CHEMPROT corpus, the highest F-score in the shared task (64.10) has been achieved by Peng *et al.* (36) with a system combination approach. Their method is composed of three separate systems: (i) a CNN-based relation extraction system with separate convolutional layers that simultaneously learn SDP representation and full sentence representation, (ii) an RNN-based relation extraction system that utilizes a bidirectional LSTM network and max pooling to learn full sentence representation and (iii) an SVM-based system that generates features based on the SDP and the full sentence. Based on recent successful works, we believe that learning two separate representations (for the SDP and for the full sentence) increases the classification performance of the relation extraction methods. Particularly in the case of neural models, since the SDP vector representation and the full sentence vector representation are usually concatenated together and input into subsequent layers, the neural models learn how to effectively integrate these two vector representations for the relation extraction task at hand. Inspired by the work of Zhang *et al.* (24) and the CNN-based system of Peng *et al.* (36), we propose the I-ANN system that simultaneously learns SDP vector representation and full sentence vector representation. For learning the SDP vector representation, we use the same architecture of the ST-ANN system and for learning the full sentence vector representation, we use a ‘bidirectional’ LSTM and max pooling, similar to the RNN-based system of Peng *et al.* (36).

All RNN architectures (standard RNNs, LSTMs and GRUs) use backward connections: assuming a sentence has n words ($W_1, W_2, \dots, W_{t-1}, W_t, \dots, W_n$), the output of the network at time-step t is a function of the input at time-step t and the output (and/or hidden state) of the network at time-step $t-1$, meaning they only capture information from the past words and the current word in the sentence. However, in many applications we want the output (i.e. representation) for a word that is dependent on whole input sequence, i.e. the context before and after that word. For example, in speech recognition, if there are two interpretations of the current word that are both acoustically plausible, we may have to look far into the future (and the past) to disambiguate them (25). For this reason, ‘bidirectional’ recurrent networks (53) and their variants (e.g. bidirectional LSTM/GRU networks) are introduced. In these networks, two separate RNNs simultaneously process the sequence, but from opposite directions, resulting in a forward and a backward representation for each word. The two representations for each word are further aggregated (e.g. by taking the sum or concatenation), and

the aggregation is used as the final representation of the word based on the past and the future words in the sentence. As suggested by Zhang *et al.* (24) and Peng *et al.* (36), and also based on our own experiments on the CHEMPROT development data, utilizing a bidirectional LSTM (instead of a forward LSTM) results in better representations for the sentences, reflected in achieving higher performances for the relation extraction tasks at hand. Consequently, we also use a bidirectional LSTM for modeling the full sentence and the two entities in it. Figure 2 shows the architecture of one neural network in the I-ANN ensemble with an example sentence from the CHEMPROT training set and its dependency parse graph as the inputs.

As Figure 2 shows, each neural network in the I-ANN ensemble utilizes three LSTM chains for learning SDP vector representation and two LSTM chains (forward and backward) for learning full sentence vector representation. Similar to the ST-ANN system, the words, POS tags and DTs along the SDP connecting the CHEMICAL entity to the GENE entity are mapped into their corresponding vector representations (embeddings) and input to the three SDP LSTM chains. Simultaneously, for each token of the sentence, its word, POS tag, relative position to the first entity, relative position to the second entity and token type are mapped into their embeddings and concatenated. Forward and backward sequences of the resulting token representations are input to the forward and backward sentence LSTM chains, resulting in two hidden representation for each token (forward and backward), which are further concatenated to obtain final representations of the sentence tokens. Applying max-over-time pooling on these representations produces a vector representation for the full sentence. The outputs of the last LSTM units of the three SDP chains and the full sentence vector representation are concatenated together and the resulting higher-dimensional vector is input to a fully connected hidden dense layer. The hidden layer finally connects to the decision (classification) layer, which has a softmax activation.

Similar to the ST-ANN system, for words, we use the same 200-dimensional pre-trained word embeddings provided by Pyysalo *et al.* (51). It should be mentioned that these embeddings are pre-trained on the ASCII-fied PubMed and PMC-OA texts. Since some CHEMICAL/GENE entity mentions in the CHEMPROT corpus include unicode characters (e.g. $\text{I}\kappa\text{B}\alpha$, M6PR Δ C, IL-1 β , 11 β -HSD1, AMPK α , to name a few), they do not have any corresponding vectors in the vector space model. Besides, we use only the top 1 million most frequent words from the vector space model. We thus replace all GENE entity names with the word 'protein' and all CHEMICAL entity names with the word 'chemical', if the entity name cannot be found in the model. This resulted in ~ 0.5 pp increase in the F-score (when

different neural network models were evaluated on the official development set).

Similar to Zhao *et al.* (35), the relative position of each token to the first and second occurring entities are first calculated and then non-linearly mapped to their corresponding 10-bit binary vectors (embeddings), where the first bit of each vector stands for the sign and the remaining bits for the distance. Additionally, inspired by the idea of 'named entity embeddings' introduced by Peng and Lu (54), for each token of the sentence, a token type (one of the following values) is assigned accordingly:

1. If the token belongs to a CHEMICAL entity mention.
2. If the token belongs to a GENE entity mention.
3. If the token is located before the first occurring entity in the sentence.
4. If the token is located between the two occurring entities in the sentence.
5. If the token is located after the second occurring entity in the sentence.

The word and POS tag embeddings are shared among the SDP LSTM and the full sentence LSTM chains. During training, the pre-trained word embeddings are fine-tuned, while POS, DT, position to the first entity, position to the second entity and token-type embeddings are all learned from scratch.

Similar to the ST-ANN system, for stabilizing the variance in the performance of the I-ANN system (caused by the random initialization of the network weights), we train an ensemble of four neural networks, all identical apart from the initial (random) weights, and aggregate their predictions using the aforementioned aggregation method. Finally, we optimize the network hyperparameters by doing a grid search and repeating the cycle of training an ensemble (with a set of selected hyperparameters) and evaluating it on the development data. Table 1 shows a comprehensive list of the hyperparameters, the list of values that are tested and the optimal values that have been found and selected to build the final neural model. For example, for the dimensionality of the POS tag embeddings, we tested the values 25, 50, 75 and 100, and 25 was shown to be the best value and thus was selected to build the final model. Similarly, we tested not using a hidden dense layer at all, or using a hidden layer with the output dimensionality of 300, 500 or 1024, and it was shown that using the additional hidden dense layer with 1024 output dimensionality leads to better performance.

System combination

Our SVM and the two deep learning-based systems are trained with different sets of features. This is a potential

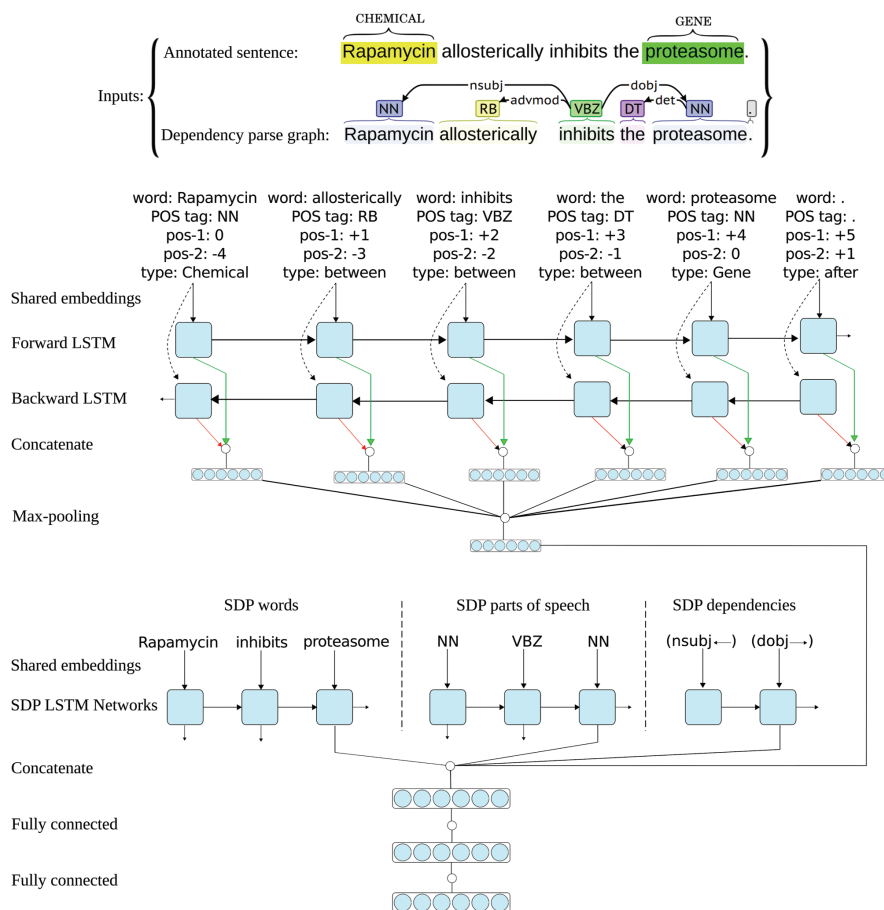


Figure 2. Architecture of one neural network in the I-ANN ensemble. The figure illustrates the architecture of one neural network in the I-ANN ensemble with an example sentence from the CHEMPROT training set and its dependency parse graph as the inputs. The model utilizes three LSTM chains for learning SDP vector representation and two LSTM chains (forward and backward) for learning full sentence vector representation. The words, POS tags and dependency types along the SDP connecting the CHEMICAL entity ("Rapamycin") to the GENE entity ("proteasome") are mapped into their corresponding vector representations (embeddings) and input to the three SDP LSTM chains. Simultaneously, for each token of the sentence, its word, POS tag, position to the first entity, position to the second entity, and token-type are mapped into their embeddings and concatenated. Forward and backward sequences of the resulting token representations are input to the forward and backward sentence LSTM chains, resulting into two hidden representation for each token (forward and backward), which are further concatenated to obtain final representations of the sentence tokens. Applying max-over-time pooling on these representations produces a vector representation for the full sentence. The outputs of the last LSTM units of the three SDP chains and the full sentence vector representation are concatenated together and the resulting higher dimensional vector is input to a hidden dense layer. The hidden layer finally connects to the decision (classification) layer, which has a softmax activation. The word and POS tag embeddings are shared among the SDP and the full sentence LSTM chains.

case for testing whether combining predictions of the two systems (SVM with either of ST-ANN or I-ANN) could help in achieving better performance for this task. We

implement this system combination by merging the relation predictions from the two systems as either a union (OR) or an intersection (AND), resolving overlapping predictions

Table 1. Hyperparameters of the networks

| Hyperparameter groups | Hyperparameters | Values | |
|--------------------------------|-------------------------------------|---------------|---|
| | | Optimal value | Tested values |
| Dimensionality of embeddings | Words | 200 | pre-trained |
| | POS tags | 25 | [25,50,75,100] |
| | DTs | 25 | [25,50,75,100] |
| | Relative position to first entity | 10 | Fixed size. See Zhao <i>et al.</i> (35) |
| | Relative position to second entity | 10 | Fixed size. See Zhao <i>et al.</i> (35) |
| | Token type | 10 | [10] |
| Output dimensionality of LSTMs | SDP words | 300 | [100,200,300,400] |
| | SDP POS tags | 200 | [100,200,300,400] |
| | SDP DTs | 200 | [100,200,300,400] |
| | Full sentence tokens | 300 | [200,300] |
| Other architecture parameters | Word-adjacency edge weight | 5 | [3,4,5,6] |
| | Hidden layer, output dimensionality | 1024 | [None, 300,500,1024] |
| | Activation functions | tanh | [tanh, sigmoid] |
| | Dropout rate | 0.2 | [0,0.2,0.3,0.4,0.5] |
| Learning parameters | Mini-batch size | 16 | [16,32,64] |
| | Learning rate | 0.0005 | [0.0005, 0.001, 0.002] |

with conflicting types using the classifier confidence scores. Since all entities are known data in this task, the predictions from the two systems can be aligned using pairs of gold standard entities.

If only one system predicts a relation for a given pair of entities, it is either included in (OR) or discarded from (AND) the combination. If both systems predict a relation, the relation with the higher confidence score is included in the combination. Both SVM and ANN systems produce confidence scores in their own ranges. These ranges are normalized into the 0–1 interval for both systems, after which the normalized scores are compared. We experiment with combining all predictions (all 11 possible classes, including the ‘negative’ class), only positive predictions (all 10 possible classes) or only predictions for the evaluated classes (only the five target classes) and find that system combination in fact leads to better performance scores on the task. Figure 3 illustrates how the predictions of the I-ANN system and the SVM system are combined to produce a final set of predictions for the test set.

Results and discussion

We conduct all of our experiments on the official development set using the official evaluation script provided by the organizers. Even though the data is annotated with 10 different relation types, the task only focuses on 5 of them by defining the official performance metric as the micro-averaged F-score of the five target classes. This is

Table 2. Performance of the systems on the development set

| Evaluation on development set | Performance metrics | | |
|--------------------------------------|---------------------|--------|---------|
| | Precision | Recall | F-score |
| SVM | 64.55 | 54.72 | 59.23 |
| ST-ANN | 61.90 | 55.01 | 58.25 |
| SVM + ST-ANN (OR, positive classes) | 58.45 | 63.99 | 61.09 |
| SVM + ST-ANN (AND, positive classes) | 75.42 | 48.14 | 58.77 |
| SVM + ST-ANN (OR, all classes) | 65.82 | 55.55 | 60.25 |
| SVM + ST-ANN (AND, all classes) | 65.82 | 55.55 | 60.25 |
| SVM + ST-ANN (OR, eval classes) | 56.47 | 65.07 | 60.46 |
| SVM + ST-ANN (AND, eval classes) | 79.28 | 45.78 | 58.04 |

most likely due to the fact that there are much less training examples available in the data for the excluded classes. We first discuss the results of our participation in the shared task (with the SVM and ST-ANN systems and their combination) and then focus on the improved results we obtained using the I-ANN and its combination with the SVM system.

Shared task results

In this section, we discuss all results we have achieved during our participation in the shared task. Table 2 shows the performance comparison of the SVM and ST-ANN systems, evaluated on the development data.

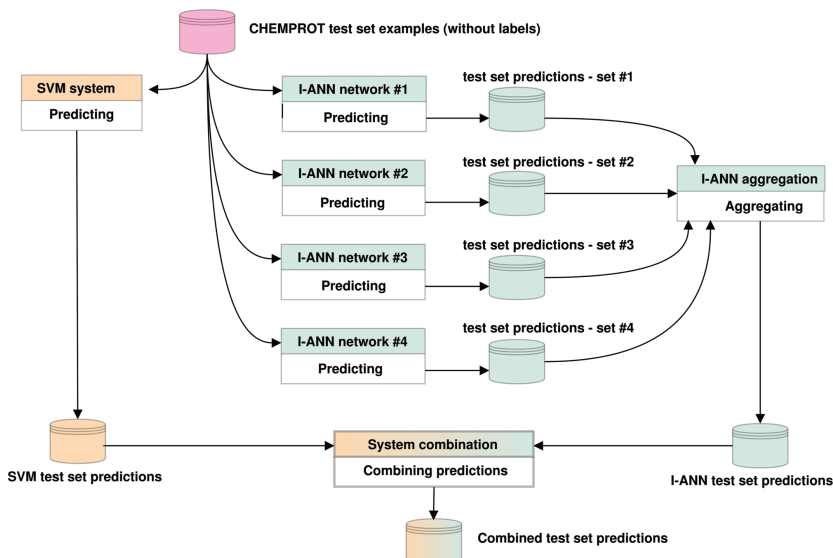


Figure 3. Predicting labels for CHEMPROT test set examples. The figure illustrates how the predictions of the I-ANN system and the SVM system are combined to produce a final set of predictions for the test set. Each neural network in the I-ANN ensemble predicts a set of confidences for each test set example. The confidences for each example are summed together and the label with the highest overall confidence score is selected as the relation type for that example. This aggregation procedure produces the final set of predictions by the I-ANN ensemble. The SVM system also predicts a set of confidences for each example, and the label with the highest confidence score is selected as the predicted relation type for that example. The confidence scores of the I-ANN and the SVM system predictions are further normalized into 0-1 interval. Using one of the aforementioned system combination methods (e.g. intersection or union), the two prediction sets are combined together, producing a combined set of predictions for the test set. The same procedure is applied for predicting labels for the development set/test set examples.

As Table 2 shows, both the SVM and ST-ANN systems have very similar performance on the task, with the SVM having an F-score 1 pp above the ST-ANN. This might be due to the fact that ST-ANN solely relies on the words and edges seen on the shortest path and we suspect that in many cases, the ‘trigger word’ (i.e. a token or sequence of tokens that expresses the actual relation between the two candidate entities) might be absent from this path. Consequently, the ST-ANN might not get the chance to see this information, whereas the SVM system generates features based on all tokens and dependencies near the two entities, as well as those on the shortest path connecting them. The best SVM performance is achieved with the TEES default settings, without using the DrugBank (44) features, using the BLLIP+biomodel+Stanford_CCProcessed parsing approach and including all 10 CHEMPROT relation types in the training data.

We highlight that the Stanford parsing conversion software (38) can produce five variants of the SD representation: ‘basic’, ‘nonCollapsed’, ‘collapsed’, ‘collapsedTree’

and ‘CCprocessed’. As an optimization step, we tried all aforementioned conversions for the CHEMPROT corpus: we first performed constituency parsing using the BLLIP (37) with the biomedical domain model of McClosky (40), and then used the Stanford conversion tool to obtain different variants of dependency parse graphs. This resulted in obtaining five variants of the parsed corpus. For each variant, we trained the SVM system on the training data and evaluated it on the development data. The evaluation resulted in obtaining 57.13 (‘basic’), 57.47 (‘nonCollapsed’), 57.92 (‘collapsed’), 57.82 (‘collapsedTree’) and 59.23 (‘CCprocessed’) F-scores. The ‘CCprocessed’ variant outperformed the other parsing conversion methods by ~ 2 pp. Similarly, parsing the corpus with the SyntaxNet parser (39) resulted in 53.19 F-score on the development set, ~ 6 pp below the best result. Consequently, we used BLLIP+biomodel+Stanford_CCProcessed variant for building the SVM, the ST-ANN and the I-ANN systems.

As Table 2 shows, for both SVM and ST-ANN systems, recall is considerably lower than precision (for instance,

recall is 10 pp below precision for the SVM system). Using the OR operation in system combination considerably improves the recall (~ 9 pp) while causing a comparatively lower drop in precision, leading to an $\sim 1\text{--}1.5$ pp increase in the resulting F-score. We observe that discarding negative predictions and building the combination from all 10 positive classes result in the highest performance on the development set.

For predicting the test set, we combine the training and development data when training the SVM system. This is a common approach when using classifiers such as SVMs. However, training the neural networks on the combined data for the ‘optimal’ number of epochs (found during the optimization) might lead to under/over-fitting, because more/less training epochs might be needed. Finding the optimal number of epochs for training the network on the combined data is challenging. In the shared task, participating teams were allowed to submit up to 5 different test set predictions. Hence, we submitted two sets of ST-ANN predictions: (i) predictions of the ensemble of networks that are trained for 3 epochs (the optimal number found in optimization), (ii) predictions of the ensemble when the networks are trained for 4 epochs. We also combined these two sets of predictions with the SVM system predictions (using our system combination approach), resulting in a total of five sets of test set predictions. Table 3 shows the official results for our submissions on the test set, as calculated by the task organizers.

As Table 3 shows, compared to the development set results, our SVM system has approximately the same level of performance on the test set, achieving an F-score of 60.99, with a similar imbalance between precision (66.88) and recall (56.62). However, for the ST-ANN submissions we notice a significant drop in recall (~ 11 pp) with a small increase in precision (~ 1 pp), leading to an F-score of 52.49 (when the networks are trained for 3 epochs) or 51.85 (when the networks are trained for 4 epochs), which is ~ 6 pp below the F-score seen on the development set. As a direct result, none of the two system combination approaches have been able to produce a result better than the SVM system alone. Hence, our best official shared task

score (60.99 F-score) has been obtained using the SVM system alone.

This massive 6 pp drop in the F-score seen on the test set for the ST-ANN system is clearly abnormal. After the organizers published the test set labels, we performed a comprehensive analysis of the results of this system and discovered a critical mistake in the pipeline: the training data had not been shuffled before each training epoch, an important step preventing mini-batches of highly correlated examples. Since the neural network objective functions are non-convex, using different ordering of training samples may lead to possibly different local minima. The gradient-descent neural network training algorithms are susceptible to becoming stuck in those local minima while a better solution might exist. To summarize, shuffling training data serves the purpose of reducing variance, increasing the chance of obtaining mini-batches that are representative of the overall dataset and thus, making sure the neural network models remain general and overfit less. In the next section, we discuss how shuffling the data changed the results on the development and test sets for the ST-ANN system.

Improved results

In this section we discuss the improved results we obtained after the shared task. Table 4 shows the precision, recall and F-score for all approaches. Row 1 (scores of the SVM system) and Row 2 (scores for the ST-ANN system, when the networks in the ensemble are trained for 3 epochs) are from Tables 2 and 3, for the sake of comparison. The Corrected-ST-ANN system (Row 3) is identical to the ST-ANN system (i.e. trained with the exact hyperparameters, including the learning rate and mini-batch size), except we have shuffled examples before each training epoch. Row 4 shows the scores achieved by our I-ANN. Note that the networks in the I-ANN systems are trained with different learning rate and mini-batch size (see Table 1), comparing to the ST-ANN system. In addition, the networks in this ensemble have been trained for 4 epochs (the optimal value based on optimization on the development set). Finally, Rows 5–10 show the scores achieved by combining the predictions of the SVM system and the I-ANN system, using various aforementioned system combination approaches. For prediction of the test set, we have combined training and development data, when training the SVM, ST-ANN, Corrected-ST-ANN and I-ANN systems.

As Table 4 shows, by comparing the scores of the ST-ANN and the Corrected-ST-ANN systems, we notice that shuffling the examples before each training epoch results in achieving ~ 1 pp increase of the F-score on the

Table 3. Performance of the systems on the test set

| Evaluation on test set | Performance metrics | | |
|---------------------------|---------------------|--------|--------------|
| | Precision | Recall | F-score |
| SVM | 66.08 | 56.62 | 60.99 |
| ST-ANN (trained 3 epochs) | 63.73 | 44.62 | 52.49 |
| ST-ANN (trained 4 epochs) | 63.37 | 43.87 | 51.85 |
| SVM + ST-ANN (3 epochs) | 61.05 | 60.06 | 60.55 |
| SVM + ST-ANN (4 epochs) | 60.88 | 59.89 | 60.38 |

Table 4. Performance of the systems on the development set and the test set

| Row | System | Development set | | | Test set | | |
|-----|-------------------------------------|-----------------|--------------|--------------|--------------|--------------|--------------|
| | | Precision | Recall | F-score | Precision | Recall | F-score |
| 1 | SVM | 64.55 | 54.72 | 59.23 | 66.08 | 56.62 | 60.99 |
| 2 | ST-ANN | 61.90 | 55.01 | 58.25 | 63.73 | 44.62 | 52.49 |
| 3 | Corrected-ST-ANN | 60.51 | 58.01 | 59.23 | 61.55 | 53.93 | 57.49 |
| 4 | I-ANN | 63.18 | 56.25 | 59.51 | 62.39 | 57.81 | 60.01 |
| 5 | SVM + I-ANN (OR, positive classes) | 58.70 | 63.78 | 61.14 | 61.65 | 66.66 | 64.05 |
| 6 | SVM + I-ANN (AND, positive classes) | 74.73 | 48.47 | 58.80 | 74.45 | 50.23 | 59.99 |
| 7 | SVM + I-ANN (OR, all classes) | 65.56 | 56.17 | 60.50 | 65.66 | 58.21 | 61.71 |
| 8 | SVM + I-ANN (AND, all classes) | 65.56 | 56.17 | 60.50 | 65.68 | 58.16 | 61.69 |
| 9 | SVM + I-ANN (OR, eval classes) | 57.65 | 65.81 | 61.46 | 59.05 | 67.76 | 63.10 |
| 10 | SVM + I-ANN (AND, eval classes) | 79.36 | 46.94 | 58.99 | 77.79 | 48.21 | 59.53 |

development set and +5 pp on the test set. We also notice that the difference between the F-score on the development set and on the test set is much smaller for the Corrected-ST-ANN comparing to the ST-ANN system (~2 pp vs ~6 pp), implying that neural networks in the Corrected-ST-ANN system are more robust classification models.

The Corrected-ST-ANN system achieves the same performance level as the SVM system on the development set (59.23 F-score), but interestingly, it performs 3.5 pp below the SVM system on the test set. This suggests the Corrected-ST-ANN overfits more on the training data. Besides, the Corrected-ST-ANN system solely relies on SDP features, while the SVM system generates features based on all tokens and dependencies near the two entities, as well as those on the SDP connecting the two candidate entities. We thus investigate the scores of the I-ANN system since it utilizes whole sentence tokens, besides the features generated from the SDP.

Comparing the I-ANN system with the Corrected-ST-ANN system, we see 0.28 pp F-score increase on the development set and a comparatively larger increase on the test set (2.52 pp). This suggests that incorporating whole sentence features—besides the SDP features—into the networks in the I-ANN system actually helps achieving a better performance for the task. This is also evident as the I-ANN system achieves a similar performance level with the SVM system, both on the development set (59.51 vs 59.23 F-score) and on the test set (60.01 vs 60.99 F-score), but with the additional benefit that the I-ANN system requires much less feature engineering than the SVM system. Still, the recall is comparatively lower than the precision for the SVM, Corrected-ST-ANN and I-ANN systems.

We further investigate the potential of different approaches of combining the predictions of the SVM and I-ANN systems for achieving a higher score for the task. As Table 4 shows (Rows 5, 7 and 9), in all different possible ways of taking the union of the predictions (OR),

the F-score on both development and test set improves over the SVM and I-ANN systems alone. The best F-score on the development set (61.46, Row 9) is achieved by first removing the negative and non-evaluated predictions and then taking the union of the predictions, and resolving overlapping predictions with conflicting types by using the normalized classifier confidence scores. This approach results in an F-score of 63.10 on the test set, and hence, this is our best F-score for the task, 2.11 pp higher than our best test set submission during the shared task. This is also very close to the highest F-score (64.10), achieved by Peng *et al.* (36) in the shared task.

Finally, we notice that our best approach (Row 9) also leads to the highest recall, both on the development set (65.81) and the test set (67.76), whereas removing negative and non-evaluated predictions and taking the intersection of the predictions (Row 10) leads to the highest precision of 79.36 on the developments set, and 77.79 on the test set.

Error analysis

In this section we perform an error analysis and compare the performance of the SVM system with the I-ANN system on the CHEMPROT test set that contains 800 article abstracts. Although CHEMICAL–GENE pairs in the test set are annotated with 10 possible positive relation types, only 5 of these classes are taken into account in the task evaluation, with the micro-averaged F-score of the five target classes as the official metric for evaluation. However, since the SVM and the I-ANN systems are trained to predict and assign one of the 11 possible labels to each pair (10 positive classes and a negative class), we find it more informative to consider all predicted labels when examining and comparing the performance of the two systems.

The test set includes 5744 positive annotations for 5665 unique CHEMICAL–GENE pairs. Even though the majority of the pairs are annotated with a single positive label,

Table 5. Test set annotations

| Class name | Evaluated in the task | Relation types | Number of annotations in the test set |
|------------|-----------------------|---|---------------------------------------|
| CPR:1 | No | PART_OF | 215 |
| CPR:2 | No | REGULATOR DIRECT_REGULATOR INDIRECT_REGULATOR | 1743 |
| CPR:3 | Yes | UPREGULATOR ACTIVATOR INDIRECT_UPREGULATOR | 667 |
| CPR:4 | Yes | DOWNREGULATOR INHIBITOR INDIRECT_DOWNREGULATOR | 1667 |
| CPR:5 | Yes | AGONIST AGONIST-ACTIVATOR AGONIST-INHIBITOR | 198 |
| CPR:6 | Yes | ANTAGONIST | 293 |
| CPR:7 | No | MODULATOR MODULATOR-ACTIVATOR MODULATOR-INHIBITOR | 25 |
| CPR:8 | No | COFACTOR | 25 |
| CPR:9 | Yes | SUBSTRATE PRODUCT_OF SUBSTRATE_PRODUCT_OF | 644 |
| CPR:10 | No | NOT (explicit mention of having no effects/interactions) | 267 |
| neg | No | Generated negatives for the pairs with no gold-standard annotations | 10025 |

there are 79 pairs in the test set with more than one assigned label. Since both SVM and I-ANN systems predict a single label for each pair, we repeat the same predicted label for these multi-label pairs for evaluation. Besides, there are five cross-sentence annotations in the test set, but since the SVM and the I-ANN systems only extract relations from single sentences, we count these five pairs as false negatives. In addition, we generate negatives (as gold-standard relations) between any CHEMICAL and GENE entity mentions in the ‘same sentence’ if they do not have any corresponding gold-standard annotations. Table 5 summarizes the information about the annotations in the test set that we have used for our internal evaluation and Table 6 shows the confusion matrix, precision, recall and F-score for the SVM and the I-ANN systems. We highlight that the evaluation numbers in this section are based on the aforementioned evaluation procedure (i.e. our internal evaluation) and could not have been obtained using the official evaluation script provided by task organizers. Hence, there is a slight difference in the micro-averaged F-score of the target classes (the task metric) between the numbers reported in Tables 4 and 6, most likely due to possible differences in the evaluation of cross-sentence and multi-label pairs (duplications).

As Table 6 shows, both systems have failed to predict any CPR:7 (modulator) and CPR:8 (cofactor) labels, the two rarest classes in the dataset by an order of magnitude. Consequently, the F-score for these classes is zero for both systems. All pairs with CPR:7 or CPR:8 true label are misclassified as having CPR:2, CPR:3, CPR:4, CPR:6, CPR:9 relations or not having any relation (neg), by both systems.

Even though the F-score for the negative class is relatively high for both systems, all other classes are highly confused with this class. For example, about 55% (958/1743) and 43% (742/1743) of the relations having CPR:2 relation are misclassified as being negative by the SVM and I-ANN systems, dramatically dropping the recall

for the CPR:2 class. This indicates that the SVM and I-ANN classifiers are not highly efficient in distinguishing positive relations from the negative ones. Building a two-step relation extraction system might be one idea to deal with this problem. These systems are generally composed of two classifiers, with the first classifier labeling each relation as being positive or negative and the second classifier detecting the type of relation for the pairs that are identified as positive. As there is high imbalance between the number of positives and negatives in the corpus, negative subsampling or class weighting might be other promising techniques to tackle this problem.

Considering the micro-averaged F-score of the target classes as the overall evaluation metric, the two systems have very similar performance for the task, with 60.10 F-score for the SVM system and 60.15 F-score for the I-ANN system. However, the precision and recall are much more balanced in the I-ANN system. For example, there is ~40 pp difference between the precision and recall for the CPR:one class in the SVM predictions, whereas the difference is ~12 pp for the I-ANN system. Similarly, precision and recall for the CPR:two classes have ~27 pp difference with the SVM system, significantly higher than the ~3 pp difference with the I-ANN system.

As Table 6 shows, and not surprisingly, the examples with types CPR:2 (regulation), CPR:3 (upregulation) and CPR:4 (downregulation) are more misclassified as each other, but less as other relation types (CPR:5, CPR:6, CPR:9 and CPR:10) that are semantically very different. For instance, ~17% (119/667) of the examples with CPR:3 (upregulation) true label are misclassified as having CPR:4 (downregulation) label by the SVM system whereas only five such examples are misclassified as having CPR:5 (agonist) or CPR:6 (antagonist) relation. A manual inspection of the CHEMPROT corpus sentences revealed that the CPR:2 (regulation) is usually associated

Table 6. Confusion matrix and evaluation metrics for the SVM and the I-ANN systems

| SVM system | Predicted labels | | | | | | | | | | | Total annotations | Precision | Recall | F-score | |
|-------------|------------------|------|------|------|------|------|------|------|------|------|-----|-------------------|-----------|--------|---------|-------|
| | CPR1 | CPR2 | CPR3 | CPR4 | CPR5 | CPR6 | CPR7 | CPR8 | CPR9 | CP10 | neg | | | | | |
| True labels | CPR1 | 64 | 11 | 4 | 3 | 0 | 0 | 0 | 2 | 0 | 131 | 215 | 70.33 | 29.77 | 41.83 | |
| | CPR2 | 5 | 466 | 81 | 178 | 2 | 8 | 0 | 1 | 36 | 8 | 958 | 1743 | 53.69 | 26.74 | 35.70 |
| | CPR3 | 0 | 23 | 295 | 119 | 3 | 2 | 0 | 0 | 3 | 2 | 220 | 667 | 55.87 | 44.23 | 49.37 |
| | CPR4 | 0 | 58 | 44 | 1175 | 0 | 3 | 0 | 0 | 7 | 7 | 373 | 1667 | 65.35 | 70.49 | 67.82 |
| | CPR5 | 0 | 14 | 1 | 19 | 74 | 3 | 0 | 0 | 1 | 3 | 83 | 198 | 73.27 | 37.37 | 49.50 |
| | CPR6 | 0 | 16 | 0 | 2 | 4 | 154 | 0 | 0 | 1 | 1 | 115 | 293 | 84.15 | 52.56 | 64.71 |
| | CPR7 | 0 | 7 | 1 | 5 | 0 | 2 | 0 | 0 | 2 | 0 | 8 | 25 | 0.00 | 0.00 | 0.00 |
| | CPR8 | 0 | 0 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 25 | 0.00 | 0.00 | 0.00 |
| | CPR9 | 0 | 10 | 4 | 28 | 0 | 0 | 0 | 0 | 233 | 1 | 368 | 644 | 67.15 | 36.18 | 47.02 |
| | CP10 | 0 | 30 | 17 | 54 | 0 | 0 | 0 | 0 | 2 | 53 | 111 | 267 | 58.24 | 19.85 | 29.61 |
| | neg | 22 | 233 | 80 | 210 | 18 | 11 | 0 | 0 | 60 | 16 | 9375 | 10025 | 79.71 | 93.52 | 86.06 |

Micro-averaged F-score (all classes): 75.39

Micro-averaged F-score (target classes): 60.10

| I-ANN system | Predicted labels | | | | | | | | | | | Total annotations | Precision | Recall | F-score | |
|--------------|------------------|------|------|------|------|------|------|------|------|------|-----|-------------------|-----------|--------|---------|-------|
| | CPR1 | CPR2 | CPR3 | CPR4 | CPR5 | CPR6 | CPR7 | CPR8 | CPR9 | CP10 | neg | | | | | |
| True labels | CPR1 | 108 | 9 | 1 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 91 | 215 | 62.43 | 50.23 | 55.67 |
| | CPR2 | 4 | 731 | 71 | 139 | 13 | 11 | 0 | 0 | 18 | 14 | 742 | 1743 | 39.43 | 41.94 | 40.64 |
| | CPR3 | 0 | 52 | 326 | 84 | 1 | 1 | 0 | 0 | 5 | 1 | 197 | 667 | 52.84 | 48.88 | 50.78 |
| | CPR4 | 0 | 79 | 65 | 1107 | 2 | 9 | 0 | 0 | 35 | 7 | 363 | 1667 | 66.65 | 66.41 | 66.53 |
| | CPR5 | 0 | 18 | 4 | 9 | 102 | 11 | 0 | 0 | 0 | 0 | 54 | 198 | 60.00 | 51.52 | 55.43 |
| | CPR6 | 0 | 21 | 0 | 8 | 12 | 198 | 0 | 0 | 1 | 0 | 53 | 293 | 73.33 | 67.58 | 70.34 |
| | CPR7 | 0 | 9 | 4 | 6 | 0 | 2 | 0 | 0 | 0 | 0 | 4 | 25 | 0.00 | 0.00 | 0.00 |
| | CPR8 | 0 | 5 | 1 | 2 | 0 | 0 | 0 | 0 | 7 | 0 | 10 | 25 | 0.00 | 0.00 | 0.00 |
| | CPR9 | 0 | 52 | 3 | 8 | 0 | 0 | 0 | 0 | 274 | 5 | 302 | 644 | 56.38 | 42.55 | 48.50 |
| | CP10 | 0 | 23 | 16 | 30 | 0 | 4 | 0 | 0 | 9 | 94 | 91 | 267 | 48.45 | 35.21 | 40.78 |
| | neg | 61 | 855 | 126 | 267 | 40 | 34 | 0 | 0 | 132 | 73 | 8437 | 10025 | 81.56 | 84.16 | 82.84 |

Micro-averaged F-score (all classes): 72.15

Micro-averaged F-score (target classes): 60.15

with words such as 'regulation', 'interaction', 'binding', 'expression', 'relationship', 'involvement', 'change' and 'initiates'. The CPR:3 (upregulation) is associated with words/phrases such as 'induced', 'promotes', 'activates' and 'increases the activity of' and the CPR:4 (downregulation) is usually expressed with words/phrases such as 'inhibits', 'blocks' and 'decreases the activity of'. However, the CPR:5, CPR:6 and CPR:9 relation types are usually expressed with semantically very different words such as 'agonist', 'antagonist', 'substrate', 'catalyzes', 'mediates' and 'metabolism'.

We performed an error analysis on incorrect test set predictions made by the two systems. We noticed that in many cases that the CPR:2, CPR:3 and CPR:4 labels are misclassified as each other, the sentences are

either complex, or if the sentence is simple, the SDP (or the full sentence) contains words that are usually associated with different classes. For example, in the sentence '<CHEMICAL>Xanthohumol</CHEMICAL> and 2-hydroxychalcone induced apoptosis by <GENE>Bcl-2</GENE> downregulation.' having a CPR:4 (downregulation) interaction between the 'Xanthohumol' and 'Bcl-2' entities, the relation is misclassified as CPR:3 (upregulation) by both SVM and I-ANN systems. The SDP in this sentence (Xanthohumol, induced, downregulation, Bcl-2) contains the word 'induced' (usually associated with CPR:3) and 'downregulation' (associated with CPR:4). Similarly, there are three CPR:4 (downregulation) relations between the chemical 'Cholesterol' and the three genes in the

sentence ‘<CHEMICAL>Cholesterol</CHEMICAL> also increases <GENE>Amyloid β </GENE> (<GENE>A β </GENE>) deposition and <GENE>tau</GENE> pathology.’. All the three relations are misclassified as upregulation by both systems, and the sentence contains the word ‘increases’, mostly associated with upregulation. In some cases, the sentences contain a mixture of positive and negative regulations, confusing the classifiers. For example, in the sentence ‘Although, <CHEMICAL>imatinib</CHEMICAL> primarily inhibits <GENE>tyrosine kinases</GENE>, it also stimulates the activity of <GENE>EGFR</GENE> <GENE>tyrosine kinase</GENE> in head and neck squamous tumors.’, the chemical entity ‘imatinib’ downregulates the (first mention of) ‘tyrosine kinases’, but upregulates the ‘EGFR’ and the second ‘tyrosine kinases’ entity mention. However, all three relations are assigned CPR:3 (upregulation) label by the I-ANN system.

It is also interesting to note that some examples with CPR:9 type (substrate/product of) are misclassified as having regulation, upregulation or downregulation relations. These usually belong to long and very complicated sentences that might be hard even for a non-expert human to distinguish. For example, the sentence ‘As discussed in this review, various progestogens including dydrogesterone and its 20alpha-dihydro-derivative, medrogestone, promegestone, nomegestrol acetate and norelgestromin can reduce intratissular levels of estradiol in breast cancer by blocking sulfatase and 17beta-hydroxy steroid-dehydrogenase type 1 activities.’ contains eight chemical entities (‘progestogens’, ‘dydrogesterone’, ‘medrogestone’, ‘Promegestone’, ‘nomegestrol acetate’, ‘norelgestromin’, ‘estradiol’, ‘17beta-hydroxysteroid’) and two gene entities (‘sulfatase’, ‘17beta-hydroxysteroid-dehydrogenase type 1’). All CHEMICAL–GENE pairs have CPR:4 relation (inhibitor), except (‘estradiol’, ‘sulfatase’), (‘estradiol’, ‘17beta-hydroxysteroid-dehydrogenase type 1’) pairs, which have CPR:9 (substrate) relation. Both classifiers have assigned CPR:4 label to all pairs and failed to detect that ‘estradiol’ is the substrate of the genes.

The class label CPR:10 is also very challenging since it is used for pairs with ‘explicit mention of not having any effects on’ relation, which is semantically different from the negative class (‘no information about having relations/interactions’). For example, in the sentence ‘The induction of <GENE>HO-1</GENE> by EIH was inhibited by <CHEMICAL>SB203580

</CHEMICAL> but not by <CHEMICAL>SP600125</CHEMICAL>, <CHEMICAL>PD98059</CHEMICAL>, nor <CHEMICAL>LY294002</CHEMICAL>.’, the (HO-1, SB203580) pair has CPR:4 (inhibitor) relation, whereas the other three chemicals have CPR:10 relation with ‘HO-1’ gene. As Table 6 shows, the F-score for this class is low (29.61 with the SVM and 40.78 with the I-ANN classifier) and this class is highly confused with all other classes. For example, in the simple sentence ‘Neither <CHEMICAL>oxycodone</CHEMICAL> nor its metabolites activated <GENE>PXR</GENE>, <GENE>CAR</GENE>, or <GENE>AhR</GENE>.’, all three CHEMICAL–GENE pairs with CPR:10 true label are mistakenly assigned CPR:3 (upregulation) labels by our classifiers, most likely because the word ‘activated’ (strong indicator of upregulation) is inside the sentence. Generally, detecting such relations is a major challenge in relation extraction and while the I-ANN system performs better than the SVM system on this class, there is clearly room for improvement.

We further manually analyzed and compared the misclassifications made by the SVM system with the I-ANN system in order to check if certain syntactic/semantic patterns can be linked to only one of the classifiers. We did not find any particular patterns that can be exclusively attributed to only one of the systems. In addition, we systematically compared the two systems based on the average length of misclassified sentences (in terms of the number of tokens) to check if one system better deals with longer sentences and found out both systems have similar performance levels on long sentences.

Comparison with other methods

In this section, we concisely compare our methods with the top performing relation extraction methods that are evaluated on the CHEMPROT corpus. Even though 13 teams participated in the shared task, only 6 teams (including us) achieved an F-score higher than 50. Table 7 lists the performance measures of the top performing methods on the CHEMPROT test set.

As Table 7 shows, the highest F-score (64.10) in the shared task has been achieved by Peng *et al.* (36), with a system combination approach. Their method is composed of three separate systems: (i) a CNN-based relation extraction system that utilizes separate convolutional layers to simultaneously learn SDP representation and full sentence representation and uses a six-dimensional decision layer (for the five positive target classes and the negative class) with softmax activation; (ii) an RNN-based relation extraction system that utilizes a bidirectional LSTM network and max-pooling and learns full sentence representation and uses a

Table 7. Top performing methods on the chemprot test set

| Row | Method summary | Authors | Task metrics | | |
|-----|--|-------------------------|--------------|--------|---------|
| | | | Precision | Recall | F-score |
| 1 | An ensemble of CNN, RNN and SVM -based systems | Peng <i>et al.</i> (36) | 72.66 | 57.35 | 64.10 |
| 2 | SVM + I-ANN (our best approach) | this paper | 59.05 | 67.76 | 63.10 |
| 3 | A deep learning-based method composed of a ‘pretraining’ network and a ‘recognition’ network, utilizing bidirectional LSTMs and CNNs | Corbett and Boyle (55) | 56.10 | 67.84 | 61.41 |
| 4 | An ensemble of Tree-LSTM networks | Lim and Kang (56) | 67.04 | 51.94 | 58.53 |
| 5 | A feature-based method with gradient-boosted trees classifier and a feature-selection component for optimization | Lung <i>et al.</i> (57) | 63.52 | 51.21 | 56.71 |
| 6 | Bidirectional LSTM networks | Matos (58) | 57.38 | 47.22 | 51.81 |

five-dimensional decision layer (only for the five positive target classes) with a linear activation; (iii) an SVM-based system that generates features based on the SDP and tokens in the full sentence. To combine the predictions and build an ensemble of the three systems, they use either majority voting or stacking. For stacking they train a random forest classifier that uses the confidence scores (obtained from the CNN, RNN and SVM systems) as features to assign a label to each example. They build SVM, CNN and RNN models using 80% of total data (training + development) and build the ensemble using the remaining 20% of the total data. In addition, they use 5-fold cross-validation using different partitions of the data to reduce variability. Hence, they obtained five SVMs, five CNNs and five RNNs in total. For participation in the shared task, they submitted five runs (i.e. five sets of predictions for the test set), two based on majority voting and three based on the aforementioned stacking approach, each run using one SVM, CNN and RNN from one cross-validation iteration. Their best F-score (Row 1 in Table 7) has been achieved with one of the submissions based on the stacking approach.

The I-ANN system and the CNN-based system developed by Peng *et al.* (36) are similar in terms of simultaneously learning full sentence representation and SDP representation. However, the I-ANN system utilizes LSTM networks whereas their CNN model uses separate convolutional layers for this purpose. For learning full sentence vector representation, both I-ANN and the RNN-based system of Peng *et al.* (36) use a bidirectional LSTM network. The I-ANN system is trained to assign 1 of the 11 possible labels to each example, whereas their CNN/RNN models assign either one of the five target class labels or the negative label to each example. The other difference is that the I-ANN system utilizes an additional dense layer after the LSTM layers. As discussed in the previous section, these choices are made based on the optimization process we performed on the development set. We highlight that unfortunately Peng *et al.* (36) have not published the performance measures of their individual systems, thus we cannot directly

compare the performance of the I-ANN or the SVM system with their individual systems. As Table 7 shows, our system combination approach and their best approach perform closely on the task, achieving 63.10 and 64.10 F-score, respectively. Their system has higher precision (72.66 vs 59.05), but lower recall (57.35 vs 67.76), compared to our system.

Corbett and Boyle (55) achieved 61.41 F-score on the task, 1.69 pp below our best approach, and 2.69 pp below the best score, with a deep learning-based approach. Their system is composed of two neural networks: a ‘pretraining’ network that utilizes a bidirectional LSTM network for transfer learning, and a ‘recognition’ network that utilizes bidirectional LSTMs and CNNs. The pretraining network—which is trained on all the titles and abstracts from PubMed records from 1809 to the end of 2015—has three inputs and two outputs. The inputs are the original sentence sequence, the ‘substituted’ sequence shifted one token to the right, and the ‘substituted’ sequence shifted one token to the left. In the substituted sequences, each token has a 50% chance of being replaced by a token randomly sampled from the lines read in that sub-epoch. The pretraining network has two outputs, one for each of the substituted shifted sequences, consisting of a sequence of numbers -1 if the token in the substituted sequence is from the original sequence, or 0 if it was randomly selected. The recognition network is trained only on the CHEMPROT data and uses the same LSTM layers that are pre-trained in the pretraining network and two additional convolutional layers and a bidirectional LSTM network. They train the pretraining and the recognition networks with series of epochs, with the first 5 epochs composed of training both networks, and the rest only training the recognition network. Although their system has a lower F-score and recall compared with our system and the system developed by Peng *et al.* (36), its main advantage is the ability to work on raw texts for extracting CHEMICAL–GENE interactions, e.g. no sentence parsing is required. Parsing is usually one of the most time-consuming steps in relation extraction system

pipelines, thus eliminating this step may considerably improve the run-time performance of large-scale real-world applications.

Lim and Kang (56) participated in the shared task with an ensemble of Tree-LSTM networks (59) that process the sentence parse graph. Each node in the Tree-LSTM architecture is a word in the graph, represented by concatenating embeddings of its words, relative positions to the two entities and a subtree containment feature. The subtree containment feature for a node is simply 'True' when one of the target entities exists in the leaves of the current node, and is 'False' otherwise. These two values are further mapped into two 10-dimensional embeddings: if the value is 'True', every element of a vector is +1; otherwise, every element in a vector is 0. In contrast with normal LSTM networks in which each unit receives the input only from the previous unit (i.e. the previous word in the sentence or SDP), in the Tree-LSTM network, each node in the tree receives the input from multiple child nodes (leaves) and updates the hidden state of current node using those inputs. Similar to the I-ANN system, they also train an ensemble of the neural networks and aggregate their predictions by simply taking the sum of the confidence scores to deal with the variance caused by random initialization of network weights. Their method achieved an F-score of 58.53 on the task, 4.57 pp below our F-score and 5.57 pp below the highest F-score achieved by Peng *et al.* (36) during the shared task.

Lung *et al.* (57) achieved an F-score of 56.71 using a feature-based method that relies on manually engineered features, extracted from both semantic pattern and dependency parse graph of the sentence. The semantic pattern reveals whether/how chemical-protein interactions are stated in the sentence, whereas the dependency graph provides the information on how words are interconnected in the sentence. To analyze the semantic pattern, they have manually built an extensive list of words (including the 'trigger' words) and check whether these words are found in the sentence or not. In addition, they employ a set of features previously found to be beneficial for protein-protein relation extraction from biomedical texts. For example, a binary feature captures whether negative words (e.g. 'not', 'incapable' and 'unable') are in the region covered by the candidate pair and a binary feature shows if sentence breaking words (e.g. 'although', 'therefore', 'whereas') exist in the region. They have reported these features are helpful in chemical-protein relation extraction as well. To analyze the sentence structure, they only target and extract a set of features from the SDP. These features include the number of tokens in the SDP, as well as binary features for checking the presence of different DT edges in the SDP. They use gradient boosted trees for classification and feature selection to optimize their system.

Finally, Matos (58) achieved 51.81 F-score on the task (11.29 pp below our F-score), using relation extraction systems composed of three to six bidirectional LSTM networks. All of their networks utilize three bidirectional LSTMs for processing the words, POS tags and DTs along the SDP. However, they also experiment with using up to three additional bidirectional LSTMs, for processing the words before/between/after the two entities. Unfortunately, a few important details are missing from their paper that are necessary for a correct comparison of their system with the ST-ANN/I-ANN systems. For example, it is not clear how the outputs of the bidirectional LSTMs are combined together in their networks (e.g. in the I-ANN system, max pooling is first applied and SDP and full sentence vector representations are further concatenated). Similarly, no details about the output dimensionality and the activation function of the decision layer in their networks is mentioned, hence it is not clear whether they assign one of the five target class/the negative label to each example, or similarly to the ST-ANN/I-ANN systems, they assign one of 11 possible labels. On the development set, they achieved 54.70 F-score (with the system that uses all six bidirectional LSTMs), 56.64 F-score (with the system that uses three bidirectional LSTMs for the SDP) and their highest F-score of 59.19 has been achieved with a system utilizing three bidirectional LSTMs for the SDP and a bidirectional LSTM for the words between the two entities. However, on the test set, the network with all six LSTMs has achieved a lower F-score of 51.81 (with ~ 7 pp increase in the precision, but 14 pp drop in the recall, compared to the results on the development set). The F-score for the system with the three bidirectional LSTMs (for the SDP) has dramatically dropped to 34.18, and for the system with the four bidirectional LSTMs to 36.77. As mentioned in their paper, further error analysis is needed to give some indication on how generalization could be improved.

Runtime performance and technical details

We implement the systems using the Python programming language (v2.7) with the Keras (60) deep learning library and the Theano tensor manipulation library (61) (as the backend engine for Keras) for implementing the neural network models. All neural network parameters not explicitly discussed in this paper were left to their defaults in Keras. All computations were run on a single server computer equipped with 64 gigabytes of memory, one 8-core CPU and one NVIDIA® TESLA® K80 GPU (with 4992 CUDA cores). Parsing and all python processing, including e.g. file manipulation, the TEES pipeline and system combination were run on the CPU, whereas all neural network related

calculations (training, optimization and prediction) were run on the GPU, using the CUDA toolkit version 7.5.

The CHEMPROT corpus contains 2432 PubMed abstracts (1020 abstracts in the training set, 612 abstracts in the development set and 800 abstracts in the test set) (1). Parsing the corpus and conversion into the TEES XML format takes about 1 hour and 35 minutes. Training the SVM system takes about 40 minutes, whereas training each of the four neural models in the I-ANN ensemble takes ~6 hours, on the combined training and development sets. These times include feature generation, but the training and development sets are already converted into the TEES XML format and parsed. Prediction of the test set using the SVM system takes ~4 minutes, and with each of the four neural networks in the I-ANN ensemble, ~6 minutes. Aggregating the predictions of the networks and running the system combination code takes less than a minute. Consequently, the prediction time for each abstract is in average ~2.2 seconds, assuming that the abstract has already been parsed.

We highlight that the number of neural networks in the I-ANN system is implemented as an input parameter in our software, with 4 being the default value and 1 as the minimum possible value. Hence, training an ensemble is optional. However, because of the reasons discussed earlier, we recommend to train an ensemble of networks when training/optimizing our system on a new corpus, if sufficient computational resources are available.

Conclusions and future work

In this study, we presented three different systems capable of extracting relations between CHEMICAL and GENE entities, as a part of our participation in the BioCreative VI Task 5 (CHEMPROT) challenge. Our SVM system relies on a rich set of features generated from all tokens and dependencies in the SDP and near the two candidate entities. Unlike the SVM system, our ST-ANN and I-ANN systems are based on deep learning and require less feature engineering. The ST-ANN system solely relies on the SDP features, whereas the I-ANN system utilizes features generated from the whole sentence, besides the features generated from the SDP. The ST-ANN system has lower performance compared to the SVM and I-ANN systems, while the I-ANN and SVM perform equally well on the development and test sets, suggesting that incorporating features gathered from the full sentence actually helps achieving better scores for the task, regardless of the classification method.

We also experimented with basic methods of combining the predictions of the SVM and either the ST-ANN or I-ANN systems (e.g. taking the union/intersection of the predictions of two systems) and noticed system combina-

tion achieves the highest F-score of 63.10 on the test set, 2.11 pp higher than our best test set submission during the shared task. Our best F-score is 1 pp below the highest score achieved by Peng *et al.* (36) in the shared task.

There are many interesting future directions that we would like to explore. As we discussed in the previous section, the SVM and I-ANN systems are not highly efficient in distinguishing positive examples from the negative examples, i.e. the examples of other classes are highly misclassified as being negative, lowering the recall of non-negative classes. We would like to investigate whether two-stage classification and/or negative sub-sampling or class weighting can diminish this problem.

Although the ensemble method we used in the I-ANN system addressed the problem of variance in the performance (caused by random initialization of network weights), it does not improve the overall F-score, because the ensemble acts like an average neural network, but robust and indifferent to the initial random weights used to train the individual networks. We would like to try better ensemble methods. One idea is to train the ensemble but instead of taking the sum of the 'all' networks confidences, we take the sum of N top-performing networks. Even though this approach might seem to be very promising, it can lead to heavy overfitting on the development set and consequently, poor generalization for unseen data. We think that heavy regularization and/or cross-validation [as used by Peng *et al.* (36)] will be necessary in that case. In addition, in this work we used basic methods (e.g. taking union/intersection) for combining the predictions of the SVM and I-ANN systems. As a future work, we would like to investigate better system combination approaches [such as the stacking approach used by Peng *et al.* (36)], and see to what extent the overall performance of our method can be improved.

Even though we used pre-trained word embeddings for the I-ANN system, other embeddings (e.g. the POS tag and DT embeddings) were initialized randomly and learnt from scratch. Similar to using pre-trained word embeddings, we would like to investigate whether pretraining the other embeddings can improve the performance of the I-ANN system. One idea is to train the I-ANN system on other biomedical relation extraction corpora (such as DDI-2013) and use the learnt embeddings to train/optimize the I-ANN system on the CHEMPROT training data.

Additionally, we would like to investigate different methods of incorporating the information in the whole parse graph into the neural networks. Although the I-ANN system utilizes the words/POS tags/DTs in the SDP, other word-dependencies far outside the SDP can play a critical role, considerably affecting the meaning of the relations expressed in the sentence. As discussed in the

error analysis section, in the simple sentence ‘Neither <CHEMICAL>oxycodone</CHEMICAL> nor its metabolites activated <GENE>PXR</GENE>, <GENE>CAR</GENE>, or <GENE>AhR</GENE>.’, all CHEMICAL–GENE pairs with CPR:10 true label were mistakenly assigned CPR:3 (upregulation) labels by our classifiers. In this sentence, the key negation words (‘Neither’ and ‘nor’) are not part of the SDP connecting the CHEMICAL to the GENE entities, and most likely because the word ‘activated’ (strong indicator of upregulation) is in the sentence, the relations are misclassified. One approach for incorporating the whole parse graph into neural networks is the Tree-LSTM neural architecture (59), used by Lim and Kang (56) for relation extraction. Unfortunately, they achieved 58.53 F-score on the task, 4.57 pp below our F-score. We would like to investigate whether/how the Tree-LSTM network architecture can be modified to obtain higher scores for the task. Finally, we want to explore different possible ways of incorporating DT *n*-grams (i.e. ‘paths’ in the sentence parse graph) into neural networks as embeddings for this aim.

We highlight that the systems we presented in this study are not applicable only to the BioCreative VI Task 5 and can be effortlessly re-trained to extract any types of relations of interest, with no modifications of the source code required, if a manually annotated corpus is provided as training data in the Interaction XML format (15).

Acknowledgements

We would like to thank the anonymous reviewers who helped us to improve this paper with their valuable recommendations and feedback. Computational resources are provided by CSC-IT Center For Science Ltd, Espoo, Finland.

Funding

ATT Tieto käyttöön grant of the Finnish Ministry of Education.

Conflict of interest. None declared.

References

- Krallinger,M., Rabal,O., Akhondi,S.A. *et al.* (2017) Overview of the BioCreative VI chemical-protein interaction Track. In: Proceedings of BioCreative VI Workshop, Bethesda, MD, USA, 141–146.
- Krallinger,M., Leitner,F., Rodriguez-Penagos,C. *et al.* (2008) Overview of the protein–protein interaction annotation extraction task of BioCreative II. *Genome Biol.*, 9, S4.
- Krallinger,M., Vazquez,M., Leitner,F. *et al.* (2011) The protein–protein interaction tasks of BioCreative III: classification/ranking of articles and linking bio-ontology concepts to full text. *BMC Bioinformatics*, 12, S3.
- Wei,C.H., Peng,Y., Leaman,R. *et al.* (2016) Assessing the state of the art in biomedical relation extraction: overview of the BioCreative V chemical-disease relation (CDR) task. *Database*, 2016, baw032, <https://doi.org/10.1093/database/baw032>.
- Segura-Bedmar,J., Martínez,P. and Sánchez-Cisneros,D. (2011) The 1st DDIExtraction-2011 challenge task: extraction of drug–drug interactions from biomedical texts. In: Proceedings of the 1st Challenge Task on Drug-Drug Interaction Extraction 2011, Huelva, Spain, 1–9.
- Segura-Bedmar,J., Martínez,P. and Herrero Zazo,M. (2013) SemEval-2013 Task 9: extraction of drug–drug interactions from biomedical texts (DDIExtraction 2013). In: Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013). Association for Computational Linguistics, Atlanta, Georgia, USA. 341–350.
- Bossy,R., Jourde,J., Bessieres,P. *et al.* (2011) Bionlp shared task 2011: bacteria biotope. In: Proceedings of the BioNLP Shared Task 2011 Workshop, Portland, Oregon, USA, 24 June, 2011. Association of Computational Linguistics, 56–64.
- Bossy,R., Golik,W., Ratkovic,Z. *et al.* (2013) Bionlp shared task 2013—an overview of the bacteria biotope task. In: Proceedings of the BioNLP Shared Task 2013 Workshop, Sofia, Bulgaria, 9 August, 2013. Association of Computational Linguistics, 161–169.
- Deleger,L., Bossy,R., Chaix,E. *et al.* (2016) Overview of the bacteria biotope task at bionlp shared task 2016. In: Proceedings of the 4th BioNLP Shared Task Workshop, Berlin, Germany, 13 August 2016. Association of Computational Linguistics, 12–22.
- Pyysalo,S., Sætre,R., Tsujii,J.I. *et al.* (2008) Why biomedical relation extraction results are incomparable and what to do about it. In: Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), Turku, Finland, 1st September 2008. TUCS General Publication, 149–152.
- Cortes,C. and Vapnik,V. (1995) Support-vector networks. *Mach. Learn.*, 20, 273–297.
- Hochreiter,S. and Schmidhuber,J. (1997) Long short-term memory. *Neural Comput.*, 9, 1735–1780.
- Mehryary,F., Björne,J., Salakoski,T. *et al.* (2017) Combining support vector machines and LSTM networks for chemical–protein relation extraction. In: Proceedings of the BioCreative VI Workshop, Bethesda, MD, USA, 175–179.
- Zhang,Q., Chen,M. and Liu,L. (2017) A review on entity relation extraction. In: 2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE), Harbin, IEEE. 178–183.
- Björne,J. (2014) Biomedical event extraction with machine learning. Ph.D. Thesis. University of Turku.
- Lever,J. and Jones,S.J. (2016) VERSE: event and relation extraction in the BioNLP 2016 Shared Task. In: Proceedings of the 4th BioNLP Shared Task Workshop, Berlin, Germany, 13 August 2016. Association of Computational Linguistics, 42–49.
- Raihani,A. and Laachfoubi,N. (2016) Extracting drug–drug interactions from biomedical text using a feature-based kernel approach. *J. Theor. Appl. Inf. Technol.*, 92, 109.

18. Zelenko,D., Aone,C. and Richardella,A. (2003) Kernel methods for relation extraction. *J.machine Learn. Res.*, 3, 1083–1106.
19. Culotta,A. and Sorensen,J. (2004) Dependency tree kernels for relation extraction. In: Proceedings of the 42nd annual meeting on association for computational linguistics, Barcelona, Spain, 21 July, 2004. Association of Computational Linguistics, 423–429.
20. Freund,Y. and Schapire,R.E. (1999) Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37, 277–296.
21. Bunescu,R.C. and Mooney,R.J. (2005) Subsequence kernels for relation extraction. In: Proceedings of the Advances in Neural Information Processing Systems 18 (NIPS 2005), Cambridge, MA, USA. MIT Press, 171–178.
22. Reichartz,F., Korte,H. and Paass,G. (2009) Dependency tree kernels for relation extraction from natural language text. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Berlin, Heidelberg, Springer, 270–285.
23. Airola,A., Pyysalo,S., Björne,J. *et al.* (2008) *All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning*, BMC Bioinformatics, 9(Suppl 11), S2.
24. Zhang,Y., Zheng,W., Lin,H. *et al.* (2017) Drug–drug interaction extraction via hierarchical RNNs on sequence and shortest dependency paths. *Bioinformatics.*, 34(5), 828–835.
25. Goodfellow,I., Bengio,Y. and Courville,A. (2016) *Deep Learning*. Cambridge, MA, USA. The MIT Press.
26. Mikolov,T., Sutskever,J., Chen,K. *et al.* (2013) Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13), Lake Tahoe, Nevada. Curran Associates Inc, 3111–3119.
27. Cho,K., Van Merriënboer,B., Gulchre,C. *et al.* (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25 October, 2014. Association for Computational Linguistics, 1724–1734.
28. Bengio,Y., Simard,P. and Frasconi,P. (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5, 157–166.
29. Yin,W., Kann,K., Yu,M. *et al.* (2017) Comparative study of cnn and rnn for natural language processing. *CoRR*, abs/1702.01923.
30. Hendrickx,J., Kim,S.N., Kozareva,Z. *et al.* (2009) Semeval-2010 task 8: multi-way classification of semantic relations between pairs of nominals. In: Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions, Boulder, Colorado, 4 June, 2009. Association for Computational Linguistics, 94–99.
31. Lim,S., Lee,K. and Kang,J. (2018) Drug–drug interaction extraction from the literature using a recursive neural network. *PLoS One*, 13, e0190926.
32. Zhou,D., Miao,L. and He,Y. (2018) Position-aware deep multi-task learning for drug–drug interaction extraction. *Artif. Intell. Med.*, 87, 1–8.
33. Liu,S., Chen,K., Chen,Q. *et al.* (2016) Dependency-based convolutional neural network for drug–drug interaction extraction. In: 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 1074–1080.
34. Quan,C., Hua,L., Sun,X. *et al.* (2016) *Multichannel convolutional neural network for biological relation extraction*. Biomed Res. Int., 2016, 1–10.
35. Zhao,Z., Yang,Z., Luo,L. *et al.* (2016) Drug–drug interaction extraction from biomedical literature using syntax convolutional neural network. *Bioinformatics*, 32, 3444–3453.
36. Peng,Y., Rios,A., Kavuluru,R. *et al.* (2017) Chemical–protein relation extraction with ensembles of SVM, CNN, and RNN models. In: Proceedings of the BioCreative VI Workshop, 147–150.
37. Charniak,E. and Johnson,M. (2005) Coarse-to-fine N-best parsing and maxent discriminative reranking. In: Proceedings of ACL, 173–180.
38. de Marneffe,M.-C., MacCartney,B. and Manning,C.D. (2006) Generating typed dependency parses from phrase structure parses. In: Proceedings of the LREC-2006, 449–454.
39. Andor,D., Alberti,C., Weiss,D. *et al.* (2016) Globally normalized transition-based neural networks. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, August 7–12, 2016. Association for Computational Linguistics, 2442–2452.
40. McClosky,D. (2010) Any domain parsing: automatic domain adaptation for natural language parsing. Ph.D. Thesis. Brown University.
41. Tsochantaridis,I., Joachims,T., Hofmann,T. *et al.* (2005) Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6, 1453–1484.
42. Kim,J.-D., Ohta,T., Pyysalo,S. *et al.* (2009) Overview of BioNLP'09 Shared Task on Event Extraction. In: *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task, pages 1–9, Boulder, Colorado*. Association for Computational Linguistics, Boulder, Colorado, 1–9.
43. Björne,J., Kaewphan,S. and Salakoski,T. (2013) UTurku: drug named entity recognition and drug–drug interaction extraction using svm classification and domain knowledge. Second Joint Conference on Lexical and Computational Semantics (*SEM). In: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2, 651–659.
44. Knox,C., Law,V., Jewison,T. *et al.* (2011) Drugbank 3.0: a comprehensive resource for omics research on drugs. *Nucleic Acids Res.*, 39, 1035–1041.
45. Mehryary,F., Björne,J., Pyysalo,S. *et al.* (2016) Deep learning with minimal training data: TurkuNLP entry in the BioNLP Shared Task 2016. In: Proceedings of the 4th BioNLP Shared Task Workshop, 71–81.
46. Mehryary,F., Hakala,K., Kaewphan,K. *et al.* (2017) End-to-end system for bacteria habitat extraction. In: Proceedings of BioNLP 2017, 80–90.
47. Cai,R., Wang,H. and Zhang,X. (2016) Bidirectional recurrent convolutional neural network for relation classification. In: Proceedings of ACL, 756–765.
48. Xu,Y., Mou,L., Li,G. *et al.* (2015) Classifying relations via long short term memory networks along shortest dependency paths. In: Proceedings of EMNLP, 1785–1794.

49. Bunescu,R.C. and Mooney,R.J. (2005) A shortest path dependency kernel for relation extraction. In Proceedings of HLT-EMNLP. 724–731.
50. Quirk,C. and Poon,H. (2017) Distant supervision for relation extraction beyond the sentence boundary. In: Proceedings of EACL. 1171–1182.
51. Pyysalo,S., Ginter,F., Moen,H. *et al.* (2013) Distributional semantics resources for biomedical text processing. In: Proceedings of LBM 2013. 39–44.
52. Srivastava,N., Hinton,G.E., Krizhevsky,A. *et al.* (2014) Dropout: a simple way to prevent neural networks from overfitting. *Mach. Learn. Res.* 15, 1929–1958.
53. Schuster,M. and Paliwal,K.K. (1997) Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45, 2673–2681.
54. Peng,Y. and Lu,Z. (2017) Deep learning for extracting protein-protein interactions from biomedical literature. In: Proceedings of the BioNLP 2017 Workshop. 29–38.
55. Corbett,J. and Boyle,J. (2017) Improving the learning of chemical-protein interactions from literature using transfer learning and word embeddings. In: Proceedings of the BioCreative VI Workshop. 180–183.
56. Lim,S. and Kang,J. (2017) Chemical–gene relation extraction using recursive neural network. In: Proceedings of the BioCreative VI Workshop. 190–193.
57. Lung,P.-Y., Zhao,T., He,Z. *et al.* (2017) Extracting chemical-protein interactions from literature. In: *Proceedings of the BioCreative VI Workshop*, 159–162.
58. Matos,S. (2017) Extracting chemical–protein interactions using long short-term memory networks. In: Proceedings of the BioCreative VI Workshop. 151–154.
59. Tai,K.S., Socher,R. and Manning,C.D. (2015) Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, July 26–31, 2015. 5 Association for Computational Linguistics, 1556–1566.
60. Chollet,F. (2015) Keras. <https://github.com/fchollet/keras>.
61. Al-Rfou,R., Alain,G., Almahairi,A., *et al.* (2016) Theano: A Python framework for fast computation of mathematical expressions. CoRR, abs/1605.02688.

**Farrokh Mehryary, Hans Moen, Tapio Salakoski, Filip Ginter
Entity-Pair Embeddings for Improving Relation Extraction in
the Biomedical Domain**

Proceedings of the 28th European Symposium on Artificial Neural
Networks, Computational Intelligence and Machine Learning, ESANN 2020,
i6doc.com publication, Online conference,
2020, 613-618

Entity-Pair Embeddings for Improving Relation Extraction in the Biomedical Domain

Farrokh Mehryary^{1,2}, Hans Moen¹, Tapio Salakoski¹, Filip Ginter¹

1- Turku NLP Group, Department of Future Technologies,
University of Turku, Turku, Finland

2- University of Turku Graduate School (UTUGS), Turku, Finland

Abstract. We introduce a new approach for training named-entity pair embeddings to improve relation extraction performance in the biomedical domain. These embeddings are trained in an unsupervised manner, based on the principles of distributional semantics. By adding them to neural network architectures, we show that improved F-Scores are achieved. Our best performing neural model which utilizes entity-pair embeddings along with a pre-trained BERT encoder, achieves an F-score of 77.19 on CHEMPROT (Chemical-Protein) relation extraction corpus, setting a new state-of-the-art result for the task.

1 Introduction

The significant amount and the increasing publication rate in the biomedical domain make it difficult for biomedical researchers to acquire and maintain all information that is necessary for their research. Biomedical relation extraction systems aim to address this problem. These systems can periodically scan the whole publicly available literature (PubMed article abstracts and PubMed Central Open Access (PMCOA) full article texts) and extract relations and interactions of biomedical named entities from the texts and build up-to-date relation databases or molecular interaction networks to facilitate biomedical research. A number of shared task challenges have been organized to promote the development and evaluation of such systems. For example, the BioCreative VI shared task [1] was recently organized, which provided the CHEMPROT corpus for chemical-protein relations extraction. Since the CHEMPROT corpus is fairly new, relatively large and carefully annotated, it has become an important benchmark for evaluating modern relation extraction systems.

So far the best results for relation extraction have been achieved using system ensemble approaches. On the CHEMPROT corpus, the best result during the BioCreative VI challenge was obtained by Peng et al. [2] (an F-score of 64.10) by using an ensemble system consisting of a recurrent neural network system, a convolutional neural network system and a support vector machine system. Recently, the introduction of transformer-based language representation models such as BERT [3], impacted the field and resulted in unprecedented jumps in F-score on many data sets. The state-of-the-art result on CHEMPROT is recently achieved by Lee et al. [4] (an F-score of 76.46), by pre-training the BERT encoder on PubMed sentences and fine-tuning it with a decision layer on the CHEMPROT data for relation extraction. This 12 percentage points

increase in F-score is substantial and raises the question to what extent further improvements on top of BERT can be achieved.

Biomedical literature includes a lot of information about the relations and interactions of biomedical named entities (e.g. genes, proteins, chemicals, and drugs). We aim to leverage this literature-wide information using unsupervised methods and for every unique named-entity pair (E_i, E_j) , capture all stated information about E_i and E_j and their relations and build embeddings (vector representations) of entities and entity pairs. Similarly to word2vec [5] for ordinary words, our objective is for similar proteins, chemicals and protein-chemical pairs to obtain similar embeddings. We are especially interested in investigating the possible effects of incorporating these entity and entity-pair embeddings into neural models, in order to improve the performance in relation extraction tasks in the biomedical domain. Given that manually annotated training data for such tasks is usually limited, the domain is an obvious target for transfer learning through pre-trained embeddings. We hypothesize that by pre-training and incorporating the entity-pair embeddings into neural networks, we can improve the performance in relation extraction tasks at hand, potentially better than using individual entity embeddings alone.

In this paper, we explore different approaches for pre-training vector representations for biomedical entities and entity pairs. We concentrate on the chemical and protein named entities in the CHEMPROT corpus and train different types of entity and pair embeddings. We show that when these embeddings are added into BERT-based neural architectures, they can boost the performance of relation extraction. Our approaches are inspired by the work of Levy and Goldberg [6], using richer contexts to extend the skip-gram architecture of word2vec model introduced by Mikolov et al. [5].

2 Method

We propose to pre-train embeddings for named entities and named-entity pairs using a word2vec skip-gram style training, whereby the named entities, or entity pairs are given as the focus terms, and elements from their contexts are predicted. We will investigate two ways to define the context: a simple linear context of words as in the base word2vec, and as an alternative a rich set of features extracted from the context. These features have previously been shown to be useful in supervised relation extraction and one might therefore expect they result in embeddings informative for relation extraction. More specifically, we will rely on the Turku Event Extraction System (TEES) [7] to generate these features, a system that has achieved numerous top ranks in biomedical relation extraction tasks.

2.1 Entity and entity-pair embeddings pre-training

We obtain the list of all chemical-protein pairs in the CHEMPROT corpus and find all sentences in PubMed and PMCOA texts [8] that contain at least one pair. For simplicity, we use exact matching approach when searching for the entities

in the texts. We then extract a set of features for each pair using the TEES system, including (1) word/lemma/POS-tag and dependency-type N-grams along the shortest path connecting the two entities in the sentence dependency parse graph, (2) word/lemma/part-of-speech N-grams of the words that are located within [-3,+3] words of the two entities, and (3) type and location of all biomedical entities occurring in the sentence. We use the word2vecf toolkit [6] for training the embeddings and use either surrounding words as the context or TEES features. Since simultaneous training of embeddings for entities and entity pairs can impact the final model (due to the shared output layer in the word2vec model), we train separate embedding models that include only entity pairs, only entities, or both pairs and entities, resulting in six models (see Table 1).

| Model | Content | Context for training |
|----------|----------------------------|--|
| P_TEEES | only pair embeddings | TEES features |
| P_Words | only pair embeddings | union of the words surrounding the two entities |
| E_TEEES | only entity embeddings | union of TEES features for every pair that includes the entity |
| E_Words | only entity embeddings | words surrounding the entity |
| PE_TEEES | pair and entity embeddings | TEES features |
| PE_Words | pair and entity embeddings | surrounding words |

Table 1: Description of the different embedding models.

2.2 Relation extraction with entity and entity-pair embeddings

We incorporate the pre-trained embeddings into the following neural network architectures: (1) **BERT_MASK**: this architecture is developed by Lee et al. [4] and has achieved the state-of-the-art on CHEMPROT corpus. A BERT encoder pre-trained on PubMed sentences is fine-tuned on the CHEMPROT training set with a decision layer for relation extraction task. This layer predicts one of the five possible relation types between the two entities, or a negative label for no relation. We replicate the method as well as use the pre-trained BERT model of Lee et al. [4]. In the **BERT_MASK** method, the entities are replaced with pre-defined tags (e.g. @PROTEIN\$) to inform the classifier where the two entities are located in a sentence; (2) **BERT_MARK**: the **BERT_MASK** model hides all information about the two entities in the sentence as a consequence of its masking strategy. Since the entity and pair embedding vectors we pre-train provide information about the entities, an improvement on top of the **BERT_MASK** model might be due to this fact. Therefore, as a fairer baseline, we introduce the **BERT_MARK** model (identical to the **BERT_MASK**) except we mark the two entities using the special “unused” symbols in BERT vocabulary¹ (e.g. [unused1]17 β -estradiol benzoate[unused2]); (3) **BERT+Pairs**: this model is similar to the **BERT_MARK**

¹This provided better results compared to using normal characters to mark entity spans (which was used by Lee et al. [4]) since the pre-trained BERT has no notion of the unused symbols.

model, except the pair embedding vector is concatenated to the BERT sentence representation vector ($[CLS]$ token), transformed through a 1024-dimensional dense layer with *tanh* activation, and then presented to the decision layer. The dense layer with the non-linear activation function learns to combine BERT features with the pair embedding features; (4) **BERT+Entities**: this model is similar to the **BERT+Pairs** model, except we concatenate the chemical and the protein embedding vectors (not the pair vector); (5) **BERT+Pairs+Entities**: This model is similar to previous models, except we concatenate chemical, protein, and pair vectors. In all models, we use the exact hyper-parameters used by Lee et al. [4] and optimize the learning-rate by grid search on the development set.

3 Evaluation and results

We evaluate all approaches on the CHEMPROT corpus which contains 4,157 training examples, 2,416 examples in the development set and 3,458 examples in the test set. Chemical-protein pairs can have one of 5 positive relations (e.g. up-regulation) or no interaction at all (negative). We use the official evaluation script provided by the task organizers which calculates the micro-averaged F-score of the positive classes as the task metric. Since initial random weights of a neural model can slightly impact the final F-score, we repeat each experiment (training on the training set and predicting development or test set) for 10 times which results in obtaining 10 F-scores for each approach. We report the average and standard deviation of the F-scores. We use the two-tailed two-sample independent t-test (Welch's t-test) to establish statistical significance.

3.1 Model selection

Table 2 summarizes the results on the development set, reporting statistical significance at $p = 0.1$. **BERT_MARK** outperforms **BERT_MASK**, suggesting that marking should be preferred over the masking approach. In fact, based on column G1, all models that utilize marking (rows 2-9) outperform the approach of Lee et al. [4]. However, as discussed previously, for us **BERT_MARK** is considered the baseline and as column G2 shows, the models that used only entity embeddings (rows 3,4), do not achieve statistically better results than the baseline. Similarly, the model that used pairs (trained with words contexts, row 5) does not achieve a better result, in contrast to the model that used pair embeddings (trained with TEES context, row 6). All models that utilized pre-trained entity *and* pair embeddings (rows 7,8) achieve statistically better results than the baseline. We further conduct another experiment and test the effect of randomly initializing entity and pair embeddings instead of using pre-trained embeddings, to check if the neural model can efficiently learn these embeddings from scratch. However, this model is not able to outperform the baseline (row 9). Thus we conclude pre-training embeddings on the literature is indeed useful. Based on these development set results, only 3 approaches outperform the baseline (rows 6-8).

| # | Neural model | Embeddings model | F-score (mean) | F-score (std) | G1 | G2 |
|---|---------------------|----------------------|----------------|---------------|-----|-----|
| 1 | BERT_MASK | - | 78.41 | 0.53 | - | Yes |
| 2 | BERT_MARK | - | 78.96 | 0.41 | Yes | - |
| 3 | BERT+Entities | E_Words | 79.23 | 0.43 | Yes | No |
| 4 | BERT+Entities | E_TEES | 79.15 | 0.42 | Yes | No |
| 5 | BERT+Pairs | P_Words | 79.27 | 0.43 | Yes | No |
| 6 | BERT+Pairs | P_TEES | 79.36 | 0.32 | Yes | Yes |
| 7 | BERT+Pairs+Entities | PE_Words | 79.47 | 0.37 | Yes | Yes |
| 8 | BERT+Pairs+Entities | PE_TEES | 79.55 | 0.40 | Yes | Yes |
| 9 | BERT+Pairs+Entities | Randomly initialized | 79.05 | 0.46 | Yes | No |

Table 2: Results on CHEMPROT development set. Columns G1 and G2 show if based on the statistical test, the F-score mean is significantly different from the F-score mean of BERT_MASK and BERT_MARK models respectively.

3.2 Final evaluation

We compare our best models selected on the development set (rows 6-8 in Table 2) with the best previous result of Lee et al. [4] (an F-score of 76.46) on the test set. To assess the statistical significance, we use the one-sample t-test ($p = 0.05$). We also evaluate the BERT_MASK model to check how well we have been able to replicate the method of Lee et al. [4].

| # | Neural model | Embeddings model | F-score (mean) | F-score (std) | G1 |
|---|---------------------|------------------|----------------|---------------|-----|
| | Lee et al. [4] | - | 76.46 | - | - |
| 1 | BERT_MASK | - | 76.41 | 0.72 | No |
| 2 | BERT+Pairs | P_TEES | 77.13 | 0.53 | Yes |
| 3 | BERT+Pairs+Entities | PE_Words | 76.71 | 0.77 | No |
| 4 | BERT+Pairs+Entities | PE_TEES | 77.19 | 0.49 | Yes |

Table 3: Results on CHEMPROT test set. Column G1 shows if based on the statistical test, F-score mean is significantly different from the F-score of Lee et al. [4].

The test set results (Table 3) validate our replication of the Lee et al. method (row 1). The model that uses surrounding words as the context (row 3) does not outperform the baseline (at $p = 0.05$), however the models that use TEES features (rows 2,4), outperform the best previous result, suggesting that pair-embeddings with rich feature-based context can improve upon a strong BERT-based baseline. Our best model (row 4) sets a new state-of-the-art for the task, improving the best previous score by 0.73 percentage points.

4 Conclusion and future work

We compared different approaches for pre-training entity and entity-pair embeddings to improve relation extraction performance in the biomedical domain. We have shown that (1) incorporation of these embeddings into neural models helps in achieving better performance, (2) using rich features as context (instead of using the surrounding words, i.e. the normal word2vec approach) leads to better results; (3) using pair embeddings with/without entity embeddings leads to better results compared to using entity embeddings alone. Our best model achieves an F-score of 77.19, improving the best previous result by +0.73pp over a strong baseline, and setting a new state-of-the-art for the task. As future work, we aim to investigate the effect of entity and entity-pair embeddings on other biomedical relation extraction data sets.

5 Acknowledgements

We would like to thank Dr. Sampo Pyysalo for his invaluable recommendations and Dr. Jari Björne for his help in running TEES software. The research was partly funded by the Finnish Cultural Foundation and Academy of Finland (315376).

References

- [1] Martin Krallinger, Obdulia Rabal, Saber A Akhondi, Martín Pérez-Pérez, Jesus Santamaría, et al. Overview of the BioCreative VI chemical-protein interaction track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, volume 1, pages 141–146, Bethesda, MD, USA, 2017.
- [2] Yifan Peng, Anthony Rios, Ramakanth Kavuluru, and Zhiyong Lu. Extracting chemical-protein relations with ensembles of SVM and deep learning models. *Database*, 2018, 07 2018.
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [4] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 09 2019.
- [5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates Inc., 2013.
- [6] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308, 2014.
- [7] Jari Björne. *Biomedical Event Extraction with Machine Learning*. PhD thesis, TUCS Dissertations, 2014.
- [8] Kai Hakala, Suwisa Kaewphan, Tapio Salakoski, and Filip Ginter. Syntactic analyses and named entity recognition for PubMed and PubMed Central –up-to-the-minute. In *Proceedings of the 2016 Workshop on Biomedical Natural Language Processing*, pages 102–107. Association for Computational Linguistics, 2016.



**TURUN
YLIOPISTO**
UNIVERSITY
OF TURKU

ISBN 978-951-29-8743-6 (PRINT)
ISBN 978-951-29-8744-3 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)