Normalization of Disease Mentions with Convolutional Neural Networks

Master's Thesis
University of Turku
Department of Future Technologies
2019
Li-Hsin Chang
Supervisors: Filip Ginter, Kai Hakala

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

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

LI-HSIN CHANG: Konvoluutioneuroverkkojen käyttö tautimainintojen normalisoinnissa

Pro gradu, 95 s., 0 liites.
Turku NLP Group
May 2019

Tautimainintojen normalisoinnilla on tärkeä rooli BioNLP-alan (Biomedical Natural Language Processing) sovelluksissa, kuten esimerkiksi biolääketieteellisten tietokantojen rakentamisessa. Tautimainintojen normalisointiin on ehdotettu useita menetelmiä, mutta ne joko pohjautuvat ehdokaskonseptien tuottamiseen tai eivät yleisty uusiin konsepteihin, jotka eivät esiinny opetusdatassa.

Tässä pro gradu-tutkielmassa tutkitaan, onko mahdollista kehittää tautimainintojen normalisointijärjestelmää, joka yleistyy opetuksessa näkymättömiin konsepteihin ilman ehdokaskonseptien tuottamiseen luottamista. Tätä varten oletetaan, että nykyaikaiset neuroverkot ovat tarpeeksi monimutkaisia tämän ongelman ratkaisemiseen. Tätä oletusta testataan rakentamalla normalisointijärjestelmä syväoppimismenetelmillä ja arvioimalla tämän järjestelmän tarkkuutta käyttämällä NCBI:n (National Center for Biotechnology Information) tautikorpusta. Tämä järjestelmä hyödyntää biolääketieteellisissä aineistoissa olevaa semanttista tietoa käyttämällä jatkuvan vektoriavaruuden representaatioita esittämään tautimainintojen ja konseptien merkkijonoja. Neuroverkkoa opetetaan muodostamaan merkkijonojen vektorirepresentaatiot. Teoriassa tämä opetettu neuroverkko mahdollistaa mallin opetuksen aikana näkymättömiin konsepteihin yleistämistä. Muodostettuja merkkijonoja käytetään vertaamaan konseptien ja tarkasteltavan maininnan samankaltaisuutta. Konsepti, jonka laskettu samankaltaisuus on korkeinta, valitaan ennustetuksi konseptiksi.

Kun järjestelmää kehitetään, synteettistä dataa käytetään helpottamaan mallin opetusta esiopetuksen aikana. Lisäksi tutkitaan eri neuroverkkoarkkitehtuureja. Vaikka malli onnistuu ennustamaan ilman ehdokaskonseptien tuottamista, sen suorituskyky ei kuitenkaan ole viimeisimpien vastaavien menetelmien tasolla. Tautimainintojen normalisointi ilman ehdokkaiden tuottamista samalla säilyttäen mallin yleistyvyys entuudestaan näkemättömiin konsepteihin ei ole triviaalia. Tuleva tutkimus voisi keskittyä esimerkiksi muiden neuroarkkitehtuurien testaamiseen ja monimutkaisempien sanarepresentaatioiden käyttämiseen.

Asiasanat: Biolääketieteellinen tekstinlouhinta, BioNLP, bioinformatiikka, nimettyjen entiteettien normalisointi, konvoluutioneuroverkot, syväoppiminen, tautimaininnat

UNIVERSITY OF TURKU
Department of Future Technologies

LI-HSIN CHANG: Normalization of Disease Mentions with Convolutional Neural Networks

Master's Thesis, 95 p., 0 app. p.
Turku NLP Group
May 2019

Normalization of disease mentions has an important role in biomedical natural language processing (BioNLP) applications, such as the construction of biomedical databases. Various disease mention normalization systems have been developed, though state-of-the-art systems either rely on candidate concept generation, or do not generalize to new concepts not seen during training.

This thesis explores the possibility of building a disease mention normalization system that both generalizes to unseen concepts and does not rely on candidate generation. To this end, it is hypothesized that modern neural networks are sophisticated enough to solve this problem. This hypothesis is tested by building a normalization system using deep learning approaches, and evaluating the accuracy of this system on the NCBI disease corpus. The system leverages semantic information in the biomedical literature by using continuous vector space representations for strings of disease mentions and concepts. A neural encoder is trained to encode vector representations of strings of disease mentions and concepts. This encoder theoretically enables the model to generalize to unseen concepts during training. The encoded strings are used to compare the similarity between concepts and a given mention. Viewing normalization as a ranking problem, the concept with the highest similarity estimated is selected as the predicted concept for the mention.

For the development of the system, synthetic data is used for pre-training to facilitate the learning of the model. In addition, various architectures are explored. While the model succeeds in prediction without candidate concept generation, its performance is not comparable to those of the state-of-the-art systems. Normalization of disease mentions without candidate generation while including the possibility for the system to generalize to unseen concepts is not trivial. Further efforts can be focused on, for example, testing more neural architectures, and the use of more sophisticated word representations.

Keywords: biomedical text mining, BioNLP, bioinformatics, entity linking, normalization, convolutional neural networks, deep learning, disease mentions

Acknowledgement

I would like to thank my supervisors Filip Ginter and Kai Hakala for their guidance throughout this master's thesis project. Filip accepted me into the Turku NLP Group despite my inexperience in programming, and has since been offering inspiring guidance, both academically and in life in general. Despite all the stress in work, he has been patient and allowed me to explore and learn myself. Kai has been helpful in bridging Filip's idea and my understanding of the topic, and has provided assistance at times when I was confused or felt unmotivated.

Couching an inexperienced master's student is probably not the first thing that comes to mind when it comes to being a visiting researcher. I would like to thank Lenz Furrer, who patiently taught me a lot of basic things that I did not know as a biologist by training, and has been helpful in answering questions even after leaving Turku.

Exchange is an enriching studying experience. I would like to thank Joakim Nivre and his research group in Uppsala for taking me in as a visiting student. Their advice has helped me to gain insight into this project.

Regarding the review of this thesis, I would like to thank Sampo Pyysalo, who not only guided me through the final stages of this project under a tight schedule, but also gave detailed and useful feedback on the draft of this thesis.

Having wonderful colleague is something that calls for luck. I would like to thank the amazing group of people in the Turku NLP Group, for their help and for the vibrant, enjoyable workplace atmosphere. In particular, I thank Suwisa Kaewphan and Farrokh Mehryary for being understanding and encouraging, not only in work, but also when it comes to being expats in a foreign place.

I would like to thank Amanda Tursi, my fellow student in the master's programme and best friend in Turku, for starting the study together, for her company, and for her help with writing this thesis.

On the personal level, I would like to thank my friends and family. Thanks especially go to my best friend Chan Hazel Tsz Ham, and my aunts Yu Chi Ching and Yu Szu-Chen, without whose encouragement I would not have come to Finland to study. Seeing one's own kid going abroad has never been easy, and I am grateful to my parents, Chang Chien-Yi and Yu Pei-Chen, for doing so despite their own worries. Finally, a ton of thanks goes to my beloved siblings, Chang Wei-Chi and Chang Chen-Jui, who have done a lot of things in my place and have always supported me in pursuing my own dreams.

In Turku, May 2019
Li-Hsin Chang
張瓅心

TABLE OF CONTENTS

1 Introduction

The amount of information available on the web grows rapidly. This vast amount of information has led to the creation of knowledge bases (KB), or collections of structured information regarding different entities. KBs usually contain not only entities but also how they relate to one another, which can take on forms such as hierarchies. There are a wide variety of KBs, both in the general domain and in specialized domains (e.g. biomedical). The most widely known KB is perhaps Wikipedia[1], while there are other notable KBs more familiar to communities that work with them, such as the general domain KBs DBpedia (Auer et al., 2007) and YAGO (Suchanek, 2007), and the biomedical domain KBs Ensembl (Hubbard et al., 2002) and UniProt (Apweiler et al., 2004). To populate the KBs with knowledge, it is beneficial to extract information related to entities of interest from unstructured data. This can be done by first detecting the spans of entities, or named entity recognition (NER), and then assigning these concepts in KBs to spans. This assignment of spans of entities to KB concepts is known as entity linking, entity grounding, or normalization [2]. In essence, the normalization task resolves semantic ambiguity caused by synonymy and polysemy.

An example of normalization would be the establishment of linkage between the textual mention “Burma” and the Wikipedia concept `Myanmar` in the sentence “Burma is a

[1]`https://en.wikipedia.org/wiki/Main\_Page`

[2]This task is usually referred to in the general domain as entity linking, while in the biomedical domain the term “normalization” is often used instead.

tropical country". This example is illustrated in Figure 1.1[3]. To perform normalization, the mention of the location "Burma" is first identified in the NER step. Subsequently, in the normalization step, the mention "Burma" is to be mapped to the concept identifier of the country `Myanmar` in a given KB such as Wikipedia, instead of other concept identifiers that bear similar surface forms such as the Indian film `Burma`. As can be illustrated by this example, normalization is not always trivial because multiple concepts may have similar or identical surface form(s): in this case, "Burma" may refer to a country name, a film, and also the name of villages in other parts of the world, depending on the context. In addition, one concept may have multiple surface forms, such as "Myanmar" and "Burma" referring to the same country. Furthermore, a mention may not have any corresponding concept in a given KB to begin with.

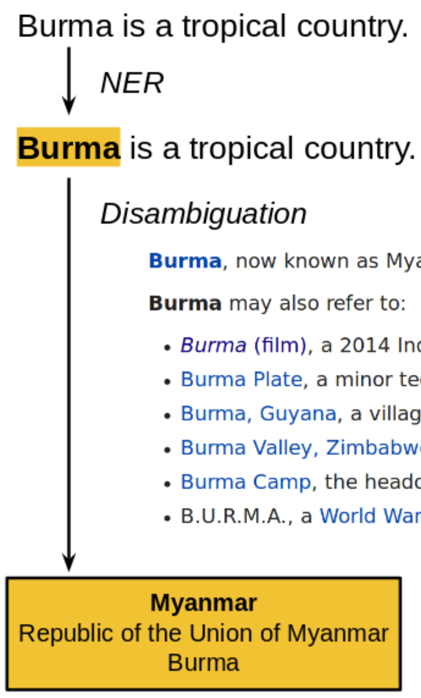

Burma is a tropical country.
NER
Burma is a tropical country.
Disambiguation
Burma, now known as Myanmar, is a country in Southeast Asia.
Burma may also refer to:
• Burma (film), a 2014 Indian film
• Burma Plate, a minor tectonic plate
• Burma, Guyana, a village in Guyana
• Burma Valley, Zimbabwe
• Burma Camp, the headquarters of the Ghana Armed Forces
• B.U.R.M.A., a World War II postal acronym
Myanmar
Republic of the Union of Myanmar
Burma

Figure 1.1: Normalization of 'Burma' to the concept 'Myanmar'.

[3]https://en.wikipedia.org/wiki/Burma_(disambiguation). Accessed May 7 2019.

Normalization and its related tasks, such as Word Sense Disambiguation[4], have been studied by different research communities. These communities include that of Natural Language Processing, Data Mining, and Semantic Web. The fact that normalization is of interest to a variety of research communities is unsurprising, as resolving ambiguities in text benefits research areas such as Information Extraction, Information Retrieval, and Machine Translation. For example, in the biomedical domain, normalization can be used to link mentions of entities of interest (e.g. genes and proteins) in the literature to identifiers in databases. This facilitates the extraction of information of scientific interest (e.g. protein phosphorylation). In this thesis, special attention is given to normalization in the biomedical domain, specifically normalization of disease mentions. However, normalization as a research area is briefly covered in Chapter 2.

In the biomedical domain, various biomedical entities have become the focus of normalization: disease names, gene names, chemicals, bacteria biotopes, etc. However, some challenges appear more frequently in this domain of normalization than in the general domain. Among these challenges are term variation and acronym resolution. Term variation refers to one concept having multiple surface forms, some of which are very different from one another. For example, "carcinoma" and "cancer" both refer to abnormal growth of body cells that may metastasize. Acronym resolution refers to the disambiguation process when different concepts share the same surface form. For instance, "KMS" can refer to `Kabuki make up syndrome` or `Kallmann syndrome`, which are completely unrelated disorders. For disease mention normalization, various systems have been developed to cope with these challenges.

[4]In computational linguistics, Word Sense Disambiguation (WSD) is a task that aims to identify the sense of a word. For example, for the word "spring" in the sentence "Spring is around the corner", WSD tries to determine which sense of "spring", the sense of a season, that of a helical metal coil, or that of a sudden movement forward, is referred to.

In the following chapter (Chapter 2), an overview of related work on the topic of normalization is presented. This overview includes recent work on general domain normalization, as well as biomedical normalization, with a special focus on disease names. After the current research involving normalization has been described, the motivation and objectives behind building a disease mention normalization system are further explained in Chapter 3. The data and the main methods behind the implementation are covered in Chapter 4, while the implementation details of the model architecture are described in Chapter 5. For the experiments carried out in this study, the experimental settings are described in Chapter 6. The reasoning behind the experimental settings, the results, and the analysis are provided in Chapter 7. In Chapter 8, conclusions are drawn and further research directions are suggested.

Throughout this thesis, the term “normalization” refers to the task of assigning a concept identifier to a textual mention. The term “mention” refers to a textual mention of an entity from a corpus. An entry in a controlled vocabulary is called a “concept identifier”, which can have multiple “names”, or terms that refer to it.

2 Related work

Due to its various applications in different domains, diverse types of systems have been developed for the task of normalization. These systems are often dependent on the domains they have been built for. Examples of these domains include the news domain, such as normalization of persons, locations, and organizations, and the biomedical domain, such as normalization of disease mentions and chemicals.

This chapter is divided into three sections. In the first section, the normalization research area is presented by briefly describing common normalization approaches and introducing selected systems. In the second section, the most recent studies on disease mention normalization are introduced. In the third section, selected normalization systems for mentions of biomedical entities other than diseases are introduced.

2.1 Normalization

As a result of substantial research on normalization from different research communities, diverse normalization data sets, systems, evaluation methods, and KBs have been created. This section does not attempt to cover all of these in detail, but tries to give a brief overview. This overview covers the categorization of KBs and approaches employed by normalization systems. No data sets are introduced, but a publicly available framework for normalization system evaluation which includes data sets is mentioned. In addition, selected normalization systems which employ remotely related approaches to this study are introduced.

Knowledge bases can be categorized into entity-centric KBs and document-centric KBs (Zwicklbauer et al., 2015). Entity-centric KBs, such as DBpedia (Auer et al., 2007) and YAGO (Suchanek, 2007), have a set of entities with identifiers and attribute fields. Document-centric KBs, such as Wikipedia, contain a set of natural language documents, each focused on one concept, with links of mentions to other concepts.

Zwicklbauer (2017) gives a detailed review on normalization approaches. Briefly, normalization approaches are divided into three components according to the order they are used in a typical normalization system (Table 2.1): (1) candidate entity generation, (2) disambiguation, and (3) abstaining. In the candidate generation step, candidate concepts from KBs are generated to reduce the number of concepts to be processed subsequently. Methods common to this step include the use of dictionaries, acronym expansion, and the use of search engines. For disambiguation of mentions, traditional machine learning approaches often use hand-crafted features. These features can be, for example, the name, popularity, and type of entities, as well as their textual context, the coherence of topic, etc. Algorithms commonly used for disambiguation include vector space model approaches, information retrieval approaches, learning to rank (LTR) approaches, graph-based approaches, probabilistic approaches, classification approaches, and ensembles thereof. The abstaining step determines whether there is a concept in the given KB that maps to the mention in question in the first place. In practice, this step is often integrated into the disambiguation step by adding in a pseudo-concept identifier `NIL`, which signifies that there is no concept identifier in the given KB that can be assigned to the mention.

Dredze et al. (2010) view normalization as ranking and engineer features for the ranking of candidate concepts. They use a KB-independent approach to generate candidate

Table 2.1: Common entity linking approaches (Zwicklbauer, 2017).

Component	**Approaches**
Candidate entity generation	Name dictionary, surface form expansion, search engine
Disambiguation - features	Entity name, entity popularity, entity type, textual context, topical coherence, joint feature modeling
Disambiguation - algorithms	Approaches based on vector space model, information retrieval, learning to rank, graph-based approaches, probabilistic approaches, classification approaches, and ensemble approaches
Abstaining	Considering `NIL` as additional entity

concepts based on morphological information of entities. A variety of features are extracted from candidates for disambiguation. These features include string-based features (e.g. string equality, partial string match), Wikipedia-derived features (e.g. Wikipedia page length), concept popularity (i.e. how frequently a concept is used), and document features (i.e. the context of mentions). The training of the ranking model maximizes the difference in scores assigned to positive and negative training instances. For abstaining, `NIL` is included as a candidate concept. Its prediction depends on `NIL` features, which include statistics of the prediction scores, and external information such as Google match.

Francis-Landau et al. (2016) use a convolutional neural network (CNN) to extract topic vectors from three granularities of text (i.e., mention, sentence, and document) and compare these vectors with those of concepts by cosine similarity. These cosine similarities are then used, along with other features, for logistic regression to predict the concept identifier for a given mention.

Inan and Dikenelli (2018) propose a two-fold neural model for domain-specific normalization. They represent mentions and concepts by Resource Description Framework (RDF) embeddings, embeddings of entities learned in RDF graphs where each entity is represented by a node (Ristoski and Paulheim, 2016). Their model first matches easy mention-concept pairs, whose domain information is then used to generate candidates for mentions in the vicinity. The more ambiguous pairs are subsequently resolved by bidirectional long short-term memory (Schuster and Paliwal, 1997) and conditional random field (CRF) (Wallach, 2004) models. Their sequence learning model views normalization as a translation task, where a mention is translated into a concept using a sequence-to-sequence model. This system achieves state-of-the-art performance on the Disambiguate to Knowledge Base (D2KB) task (Usbeck et al., 2015), a domain-specific normalization evaluation.

Many metrics have been used for the evaluation of normalization systems due to the wide spectrum of research. Among these metrics are accuracy and recall, precision, and F1-score. This difference in evaluation methods poses a challenge to the comparison of normalization systems. To facilitate normalization system comparison, Usbeck et al. (2015) propose the General Entity Annotator Benchmarking Framework (GERBIL). This framework is publicly available, and allows users to submit prediction results from their own systems on the supported data sets. The GERBIL platform computes the evaluation scores for its users.

2.2 Disease mention normalization

This section reviews disease mention normalization-related work. Commonly used corpora created to facilitate the development of such systems with machine learning approaches are briefly introduced here. The NCBI disease corpus and the ShARe/CLEF corpus are described further in Section 4.1.1. After these corpora have been introduced, a review of selected recent work follows. The approaches adopted by selected systems and their results on the NCBI disease corpus are also organized into a table.

The NCBI disease corpus was constructed by the National Center for Biotechnology Information (NCBI) for the development of disease entity recognition and normalization systems (Dogan et al., 2014). The corpus contains 793 fully annotated PubMed abstracts. The disease entities are annotated with their spans and the corresponding concept identifiers from the MErged DIsease voCabulary (MEDIC) (Davis et al., 2012).

The ShARe/CLEF corpus was originally constructed for the ShARe/CLEF eHealth Evaluation Lab 2013 (ShARe/CLEF 2013) (Suominen et al., 2013), though its test set was expanded in the follow-up shared task SemEval-2014 Task 7 (Pradhan et al., 2014). The vision of ShARe/CLEF 2013 was to facilitate patients' and their next of kin's access to medical information. The corpus consists of 300 de-identified clinical reports with annotation of disease mention spans and identifiers. The disease mentions are annotated with concept identifiers from the controlled vocabulary (i.e. the KB), namely, the SNOMED-CT subset of the Unified Medical Language System (UMLS) Metathesaurus (Schulz and Cornet, 2009). There are two evaluation metrics used for this corpus: strict accuracy and relaxed accuracy. Strict accuracy requires both the predicted spans and concepts to match with the gold standard, while relaxed accuracy allows for overlapping spans with correct predicted concepts.

The BioCreative V Chemical-Disease Relation (BC5CDR) corpus was constructed for the shared task of the same name (Li et al., 2016). The shared task aimed to advance the relation extraction of chemical-induced diseases. The BC5CDR corpus contains 1,500 articles where the disease and chemical entities are annotated with spans and identifiers from Medical Subject Headings (MeSH), National Library of Medicine's controlled vocabulary for literature indexing (Lipscomb, 2000).

Taking advantage of these corpora, various normalization systems for disease mentions have been developed over the past decade. These systems have deployed a variety of approaches, including rule-based approaches, non-neural machine-learning approaches, deep learning approaches, and combinations thereof. While the rule-based approaches were developed earlier, neural networks are the methods featured in the latest publications at the time of writing this thesis. In addition, since normalization is conducted subsequent to NER, and its accuracy is limited by the results of NER, there are also systems that combine these two steps.

MetaMap was developed by the National Library of Medicine (NLM) for biomedical text indexing purposes (Aronson, 2001). It maps biomedical mentions to UMLS concepts by (1) generating lexical and morphological variances of the mentions, (2) retrieving candidate concepts in UMLS containing the variances, and then (3) scoring the candidates using metrics designed based on linguistic principles. MetaMap is a tool of the Medical Text Indexer, which is used for semiautomatic and automatic biomedical literature indexing.

Dogan and Lu (2012b) propose addressing the normalization task from an information retrieval perspective, i.e. disease mentions are regarded as input queries, concept identifiers search results. They built a search engine for disease mentions using the information retrieval software library Lucene[1]. The search results are further processed and ranked; abbreviations are resolved, and candidates are inferred using a set of rules based on their similarity to the control vocabulary terms and the Lucene search scores. This method

[1] http://lucene.apache.org/

achieved an F-score of 0.783 on the NCBI disease corpus.

D'Souza and Ng (2015) apply a multi-pass sieve approach to normalization. Their system features ten sieves, each consisting of heuristic rule(s), ordered at different priorities according to their precisions. These sieves include exact match, abbreviation expansion, number replacement, etc. The system obtained accuracy scores of 0.908 and 0.847 for the ShARe/CLEF corpus and the NCBI disease corpus, respectively.

AuDis is a pipeline architecture for NER and normalization (Lee et al., 2016). NER is performed using CRF model with post-processing steps, and normalization is performed using a dictionary-lookup approach. The system achieved the best performance with an F-score of 0.865 on the disease normalization task in the official evaluation of the CDR task in BioCreative V.

Ghiasvand and Kate (2014) were the second best performing team for the disease normalization sub-task in the SemEval-2014 Task 7 (Pradhan et al., 2014). Their normalization system searches for exact matches and, whenever those are unavailable, considers variations in disorder mentions and learns them using an edit distance pattern-based method. The system achieved a strict accuracy of 0.66 for normalization on the ShARe/CLEF corpus.

While the above mentioned methods achieve reasonable results for disease mention normalization, they rely heavily on complete specialist lexicons and/or heuristics. This is time- and labor-consuming, and makes the application of these systems rather restricted to the given domain and the specific language. These restrictions are alleviated with the use of word embeddings (introduced in Section 4.3.2), which became a popular method for text representation following the research of Mikolov et al. (2013a). At the same time, machine learning-based approaches were developed with the aim of relying less on such resource-demanding processes. The following systems introduced are examples of research in the wake of this increased popularity in machine learning-based approaches.

The top-ranking system in ShARe/CLEF 2013 uses Term Frequency-Inverse Document Frequency (TF-IDF) vector representation (introduced in Section 4.3.1) for disease mentions and concept names, and selects the normalized concept according to the cosine similarity of the vector representations (Tang et al., 2013). The system achieves a strict accuracy of 0.514 and a relaxed accuracy of 0.729. The same research group again obtained top-ranking results in Sem-Eval 2014 Task7. The cosine similarity of vector space representations was used again for disease normalization, and the team achieved a best strict accuracy of 0.741, and a relaxed accuracy of 0.873 (Zhang et al., 2014).

Kaewphan et al. (2014) also use the cosine similarity of the vector space representation of mentions and concept names as their normalization measure. They use the *word2vec* method (Mikolov et al., 2013a) to train word embeddings on billions of tokens from unlabelled biomedical texts on PubMed and PubMed Central (Mikolov et al., 2013b). The resulting embedding (referred to as emb-Kaewphan) is further described in 4.3.2. This system achieved a strict accuracy of 0.601 and a relaxed accuracy of 0.783 in Sem-Eval 2014 Task 7, ranking fifth in the normalization sub-task.

While cosine similarity of vector representations was used in some of the entries of ShARe/CLEF 2013 and Sem-Eval 2014 Task7 to measure the similarity of mentions and concept names, Leaman et al. (2013) use a trainable scoring function to calculate the similarity of vector representations. Their system, DNorm, was the first to apply the IR machine-learning approach pairwise learning to rank (further described in Section 4.4) to the normalization task. It uses the NCBI disease corpus and the MEDIC vocabulary.

DNorm represents mentions and concept names by TF-IDF vectors within a defined vector space (Manning et al., 2008). During training, a scoring function that scores a given mention and a concept name is modelled by a linear function whose weights are contained in a matrix, which is optimized using stochastic gradient descent (Burges et al., 2005). This matrix is subsequently used to rank the concept names with the mentions in a pairwise fashion. DNorm achieves a micro-averaged F-score of 0.782 and a macro-average F-score of 0.809, which are higher than their best baselines by 0.121 and 0.098, respectively.

DNorm's vector representation is based on mentions and concept names, and thus relies heavily on the quality of the labelled data and that of the predefined dictionary. Cho et al. (2017) use a method similar to that of Kaewphan et al. (2014) for disease normalization, except that they use both unlabelled PubMed data and the detected mention spans for the training of *word2vec* word embeddings. Their system obtained an F-score of 0.808 on the NCBI disease corpus, and 0.690 on their manually constructed plant corpus. In addition, the system was reported to outperform the best disease mention normalization system in the BioCreative V challenge.

DNorm, as well as some of the abovementioned entries of ShARe/CLEF 2013 and SemEval 2014 Task7, suffers from cascading of NER error to normalization. To handle this issue, joint models for NER and normalization were proposed. Leaman and Lu (2016) introduce a joint model using a machine learning approach during both training and prediction on biomedical text. Their model, non-restricted to any entity types, features a semi-Markov structured linear classifier. The classifier has features for NER and uses vector space representation similar to that of DNorm for normalization. Their Java implementation of the model, TaggerOne, obtains F-scores of 0.829 and 0.807 for NER and normalization respectively on the NCBI disease corpus.

TaggerOne, however, does not use context features due to the use of exact inference and dynamic programming. Lou et al. (2017) propose a transition-based joint model that is able to leverage context features. This is done by using previous predictions from the same document as features. This model, trained on the BC5CDR corpus, achieves an F-score of 0.826 for normalization on the NCBI disease corpus without corpus-specific fine-tuning.

ter Horst et al. (2017) use undirected graphical models to build a probabilistic system that jointly handles NER and normalization. Their system, JLink, achieves F-scores of 0.859 and 0.884 on disease and chemical mentions respectively on the BC5CDR corpus, out-performing D-Norm and TaggerOne on disease mentions and achieving comparable performance on chemical mentions.

With the rise of popularity of deep learning methods, several neural network-based systems have been developed for disease normalization. Liu and Xu (2018) propose a model incorporating deep learning into the generation of word representations. They train distributed representations on large, unlabelled literature using *word2vec* and integrate individual tokens of a disease mention into the representation of the disease mention using tree-structured long short-term memory (Tree-LSTM) (Tai et al., 2015). The similarity score for a given pair of mention and concept name is calculated by a single perceptron, and the final decision is made using a pairwise learning to rank approach. When trained on the NCBI disease corpus, the system achieves accuracy scores of 0.853 and 0.765 on the NCBI disease corpus and the BC5CDR corpus respectively.

Li et al. (2017), similarly, use the pairwise learning to rank method for the ranking of concepts. Their system generates candidates using modified rules from D’Souza and Ng (2015). Candidates are then represented by word vectors and encoded by a CNN architecture. The encoded mentions and candidates are subsequently ranked according to their morphological and semantic information (as described in detail in Section 5.2.1) . The system obtains accuracy scores of 0.861 and 0.903 on the NCBI disease corpus and the ShARe/CLEF corpus, respectively. This system views normalization as a zero-shot (Socher et al., 2013) multi-class classification[2] problem.

NormCo is a deep coherence model for disease mention normalization (Wright et al., 2019). This model is trained to map disease mentions to a disease concept space, with the goal of minimizing the distance between the vector representation of a mention and its corresponding concept. To achieve this, the model takes into account the semantics of mentions by using phrase embeddings (Hill et al., 2016), and considers topical coherence at the document level. To deal with data sparsity, distantly supervised and synthetic data are used. Compared with existing state-of-the-art systems, the model is more efficient and achieves comparable results with regard to accuracy and F-score. In addition, the falsely predicted concepts are closer in semantic meaning to the gold standard. However, this model, unlike the systems of Li et al. (2017) and Liu and Xu (2018), do not have the potential to generalize to unseen concepts. This is because NormCo does not have an encoder for concepts, and thus views disease normalization as a standard multi-class rather than a zero-shot multi-class classification problem.

Table 2.2 shows selected systems, their approaches, and results on the NCBI disease corpus. It can be seen from this table that the latest state-of-the-art systems are neural network-based.

[2]In supervised machine learning, a classification task involves assigning a category for a data point. Depending on the number of categories, classification can be divided into binary classification and multi-class classification. Whereas binary classification has only two categories, multi-class classification has more than two categories.

Table 2.2: Selected systems, their approaches, and results on the NCBI disease corpus.

Name of system and/or authors	**Approach**	**Results on the NCBI disease corpus**
Dogan and Lu (2012b)	Search engine for disease mentions	0.783 (F-score)
D'Souza and Ng (2015)	Rule-based	0.847 (accuracy)
DNorm (Leaman et al., 2013)	TF-IDF vector representations, a trainable scoring function for similarity measurement	0.782 (micro-averaged F-score) and 0.809 (macro-average F-score)
Cho et al. (2017)	*word2vec* word embeddings, cosine similarity	0.808 (F-score)
TaggerOne (Leaman and Lu, 2016)	TF-IDF vector representations, semi-Markov structured linear classifier	0.807 (F-score)
Lou et al. (2017)	Transition-based joint model	0.826 (F-score)
Liu and Xu (2018)	*word2vec* word embeddings, Tree-LSTM	0.853 (accuracy)
Li et al. (2017)	Heuristics for candidate generation, word embeddings, CNN	0.861 (accuracy)
NormCo (Wright et al., 2019)	Neural networks, phrase embeddings, and topical coherence	0.878 (accuracy)

In summary, disease mention normalization systems can roughly be categorized into rule-based systems and machine learning-based systems. Rule-based systems suffer from the limitations of rules not being directly transferable across different types of mentions, and the impossibility of an exhaustive collection of rules (Li et al., 2017). Existing machine learning-based systems, on the other hand, use various strategies such as candidate concept generation by heuristics, considering both semantic and morphological information, using vector representations of text, leveraging silver standard corpus to overcome data-sparsity, taking into account document-level coherence, etc. Many of the systems, both rule-based and machine-learning based, also rely on the completeness of domain-specific ontologies, which restricts the effective application of such systems to only languages and domains with more complete ontologies. That is, they require a comprehensive collection of concepts to which the mentions can be normalized.

2.3 Normalization of other types of biomedical mentions

Apart from disease normalization, various other types of mentions have been the focus of normalization. For example, several systems have been developed for the normalization of gene names as entries to the BioCreative series (Morgan et al., 2008; Lu et al., 2011). These gene normalization (GN) shared tasks have focused on gene detection and normalization to the standard database NCBI Gene (Maglott et al., 2011). In the most recent GN shared task, the evaluation metrics Threshold Average Precision at a median of k errors per query is used. This measure relates to average precision, and reflects how reliable a GN system is by its output score (Lu et al., 2011).

GNAT Java library is used to process gene and protein mentions (hereafter gene mentions) in biomedical text (Hakenberg et al., 2011). It supports text retrieval, NER, and normalization of gene mentions. It can be used for integration with other systems, be the framework for integration, or as a stand-alone application.

GenNorm ranked top three in the GN task of BioCreative III (Wei and Kao, 2011). It recognizes genes by dictionary lookup and predicts the species they belong to based on a combination of heuristics that takes into account the gene name and associated species. Its stand-alone module, SR4GN, compares favorably to the top-performing systems when evaluated on the benchmark experiments of the BioCreative series (Wei et al., 2012).

GNormPlus handles both NER and gene normalization by integrating multiple biomedical text mining systems, viz., Ab3P (abbreviation resolution), GNR (gene name recognition), SR4GN (species recognition and species assignment), SimConcept (composite mention simplification), and GenNorm (gene name normalization) (Wei et al., 2015).

The normalization of bacteria to taxa and their locations to habitats is another normalization task spurred by shared tasks. The Bacteria Biotope tasks of the BioNLP Shared Task focused on the recognition and normalization of bacteria and their habitats, as well as the extraction of related events (Deléger et al., 2016). The best performing system in the latest edition of this task, BOUN, uses string matching methods for bacterial name normalization, and information retrieval techniques, such as TF-IDF scoring and cosine similarity, for habitat name normalization (Tiftikci et al., 2016). This system achieved an overall precision of 0.679, and precision of 0.801 and 0.620 for bacterial name normalization and habitat name normalization, respectively. Adopting a similar approach as that of Tiftikci et al. (2016) for habitat name normalization, Mehryary et al. (2017) use the cosine similarity of TF-IDF vector space representations for both bacteria and habitat normalization, and obtained precision of 0.816 for bacteria normalization, and 0.630 for habitat normalization, resulting in an overall precision of 0.691.

Ferré et al. (2018) use a combination of rule-based and embedding-based approaches for bacterial habitat normalization. The rule-based approach, ToMap, assumes that the syntactic head of an entity is its most informative token, and maps the entity based on its head (Golik et al., 2011). The embedding-based approach, CONTES, creates an ontological space that takes into account the hierarchical information of the ontology. CONTES then computes embedding representations for mentions and concepts, and minimizes their distance in the ontological space (Ferré et al., 2017). CONTES is used to complement ToMap by using it to predict mentions which cannot be normalized by ToMap. As distant supervision, ToMap is used to generate additional training data from unlabelled biomedical text for CONTES because it is rule-based and has high precision. The supervised and distantly supervised setups achieved precision of 0.73 and 0.72 respectively.

Karadeniz and Özgür (2019) represent biomedical entities by forming representations from word embeddings where the representation of each token is weighted by their syntactic role. The ontological concepts are ranked according to the cosine similarity of their representation with the mentions. This approach, unlike that of Ferré et al. (2017), is unsupervised because the representation of mentions and concepts are computed the same way, eliminating the need to transform the vectors before computation of similarity. This approach attained a precision score of 0.659 for habitat name normalization for the Bacteria Biotope data set, and a macro-average precision score of 0.687 on an adverse drug reaction data set from the Text Analysis Conference 2017 (Roberts et al., 2017).

Apart from gene names, bacterial taxa and habitats, and adverse drug reactions, other biomedical named entities, such as chemicals and organisms, have also been the focus of normalization systems. The tmChem system is an NER system for chemicals formed by the ensemble of two CRF models. The system is paired with a normalization system, which uses a dictionary lookup approach to assign chemical mentions to MeSH or ChEBI[3] (de Matos et al., 2010) identifiers (Leaman et al., 2015). OrganismTagger is a hybrid rule-based and machine-learning system that detects mentions of organisms (Naderi

[3]Chemical Entities of Biological Interest (ChEBI) is a database for chemical compounds.

et al., 2011). In addition, it resolves abbreviations and acronyms, and normalizes mentions to the NCBI Taxonomy database. The system achieves high normalization accuracy scores of 0.975 on the authors' own manually annotated corpus, the OT corpus, and 0.974 on the Linnaeus-100 corpus (Gerner et al., 2009).

Various approaches, supervised and unsupervised have been devised for normalization of different types of biomedical named entities. These approaches can be dictionary- or vector space representation-based. In comparison with the systems that have been proposed for disease mention normalization, the systems that have been developed for bacteria habitat normalization are the most similar in terms of the strategy of using vector space representations and measuring similarity between mentions and concepts. This is perhaps due to the fact that both of these types of mentions share semantic similarity with their corresponding ontological concepts, whereas other types of biomedical mentions, such as gene names[4] and chemicals, require different flavors of normalization that are not captured by biomedical word embeddings.

[4]Gene normalization requires tailored solutions mainly due to systematic species ambiguity.

3 Motivation and objectives

As introduced in Section 2.2, state-of-the-art systems for disease mention normalization are based on neural networks. There are, however, some shortcomings to the existing systems that we would like to address in this study. One shortcoming is the use of candidate generation, i.e., pre-selection of candidates from the controlled vocabulary by heuristics. This is the approach adapted by Li et al. (2017). Another shortcoming concerns the ability of the model to generalize to unseen concepts during training. This shortcoming is present in the system of Wright et al. (2019), which learns a space for disease mentions and concepts.

This thesis explores the possibility of building a normalization system that generalizes to unseen concepts and does not rely on candidate generation. This is done by postulating that modern neural networks have the computational power to serve as building blocks for such normalization systems. More specifically, the following three hypotheses are made:

(1) Given the baseline method of averaging the word embeddings (introduced in Section 4.3.2) from each token in a given multi-token string as the representation of mentions and concept names, neural networks, such as convolutional neural networks, are sufficiently sophisticated to learn to encode entities in a way that leads to better normalization performance than this baseline method.

(2) The encodings from the aforementioned neural encoder allow ranking of identifier concepts in a way that does not lead to performance degeneration of the normalization system when given the whole set of possible concepts to rank, as opposed to being given only a small number of generated candidates.

(3) A normalization architecture containing the aforementioned neural encoder that is able to generalize to previously unseen concepts is possible.

To test these hypotheses, an entity linking system is built and evaluated on a corpus for disease name recognition and normalization, the NCBI disease corpus. The following chapter introduces the main methods used for constructing the normalization system.

4 Methods

The theoretical background underlying the proposed normalization system is introduced in this chapter. It includes vector space representation for word representation, pairwise learning to rank for ranking, and neural network as the backbone of the model. In addition, the corpus and the controlled vocabulary used are introduced.

4.1 Data

As the benchmarking data set of the system, the NCBI disease corpus and the controlled vocabulary it uses, MEDIC, are selected. This section describes these data and some of their general statistics.

4.1.1 The NCBI disease corpus

The NCBI disease corpus was created to provide resources for the development of disease NER and normalization systems (Dogan and Lu, 2012a). The corpus consists of 793 PubMed titles and abstracts, where all the spans of disease names are annotated and mapped to identifier concepts in MEDIC. In total, there are 6,900 disease mentions in the corpus, and 2,161 of them are unique.

An example sentence extracted from this corpus is shown in Figure 4.1. This sentence has four annotations of mention spans, namely, “colon cancers’, “adenomatous polyposis

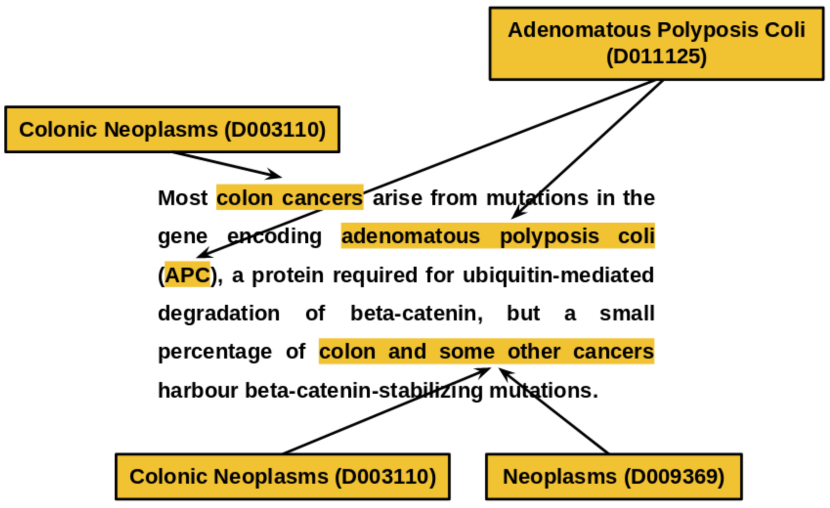

Adenomatous Polyposis Coli (D011125)
Colonic Neoplasms (D003110)
Most colon cancers arise from mutations in the gene encoding adenomatous polyposis coli (APC), a protein required for ubiquitin-mediated degradation of beta-catenin, but a small percentage of colon and some other cancers harbour beta-catenin-stabilizing mutations.
Colonic Neoplasms (D003110)
Neoplasms (D009369)

Figure 4.1: A sentence with annotations from the NCBI disease corpus.

coli”, “APC”, and “colon and some other cancers”. Each of these mentions is mapped to one or multiple concept identifiers. For example, the mention “colon cancers” is mapped to the concept with the identifier `D003110` and the preferred term “Colonic Neoplasms”.

As shown in the example, some mentions are mapped to multiple concept identifiers. There are not many of them in the corpus; only 71 mentions, or 9% of the mentions, are mapped to more than one concept identifier. Of these, 52 are composite mappings, or mappings to multiple concepts due to sentence structure, and the concepts are separated by “|” in the annotation. For example, “breast and ovarian cancer” is annotated as `D001943 | D010051`, where `D001943` is the concept identifier for `Breast Neoplasms` and `D010051` that of `Ovarian Neoplasms`. The other ones are multiple concept mappings, or mappings to multiple concepts due to the inherent meaning of the disease mention, and these concepts are concatenated with “+”. For instance, “inherited muscle disorder” is annotated as `D009135 + D030342`, where `D009135` is the concept identifier for `Muscular Diseases` and `D030342` that of `Genetic Diseases, Inborn` (Dogan et al., 2014; Davis et al., 2017). In terms of the distribution of these multi-concept mentions in the training, development, and test sets, they occur 115, 30, and 15 times in these three sets, accounting for 2.24%, 3.81%, and 1.56% of the mentions respectively.

4.1.2 The MEDIC disease vocabulary

MEDIC is maintained for curational purposes by Comparative Toxicogenomics Database (CTD), an organization that facilitates understanding of the effect of environmental exposure on human health. To achieve this, CTD provides a manually curated, publicly available database of chemical-gene/protein interactions, gene-disease and chemical-disease relationships (Davis et al., 2017). MEDIC was created by merging Online Mendelian Inheritance in Man (OMIM), a controlled vocabulary for human genetic diseases, and some parts of MeSH. OMIM is a flat list of genetic disease-related concepts that includes phenotypes, phenotype descriptions, genes with phenotypes, etc. MeSH is a controlled vocabulary thesaurus organized hierarchically into 16 branches. Of these branches, the `Disease` branch and the `Mental Disorders` sub-branch of the `Psychiatry and Psychology` branch contain disease concepts relevant for the MEDIC construction. The creation of MEDIC saw the mapping of concepts in OMIM into the `Disorders` and `Mental Disorders` (sub)branches of MeSH. OMIM concepts with synonyms in MeSH were merged with their synonyms, and those without were made leaves of appropriate MeSH concepts. This use of the MeSH hierarchy as the backbone allows users to navigate through broader and narrower concepts (Nelson et al., 2001; Davis et al., 2012).

MEDIC is updated monthly. The version of MEDIC disease vocabulary used in this study is the one used by DNorm[1] (Davis et al., 2012; Leaman et al., 2013). In this version used, there are 9,664 disease concepts, with a total of 67,782 names. That is, on average, every disease concept has 7.01 names, including the preferred term and their synonyms. 91% of disease concepts have synonym(s) in addition to their preferred terms, and 47%

[1]There are two MEDIC versions that come with downloaded DNorm, one has 9,664 disease concepts, the other 11,885. The larger one was used for the ShARe/CLEF 2013 shared task.

of the concepts have definitions, or descriptions of these concepts.

An example concept from MEDIC is shown in Figure 4.2. This concept has the identifier `MESH:D009369`, which shows that it originally comes from MeSH. The concept has the preferred term `Neoplasms` and nine other terms listed as synonyms. It can be seen that the synonym terms can be alternative names of the preferred term, morphological variants of the preferred term and other synonyms, as well as syntactic variants, such as `Neoplasm, Benign` and `Benign Neoplasm`. In some cases, it can also be the acronym of other names of the concept, or names with adjective modifiers.

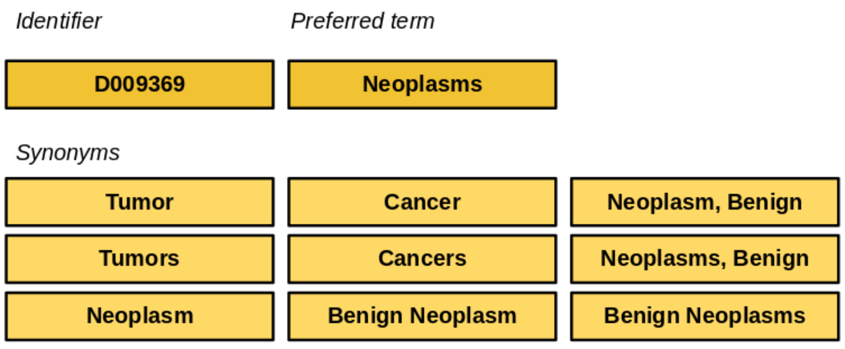


Figure 4.2: A MEDIC concept with its identifier, preferred term, and synonyms.

4.2 Artificial neural networks

Artificial neural networks (ANNs) are computing systems that approximate other functions. They were initially created to simulate nervous activity (McCulloch and Pitts, 1943), though nowadays, in the field of artificial intelligence, the biological nervous system is merely an inspiration to ANNs, instead of something they attempt to imitate. This section briefly introduces artificial neurons, the building units of ANNs, and two types of artificial neural networks. In addition, the activation and loss functions used in the neural networks in this study are also described.

4.2.1 Artificial neurons

As with the biological network of neurons, ANNs consist of inter-connected artificial neurons. Figure 4.3 shows an artificial neuron. The inputs of artificial neurons can be from other connected neurons, or from the input of the neural network. These inputs are multiplied with the weights of the neurons, which can be adjusted during the training stage. A neuron also has a bias term, which is a constant that is added to the sum of the weighted inputs. Before the result is output, an activation function is applied to the calculated number. The activation function can be linear or non-linear. Non-linear activation functions that are commonly used include the hyperbolic tangent function (tanh), the Rectified Linear Unit (ReLU), as well as the sigmoid function and the softmax function for the decision layers (Epelbaum, 2017). The output of one neuron can become the input of other neurons, or it can be the output of the network.

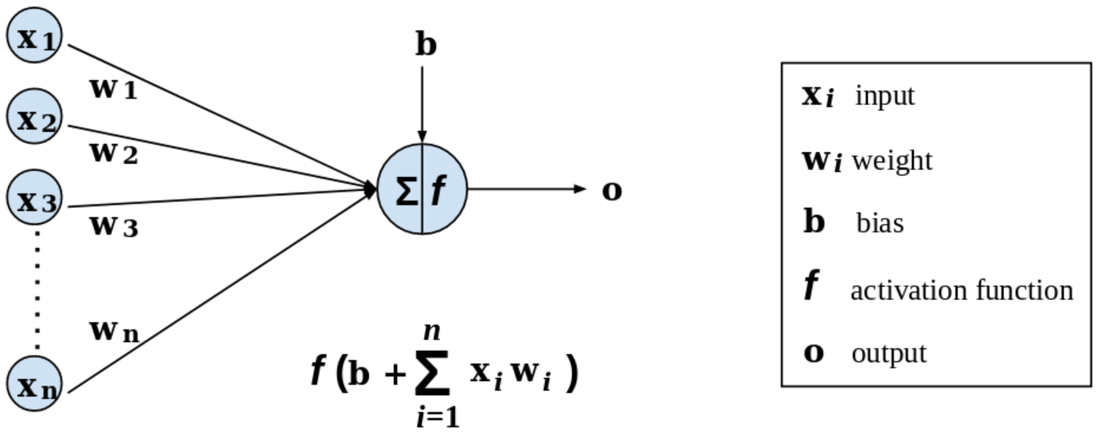


Figure 4.3: An example of an artificial neuron, with inputs (x_1-x_n from connected neurons, weights (w_1-w_n, bias b, the activation function f applied to the weighted sum of the inputs, and output o that can in turn be the input of other neurons. Adapted from `https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05`.

4.2.2 Feedforward neural networks

Artificial neurons connect to one another to form a neural network. Figure 4.4 shows a feedforward neural network, a type of network formed by artificial neurons. During training, a loss function is chosen to evaluate the error between the expected output and the actual output of the network. Examples of common loss functions include the mean squared error loss used for regression, and the cross-entropy loss for classification (Epelbaum, 2017). To train a neural network, its weights are updated according to the error calculated from the loss function using the backpropagation technique (Rumelhart et al., 1986). The loss function is chosen so that minimizing the loss helps the function represented by the neural network to approximate a real function the network is intended to learn during training.

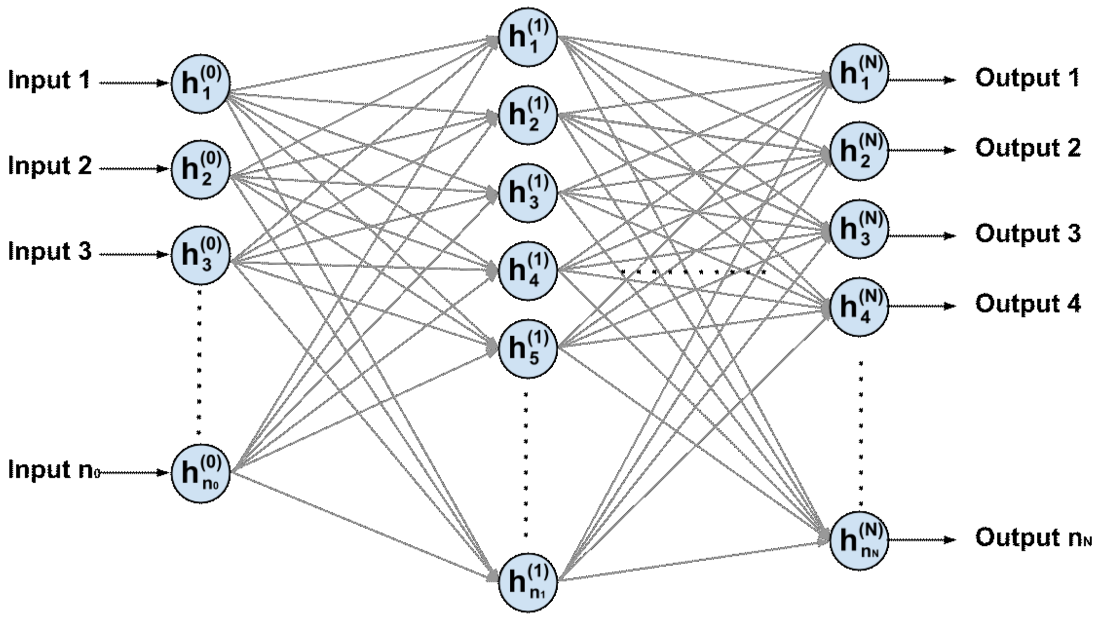


Figure 4.4: An example fully connected feedforward neural network consisting of $N + 1$ layers, and n_i neurons in each layer.

4.2.3 Convolutional neural networks

Convolutional neural network (CNN) is a type of ANN (LeCun, 1998). CNNs contain convolutional layers and, optionally, pooling layers. Convolutional layers consist of filters (or kernels) which contain parameters that are learned during training. Figure 4.5 shows an operation of a filter. When the filters operate, or convolve, across the input, the dot products of the input and the filter are taken. The outputs of a filter form a feature map, which becomes the input of the pooling layer (e.g. max-pooling). The number and size of filters, and the stride at which filters move across the inputs are all hyperparameters that can be adjusted for CNNs. CNNs are best known for computer vision algorithms, though they have also been applied to text processing (Severyn and Moschitti, 2015; Li et al., 2017).

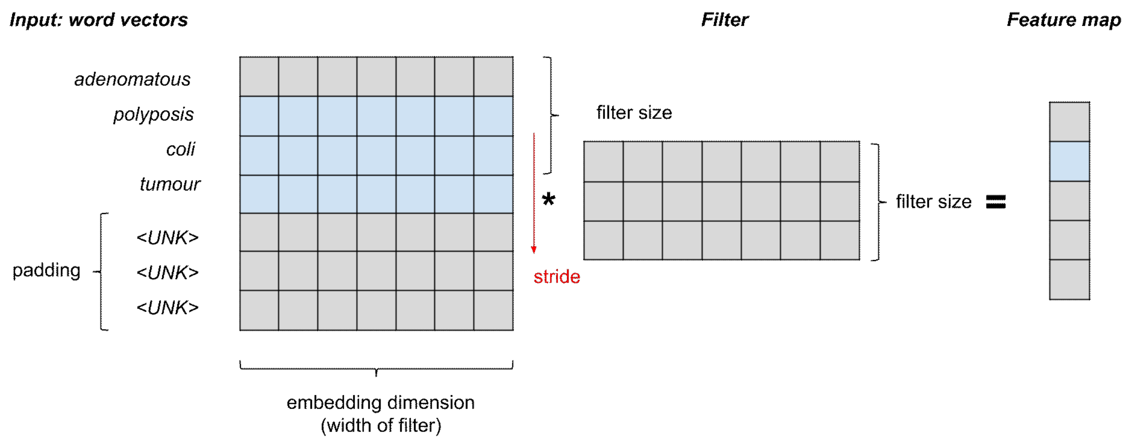


Figure 4.5: An operation of a CNN filter, where the dot product of the selected input vectors (in blue) and the filter is calculated .

4.2.4 Activation functions

Activation functions are applied to the sum of the weighted inputs of a neuron before output. There are three activation functions that are used in this study: ReLU, tanh, and the sigmoid activation function (Figure 4.6). The ReLU function is defined as

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}.$$

ReLU zeros out neurons whose sum of input weights is negative, and returns the positive sums of input weights as they are. It is a widely used activation function and has the advantage of fast computational speed (Ramachandran et al., 2018).

The tanh function is defined as

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

The hyperbolic tangent function has its output between -1 and 1. It is not as popular as it used to be due to the gain in popularity in ReLU (Epelbaum, 2017).

The sigmoid function here refers to the logistic function, which is defined as

$$f(x) = \frac{1}{1 + e^{-x}}.$$

Its output is between 0 and 1. When $x = 0$, the function outputs 0.5 (Figure 4.6). It is used, for example, in binary classification.

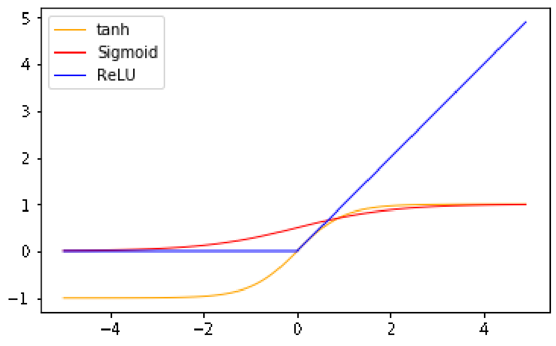


Figure 4.6: ReLU, tanh, and sigmoid activation functions.

4.2.5 Binary cross-entropy

Binary cross-entropy (BCE) is a loss used for binary classification. It is given by

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{y_i}) + (1 - y_i) \cdot \log(1 - \hat{y_i}),$$

where N is the total number of instances, x_i is one given input, $y_i \in 0, 1$ is the label, or gold standard, of the input, and $\hat{y_i}$ is the estimated value, or prediction, for the input. For a set of observed events (e.g. training instances), the BCE loss measures the difference between the true distribution of the data and the distribution of the data approximated by a function (a neural network, in this case) (de Boer et al., 2005; Epelbaum, 2017).

4.3 Vector representations of text

Text is represented as vectors to be input into the model. This derivation of vectors from text can be explained by the distributional hypothesis, which states that the meaning of linguistic items such as words can be derived from their contexts (Harris, 1954). Following this line of thought, words with similar distributions, i.e., occurring under similar contexts, are semantically alike. Various methods have been developed to induce vectors from distributional information of words from large amount of unstructured text (Mikolov et al., 2013a; Pennington et al., 2014; Basirat, 2017). The collection of induced word vectors is also called word embeddings. This section introduces two types of methods for the induction of word vectors from text: (1) TF-IDF word representations, which are used for the baselines, and (2) *word2vec* word embeddings, which are used both to establish baselines and to represent text in the proposed disease mention normalization system.

4.3.1 Term frequency-inverse document frequency word representations

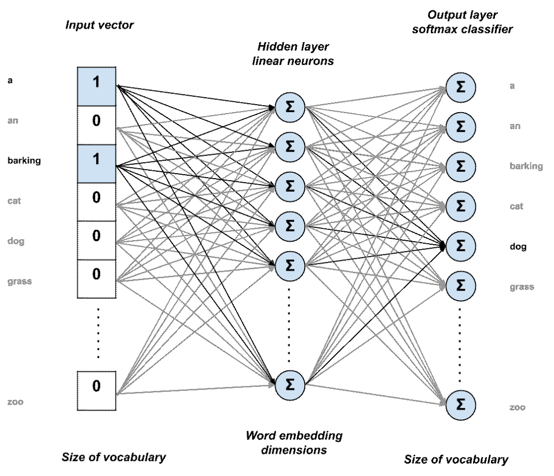
The numerical statistic TF-IDF, short for term frequency-inverse document frequency, is used to estimate the importance of a given word to a document from a collection of documents (Salton and Buckley, 1988). It is the product of the statistics term frequency and inverse document frequency. Term frequency counts the occurrence of a term in a document. There are several common weighting scheme variants for term frequency, with the simplest being a raw count of term occurrence. Inverse document frequency measures the amount of information a given word provides based on the assumption that, the fewer documents a word appears in, the more information it carries when it appears in a document. As with term frequency, inverse document frequency also has various formulations. An intuitive one is calculated as the logarithm of the inverse of the ratio of the number of documents a word appears in to the total number of documents:

$$idf\big(t, N\big) = \log \frac{N}{n_t}$$

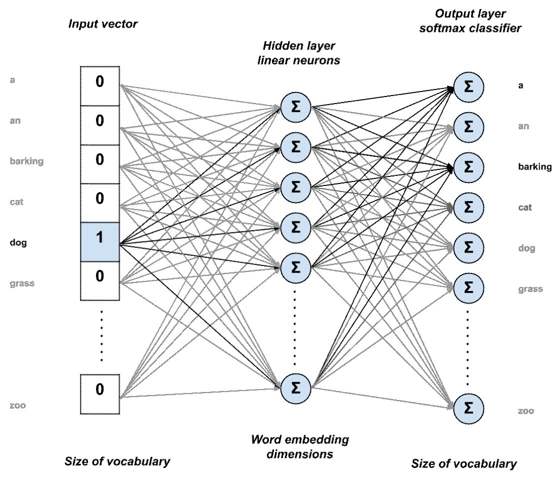
where t is the term, N is the total number of documents, and n_t is the number of documents that contain the term. The TF-IDF value increases as the number of times a term appears in a document increases, and decreases as the term appears in more documents. In information retrieval, TF-IDF is used for weighting how relevant a document is to a query; it lowers the weight commonly seen terms that may appear in a query carry while ranking the retrieved documents. For example, given a query “the Bible”, documents with the word “the” are less likely to be relevant than those with the word “Bible”, since “the” appears extremely frequently in English text.

4.3.2 *word2vec* word embeddings

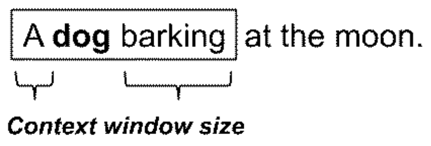
word2vec is a software developed by Mikolov et al. (2013a) to create word embeddings using neural networks. There are two model architectures used by *word2vec*, the Contin-


Input vector
Hidden layer linear neurons
Output layer softmax classifier
a
an
barking
cat
dog
grass
zoo
1
0
1
0
0
0
0
Σ
Size of vocabulary
Word embedding dimensions
Size of vocabulary

(a) CBOW


Input vector
Hidden layer linear neurons
Output layer softmax classifier
a
an
barking
cat
dog
grass
zoo
0
0
0
0
1
0
0
Σ
Size of vocabulary
Word embedding dimensions
Size of vocabulary

(b) Skip-gram



(c) Example sentence

Figure 4.7: Example sentence and its use when training on CBOW and skip-gram models. Adapted from `https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b`

uous Bag-of-Words (CBOW) architecture, and the skip-gram architecture, as illustrated in Figures 4.7a and 4.7b. Briefly, each model has an input layer of the vocabulary size, a hidden layer of linear neurons of a given size, and an output layer of the same size as the input layer. The size of vocabulary depends on the training text, and is affected by the minimum-count hyperparameter, which excludes words whose number of occurrences does not exceed the threshold. The size of the hidden layer is a hyperparameter to be determined for training. The models are trained to reconstruct the context of words. During training, the CBOW architecture (Figure 4.7a) takes in the context of a word and predicts the word, whereas the skip-gram architecture (Figure 4.7b) takes in a word and predicts its context. An example of a training instance is given in Figure 4.7c. For the phrase "A dog barking at the moon", the word "dog" is selected to be the focus word. The context of the focus word consists of a given number of words (in this case 1) that come before and after it. At the end of the training, the weights from the hidden layer are extracted as word embeddings.

Word embeddings, or continuous word representations, have also been demonstrated to capture linguistic regularities to some extent (Mikolov et al., 2013b). As a result of this property, word vectors can be added to and subtracted from one another. For example, in the experiments conducted by Mikolov et al. (2013b), operations from vectors "King - Man + Woman" yields a vector closest to "Queen" among all other word vectors.

4.4 Learning to rank

Learning to rank (LTR) (Li, 2011) are supervised approaches to building ranking models. In information retrieval, given a query and a set of documents, LTR aims to retrieve a subset of documents relevant to the query. Based on the number of documents taken into account in the loss function, LTR approaches can be categorized into pointwise, pairwise, or listwise approaches (Liu, 2010). The pointwise approach assumes LTR to be a regression problem. This approach associates every query-document with a score during training, and assigns a score for every query-document pair during prediction. The pairwise approach views LTR as a classification problem. This approach takes two documents at a time and uses a binary classifier to determine which one is more relevant to the query. Listwise approaches directly take into account a whole list of documents and attempt to optimally order this list of document. This class of approach is computationally more complex than pointwise and pairwise approaches.

LTR approaches have been applied in Natural Language Processing field. LTR in the context of normalization views mentions as queries and concept names as documents. For example, pairwise learning to rank (PLTR) has been used for disease mention normalization (Leaman et al., 2013; Li et al., 2017). PLTR used for normalization counts positive mention-concept name pairs as relevant and negative ones as irrelevant during training. During prediction, a binary classifier outputs a score for each mention-concept pair, and the concept with the highest predicted score for each mention is selected.

In this chapter, the corpus and the controlled vocabulary used in this study have been introduced. In addition, the theoretical background for the system described in this study has been covered. This includes the topics continuous vector space representation, pairwise learning to rank, and neural networks. In the next chapter, this background knowledge is integrated to build the disease mention normalization system.

5 Model architecture

In this chapter, the implementation of the disease mention normalization system is described. The model is developed based on the neural network architecture in Li et al. (2017) (hereafter termed `CNN-Li`), with the aim of removing the candidate generation step. The strings of the mentions from the corpus and the concepts from the controlled vocabulary are preprocessed and converted into vector representations. These vector space representations are then encoded by neural network architectures. The encoded string representations of concepts are ranked according to their similarity with that of a mention under examination, and the highest ranking concept is predicted as the normalized concept for the mention. The model is evaluated according to its accuracy. This chapter covers (1) the details of data preprocessing, (2) the model architectures and the reasoning behind their design, (3) the selection of training instances, and (4) how the models are evaluated.

5.1 Data preprocessing

For the NCBI disease corpus, the mention spans and their gold standards are read in from the `.txt` files for each set, which can be downloaded from the NCBI disease corpus website[1]. For the MEDIC dictionary, all concept names, including preferred terms and synonyms, are read in along with their corresponding identifiers from the `.tsv` file,

[1]`https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/`

which is included in the DNorm download[2]. After reading in the mention spans and concept terms, these strings are first lowercased and tokenized. The tokenizer used is `word_tokenize` from the `nltk` library, which splits the terms at punctuation other than periods (Bird et al., 2009). The maximum number of tokens the terms contain after tokenization for each set of the corpus and for the controlled vocabulary is tabulated in Table 5.1. For model development, the cutoff length chosen is 10, at which level over 98% of entities are completely represented when vectorized for all sets of data. For vectorization, the best performing embedding according to our extrinsic evaluation is used (detailed in Section 7.2). The embedding is described in Section 4.3.2. Tokens that are not in the vocabulary are represented with the vector for `<UNK>`, the unknown token. The vectorized strings are the inputs of the models.

Table 5.1: The maximum length of tokenized entities for the controlled vocabulary and the training, development, and test sets of the NCBI disease corpus. At the cutoff length of 10, over 98% of terms from all sets of data have all of their tokens represented by the vectorized representations.

Data	**Length of longest entity**	**Ratio of entities completely covered at length=10**
MEDIC	33	0.9878
NCBI (training set)	16	0.9986
NCBI (development set)	10	1.0000
NCBI (test set)	11	0.9990

[2]DNorm can be downloaded at `https://www.ncbi.nlm.nih.gov/research/bionlp/Tools/`. The file name of the MEDIC version used is `CTD_diseases.tsv`.

5.2 Model architecture

The model architecture of `CNN-Li` is altered in search for better performing architectures with the potential of not relying on candidate generation. The model is implemented using the Python deep learning library Keras (Chollet et al., 2015). This section explains the reasoning behind the design of the model, and introduces the model architectures that have been tested.

5.2.1 Model design

Figure 5.1 shows the model architecture of `CNN-Li`. It consists of two modules: the semantic representation module, and the similarity-based ranking module. The semantic representation module trains an encoder to encode the strings represented by word vectors. A schematic representation of the encoder used is shown in Figure 5.2. This encoder consists of an embedding layer, through which the strings are represented by word vectors. On top of this embedding layer, a convolutional layer is applied to extract feature vectors. The number of feature vectors extracted is the number of filters used. The max-pooling layer extracts the largest element from each feature vector. Since a given filter yields a higher number when the feature searched for by this filter is in a string, this max-pooling operation examines if a given feature is present in an examined string. The encoded representation of a string is the concatenation of the information of whether the string contains the set of features screen by the filters. `CNN-Li` has separate encoders for encoding mentions and concept names. The encoded strings are passed into the similarity-based ranking module. This module first concatenates the encoded strings and their similarity as measured by a trainable matrix. The concatenated information is then input into a fully connected hidden layer and a prediction layer for ranking. The top ranking concept is selected as the normalized concept for a given mention.

The architecture around which the experiments in this thesis is carried out is shown in Figure 5.3c. It differs from `CNN-Li` in that the mentions and the concepts share an encoder. This architecture is determined by comparing between shared and separate encoder(s), as well as ablation studies of the shared encoder architecture.

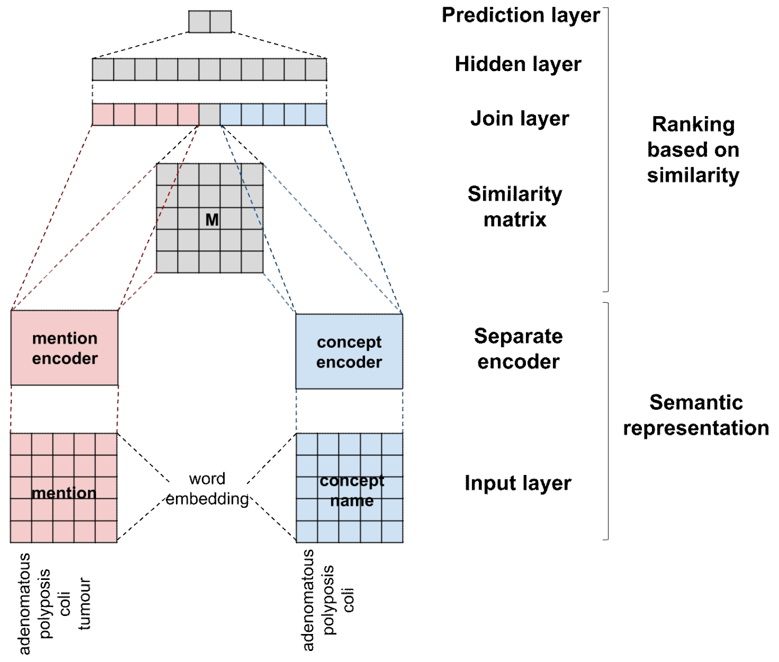


Figure 5.1: Architecture of `CNN-Li`

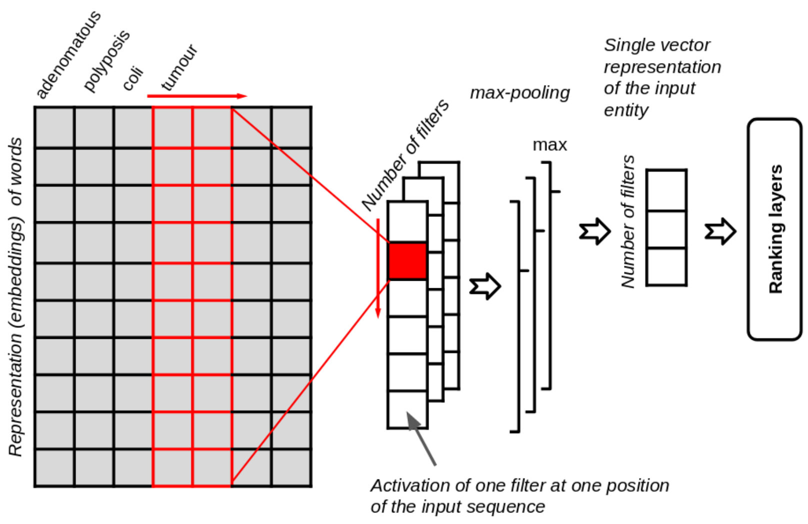

adenomatous
polyposis
coli
tumour
Representation (embeddings) of words
Number of filters
max-pooling
max
Single vector representation of the input entity
Number of filters
Ranking layers
Activation of one filter at one position of the input sequence

Figure 5.2: Architecture of the CNN-based encoder.

5.2.2 Tested architectures

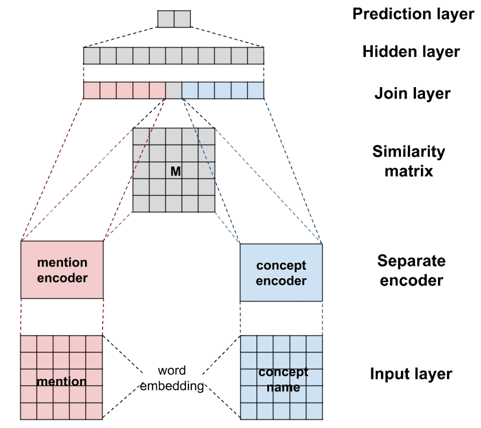
For the comparison of models with a shared encoder with that with separate encoders, the four architectures shown in Figure 5.3 are tested. These architectures are: full model with separate encoders (`CNN-Li`), full model with a shared encoder, and ablation models of these two full models where the ranking is directly based on the output of the similarity matrix.

To evaluate the efficacy of the similarity matrix, ablation models where the matrix is substituted with cosine similarity or completely removed (as shown in Figure 5.4) are tested. Cosine similarity is used because the similarity matrix is a generalization of the cosine similarity operation, i.e., cosine similarity matrix is a special case of the similarity matrix, where the matrix is an identity matrix.

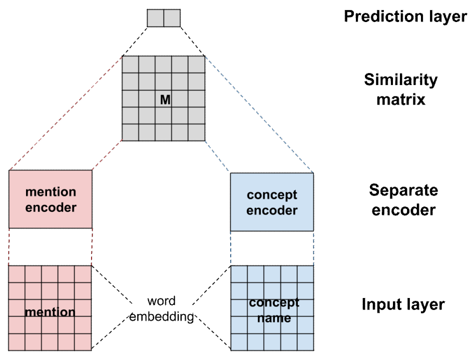
5.3 Selection of training pairs

In total, there are 5,145 mentions in the training set of the NCBI disease corpus and 67,782 concept names in the MEDIC version used. This makes 348,738,390 possible mention-concept name training pairs. It is not feasible to train the model on all pairs of mention and concept name. In addition, preliminary experiments suggest that the model does not learn well, probably because nearly all of the mention-concept name pairs it is trained on are negatives. Thus, training instances are sampled for the training of the model. To estimate the loss on the development set during training, mention-candidate pairs are sampled the same way as the training data are sampled.

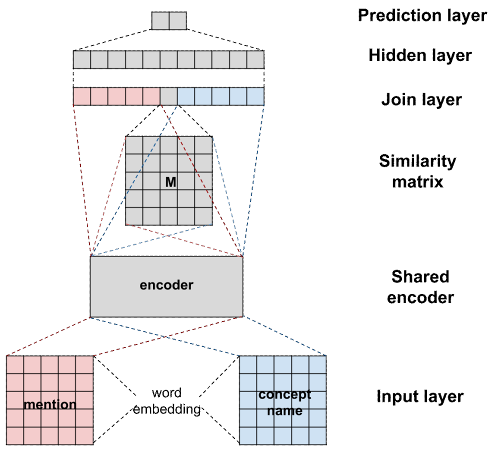
In the NCBI disease corpus, some of the mentions are mapped to more than one concept identifier. There are 115 and 30 such cases in the training and development sets respectively. In addition, there are 142 and 35 occurrences of mentions in the training and



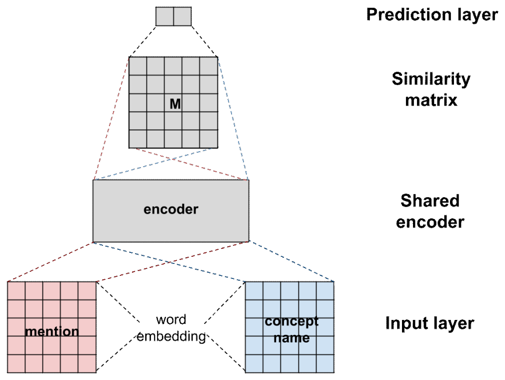
(a) Separate encoder with join layer and fully connect hidden layer.



(b) Separate encoder without join layer and fully connected hidden layer.

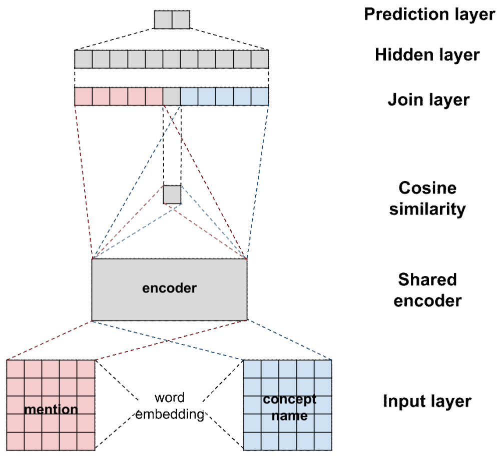


(c) Shared encoder with join layer and fully connected hidden layer.

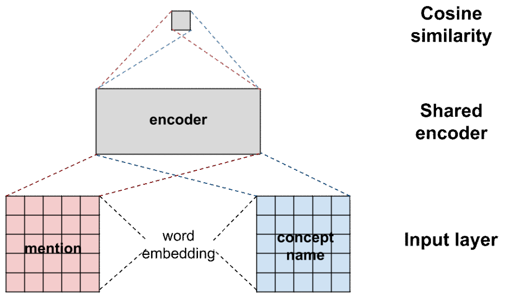


(d) Shared encoder without join layer and fully connected hidden layer.

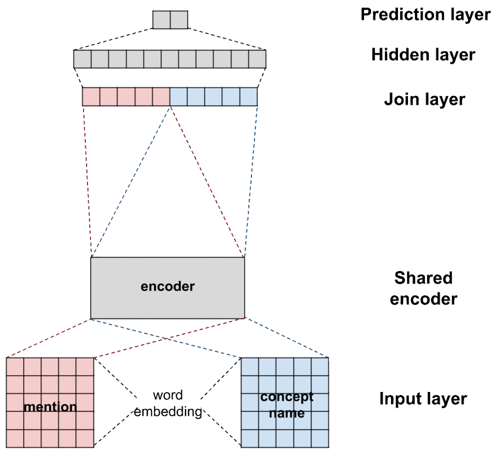
Figure 5.3: Architectures used for testing pre-training.



(a)



(b)



(c)

Figure 5.4: Architectures used for testing the performance of the similarity matrix. The similarity matrix is replaced with cosine similarity or removed.

development sets respectively with concept identifiers that are not included in the MEDIC version used. This is because, during the construction of the NCBI corpus, the disease mentions are annotated if there is a suitable concept in the UMLS Metathesaurus. However, not all of the concepts used are present in the MEDIC version used by DNorm. The version provided by DNorm is used for easier comparison with existing systems, though it is possible to use newer versions of MEDIC, which contain more concepts. The mentions whose concept identifiers are not in the MEDIC version used are excluded for the training of the model.

5.3.1 Selection of positive training pairs

For each mention in the training corpus, one positive mention-concept name pair is used for training. Since some concepts have synonyms, and therefore multiple concept names, the concept name for training has to be selected. The selection is based on string similarity as measured by the ratio of the number of matching characters to the number of characters in two examined strings. Matching characters are the longest matching substrings of the two strings, and, recursively, those in the flanking unmatched regions[3] (Ratcliff and Metzener, 1988). The concept name with the highest similarity to the mention is chosen as the positive training instance.

5.3.2 Selection of negative training pairs

For each pair of positive mention and concept name, a given number of pairs of negatives are randomly sampled. The negatives are re-sampled every epoch. The random sampling of negatives ensures that the model is presented with a non-biased sample of negatives during training, as opposed to sampling only hard negatives (i.e. concept names that pass heuristic filters) for the training of the model. Preliminary experiments suggest that,

[3]Implemented using the `ratio` method of the `SequenceMatcher` class from the `difflib` library. Documentation available at `https://kite.com/python/docs/difflib.SequenceMatcher`

when the model is trained on only hard negatives and is to predict on all concept names, the performance of the model may drop because the training data does not represent the test data.

5.4 Evaluation

The development set of the NCBI corpus has 787 mentions, of which 363 are unique. Since evaluation of model performance on the whole development set is time- and memory-consuming, 100 unique mentions are randomly sampled to form the sampled NCBI development set. This sampled set is used for evaluation during model development.

The metric used for evaluation is accuracy. It calculates the percentage of correct prediction out of all predictions. A prediction is considered correct if the concept ranked the highest for a mention is the same as its gold standard. Mentions whose gold standard are not included in the MEDIC version used are always falsely predicted. In addition, the system described in this thesis is not able to predict multiple concepts for a single mention, because the system always predicts the highest ranking concept as the normalized concept. Thus, the prediction for mentions with composite mappings or multiple concept mappings is always counted as false, even if one of the concepts in the gold standard is predicted[4].

[4]This evaluation standard follows those of related studies using the NCBI disease corpus

6 Experimental evaluation

The workflow of the experiments described in this thesis is shown in Figure 6.1. Briefly, the baselines are established and the embedding selected through evaluation. For initial evaluation of the model, the neural architecture of `CNN-Li` with the same hyperparameters (except for using tanh instead of ReLU as the CNN activation function and no dropout) is used[1]. However, the results (not reported) showed that the model with the initial setup does not outperform even the exact match baseline. Pre-training tasks are thus designed to try to improve the ability of the network to at least predict exact matches. During the pre-training experiments, different model architectures, i.e., models with a shared encoder or separate encoders, are tested. The results from the pre-training experiments show that synonym pair pre-training with the shared encoder architecture works best. This experimental setting is then used in the subsequent set of experiments, where different similarity measures for the ranking module are tested. The results indicate that cosine similarity of encoded strings works best as the similarity measure. The architecture with a shared encoder and the cosine similarity measurement is used to conduct a hyperparameter search. The best performing models from this search are evaluated on the full NCBI development set. The top-performing model from this evaluation is evaluated on the NCBI test set.

This chapter describes the the experimental setup for the experiments reported in this

[1]The difference in the hyperparameters used is a result of adjusting hyperparameters in preliminary experiments not included in this thesis.

thesis. These experiments (1) establish baselines, (2) select the word embedding, (3) compare models trained without pre-training and with either exact match or synonym pair pre-training, and (4) compare different similarity measures. The next chapter reports the results of experiments and discusses about them. In the discussion, the reasoning behind these experimental design is provided, as the design of one experiment often depends on the results from a previous one.

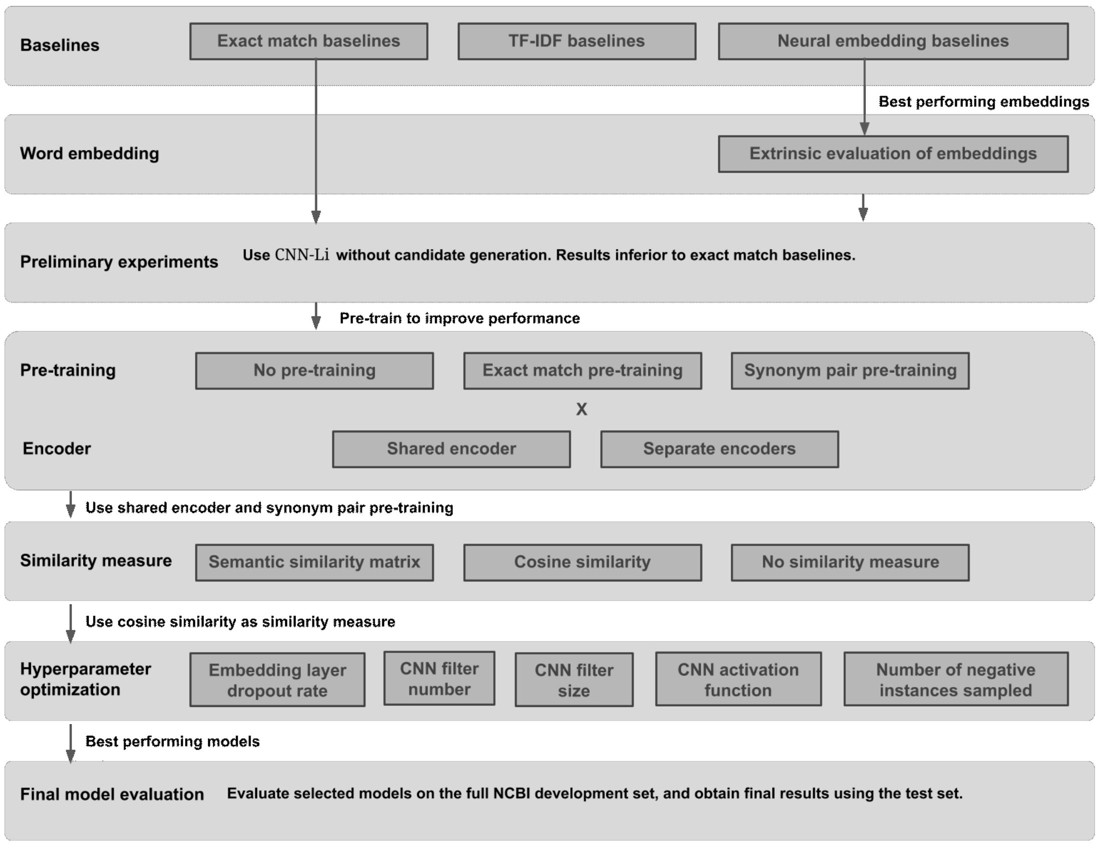

Baselines
Exact match baselines
TF-IDF baselines
Neural embedding baselines
Best performing embeddings
Word embedding
Extrinsic evaluation of embeddings
Preliminary experiments
Use CNN-Li without candidate generation. Results inferior to exact match baselines.
Pre-train to improve performance
Pre-training
No pre-training
Exact match pre-training
Synonym pair pre-training
X
Encoder
Shared encoder
Separate encoders
Use shared encoder and synonym pair pre-training
Similarity measure
Semantic similarity matrix
Cosine similarity
No similarity measure
Use cosine similarity as similarity measure
Hyperparameter optimization
Embedding layer dropout rate
CNN filter number
CNN filter size
CNN activation function
Number of negative instances sampled
Best performing models
Final model evaluation
Evaluate selected models on the full NCBI development set, and obtain final results using the test set.

Figure 6.1: Experimental workflow.

6.1 Baselines

The experimental setups of three types of baselines, the exact match baselines, the neural word embedding baselines, and the TF-IDF embedding baselines are described in this section.

6.1.1 Exact match baselines

For the exact match baseline, a concept is predicted for a given mention if the surface form of any of its concept names is exactly the same as that of the mention. If multiple concepts have the matching form for a mention, all of these concepts are predicted for the given mention.

For the lowercase exact match baseline, a concept is predicted for a mention if the lowercase version of the two forms are identical.

6.1.2 Word embedding baselines

To take into account the semantics of the mentions, cosine similarity of word vectors are used to predict the concept for a given mention. Similar to the method used by Kaewphan et al. (2014), word embeddings are used to represent the mentions and the concepts. For each name of a mention or a concept, the name is first changed to lower case and tokenized. Taking advantage of the linguistic regularities of word embeddings shown by Mikolov et al. (2013b), the vector representation of a name is then taken as the mean of all the vector representations of its tokens. The cosine similarity is computed for every pair of mention and concept, and the concept with the highest cosine similarity is then chosen as the prediction for a given mention. For the word embedding baseline, the publicly available word embeddings described below are tested:

(1) The *word2vec* embeddings trained on PubMed abstracts and PubMed Central open access (PMCOA) full-text articles[2] (Pyysalo et al., 2013) (`emb-Pyysalo`),

(2) The embeddings from Chiu et al. (2016)[3], also trained on Pubmed abstracts and PM-

[2] `http://bio.nlplab.org/`

[3] `https://github.com/cambridgeltl/BioNLP-2016`

COA full text articles using the *word2vec* algorithm. There are two embedding models available, one trained with context window size 2 (`emb-Chiu-2`), and the other context window size 30 (`emb-Chiu-30`),

(3) The embeddings provided by Li et al. (2017) (`emb-Li`) used in the previous state-of-the-art system[4], and

(4) The embeddings trained on PubMed abstracts with more than 28 million biomedical publications. The *word2vec* algorithm was used, with a context window size set to 5 (`emb-Brokos`) (Georgios-Ioannis, 2018).

Since word embeddings trained on non-biomedical text do not necessarily underperform those trained on biomedical text in various bioNLP tasks (Wang et al., 2018), two word embeddings trained on general-domain text were also included:

(5) The embedding trained on part of the Google News data set with about 100 billion words (`emb-GoogleNews`)[5]. The embedding was trained on part of the Google News data set with about 100 billion words using the *word2vec* architecture and the negative sampling algorithm, and

(6) The embedding trained on Wikipedia 2017, the UMBC webbase corpus, and the news data sets from `statmt.org`, with 16 billion tokens in total (`emb-Mikolov`) (Mikolov et al., 2018).

[4]`https://github.com/wglassly/cnnormaliztion`, no detail information could be found for this embedding

[5]`https://code.google.com/archive/p/word2vec/`

6.1.3 TF-IDF baselines

To measure how similar a document is to a query, vector representations are created based on the TF-IDF values of the words contained in the query and the document. The similarity is calculated as the cosine similarity of these vectors.

TF-IDF sparse vector space representation of disease names has previously been used for normalization (Leaman et al., 2013; Mehryary et al., 2017). As the TF-IDF baseline, a similar approach to that taken by Mehryary et al. (2017) is adopted. That is, the mentions are seen as queries and concepts as documents. The "words" are ngrams, or more specifically, unigrams, bigrams, and trigrams, of the mentions and concepts. For the experimental setup, TF-IDF sparse matrix representations are built using both stemmed and original mentions and concepts after they are lowercased. Stemming is tested because it may help to improve performance depending on the mention type (Mehryary et al., 2017). The stemming is carried out using the nltk[6] implementation of Porter stemmer (Porter, 1980). The TF-IDF vector representations of mentions and candidates are built using the scikit-learn library (Pedregosa et al., 2011). For each mention, the concept whose vector representation has the highest cosine similarity with that of the mention is selected as the predicted term.

6.2 Selection of embeddings

To determine the word embedding to be used for experiments, an extrinsic evaluation is carried out on the top performing embeddings in Section 7.1.2. The extrinsic evaluation uses the architecture shown in Figure 5.3c to test how well models perform on the NCBI development set when trained on the synonym pairs of the controlled vocabulary using the evaluated embeddings. This architecture is chosen because preliminary experiments

[6]https://www.nltk.org/

indicate that it is one of the architectures worth further experimentation. For the representation of disease mentions and concept names, the model encodes vectorized strings by a shared CNN encoder (as shown in Figure 5.2). The ranking of concept names takes into account the similarity as measured by a matrix, and the encoded strings. The hyperparameters used are: padding length of 10 for vectorized disease mentions and concept names, dropout rate of 0.5 for the embedding layer, 50 filters and kernel size of 3 with the stride of 1 for the CNN encoder, binary cross-entropy as loss, and adam optimizer with learning rate set to 5×10^{-5}. During training, for every positive mention-concept name pair, 29 negatives are sampled. The vocabulary size of the embedding is limited to 1,000,000.

6.3 Pre-training

Pre-training tasks are designed in an attempt to improve the performance of the model in predicting at least exact matches, or cases where the mention and the concept name share an identical string. Two pre-training tasks, the exact match prediction task, and the synonym prediction task, are tested. The implementation details of these two pre-training tasks are described in this section.

6.3.1 Exact match prediction

The exact match pre-training trains the model to identify identical strings. For each string of mention from the training set of the NCBI corpus, or concept name from the MEDIC dictionary, the model is given two candidates to rank. One is an identical string, the other is another random string. The model is expected to rank the identical string over the random one.

The initial pre-training setup assumes that the vector representation of a randomly selected candidate is different from that of a given term. However, using the vocabulary size of one million for the word embedding, approximately 0.2% of the randomly selected candidates have the same vector representation as the mention. This is because some acronyms or rare terms have the `UNK` token as their representation, which results in the same vector representation from the word-level embedding used. With the initial experimental setup, there are 18, 10 and 3270 terms represented by `UNK` from the training corpus, the development corpus, and the controlled vocabulary respectively. By coincidence, one of these terms may be paired up with another. Consequently, for the creation of the pre-training data, negative candidates are selected in a way so that they do not coincidentally have identical vector representations to the mentions.

To evaluate how well the model predicts on exact matches, a set of synthetic validation data is created. This set of data contains 7293 synthetic mentions, taken either from the mentions of NCBI training set or the MEDIC concept names. For each of these mentions, two candidates are given. One is identical to the mention, and the other is a random mention or concept name. The model is expected to predict the exact match.

For the exact match pre-training, the four architectures shown in Figure 5.3 are tested: architectures with a shared encoder or separate encoders, each with or without the join and fully connected hidden layer. The separate encoder architecture is tested because it is the architecture used by Li et al. (2017). The shared encoder architecture is tested because the task of exact match prediction amounted to predicting candidates with vectors whose cosine similarity with that of the mention is one. Consequently, it is intuitively easier for networks with a shared encoder to learn the exact match prediction task, since the encoded representations of a string on the mention and the candidate sides are always the same. This use of shared encoder also halves the number of trainable parameters at the encoding stage. In addition, ablation architectures for the separate encoder architecture and the shared encoder architecture are also tested. These ablation architectures are tested to measure how well the models perform with only similarity information, without any further information about the encoded strings. The ablation architectures have the same semantic representation module as their full architecture counterparts. However, for the similarity-based ranking module, these architectures have the prediction layer directly on top of the similarity layer. That is, they do not contain a join layer nor a dense layer on top of the similarity layer.

The word embedding used is `emb-Brokos-200` with vocabulary limit set to 1,000,000. Other hyperparameters are the same as in other experiments. The number of epochs for pre-training is set to 100, with an early-stopping tolerance[7] of 15 epochs. During pre-training, the model is evaluated on both synthetic validation data and the sampled NCBI development set. When the pre-training finishes, the training continues on the NCBI training set. The hyperparameters are kept the same[8]. As with pre-training, the number of epochs for pre-training is set to 100, with an early-stopping tolerance of 15 epochs. The model is evaluated on the sampled NCBI development set.

6.3.2 Synonym prediction

The synonym pair prediction task trains a model to predict the synonyms or exact match of a concept. The synthetic training data is generated from the MEDIC dictionary. The training instances that would be generated for a hypothetical concept with the preferred term `NEOPLASM` and the synonyms `TUMOR` and `CANCER` is illustrated in Figure 6.2. During training, the model would be given all combinations of the name pairs of this concept (e.g. `NEOPLASM` and `TUMOR`), as well as the duplicate of each of these names

[7]Early-stopping tolerance keeps track of how many epochs has passed since there is an improvement in the model performance. If the threshold of early-stopping tolerance is exceeded, the training is terminated.

[8]The learning rate is kept the same because (1) the performance of the model does not drop after switching to the real training data, and (2) it is easier to compare the results obtained to those from models trained without pre-training.

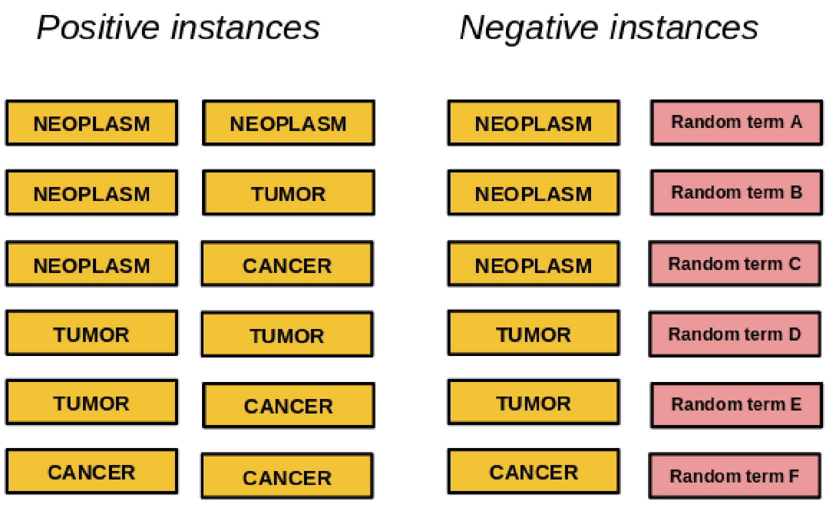


Figure 6.2: Synonym pair training instances generated for a hypothetical concept.

(e.g. `CANCER` and `CANCER`), as the positive instances. In this case, 6 positive instances would be generated for this concept. The negative instances are randomly chosen, as described in Section 6.3.1. In addition, for comparison to the results of exact-match pre-training, the experimental settings are the same as those for the exact match pre-training as described in the same section.

6.4 Similarity measures

To evaluate the effect of the similarity measure on the performance of the system, different measures of similarity are experimented with. These architectures, as illustrated in Figure 5.4, are (1) shared encoder architecture with join and fully connected hidden layers, with the similarity matrix replaced by cosine similarity, (2) shared encoder architecture with only cosine similarity, and (3) shared encoder architecture with join and fully connected hidden layers, with the similarity matrix completely removed. These architectures are pre-trained with the synonym pair prediction task before formal training on the NCBI training set. For comparison purposes, a set of experiments without any pre-training is also performed. All the experimental hyperparameters and settings are otherwise identical to those in Section 6.3.1.

6.5 Hyperparameter optimization

A hyperparameter search is done on the best performing architecture from Section 6.4. The hyperparameters that are tuned are mostly the ones from the encoder. These include embedding layer dropout rates of 0, 0.1, 0.25, 0.5, and 0.75, filter numbers of 25, 50, 100, 150, kernel sizes of 1,2,3, and ReLU or tanh as the CNN activation function. In addition, the number of negative instances sampled for each positive instance is adjusted: Negative sample numbers 1, 9, 29, 99, and 999 are tested. During the hyperparameter tuning, not all combinations of the above hyperparameters are tested. Instead, choices of hyperparameters that lead to better results are fixed while other hyperparameters are continually adjusted. Where there is no result to suggest a better value for a given hyperparameter, or the results are not clear enough to do so, hyperparameter values that are computationally less expensive are used. Finally, selected top performing models in the hyperparameter search are evaluated on the full NCBI development set.

6.6 Evaluation on held-out data

To evaluate the performance of the model on held-out data, the top performing model on the full NCBI development set is evaluated on the NCBI test set.

7 Results and discussion

In the previous chapter, the experimental setup has been described. These experiments include (1) baseline experiments, (2) extrinsic evaluation of word embeddings, (3) pre-training experiments, and (4) similarity measure evaluations. In this chapter, the results from these experiments are reported and discussed. In the discussion, the reasoning for the design of the subsequent experiments are presented taking into account the results from the experiment under analysis.

7.1 Baselines

Three types of baselines are used: the exact match baselines, the neural word embedding baselines, and the TF-IDF embedding baselines, as described in Section 6.1. This section describes the reasoning behind their usage, and establishes these baselines for this study.

7.1.1 Exact match baselines

Using exact match without case normalization results in an accuracy of 0.165 on the development set, and 85% of the correctly mapped mentions are acronyms. While acronyms can be successfully mapped to their corresponding concepts because they are usually in upper case, mentions such as "colorectal cancer" are not mapped to `Colorectal Cancer` simply due to the case difference. The effect of this difference is especially noticeable for the NCBI disease data set because, while the mentions in the corpus tend to be in lowercase except for proper names, the MEDIC concepts tend to have the first letter of every token of a concept name in uppercase if the name originally comes from MeSH, and the whole name in uppercase if it comes from OMIM.

Table 7.1: Accuracy scores for different methods on the NCBI development, and sampled development sets.

Method	**Development set**	**Development set (Sampled)**
Exact match	0.165	0.10
Lowercase matching	0.529	0.39

This difference in case is taken into account by case normalization before comparison in the lowercase matching baseline. This method yields an accuracy of 0.529 on the development set, an improvement of 0.364 from the exact match baseline. The only false prediction made on the test set with this method, which the exact match method does not make, is the prediction of the mention "Aniridia", whose gold standard is `MESH:D015783`. While this symptom is mapped to the correct concept `Aniridia` (`AMESH:D015783`) by the exact match, the lowercase matching method also yields `Aniridia, type 2` (`MESH:C536372`), a child concept of the correct concept, because one of its alternative names is `ANIRIDIA`.

These two string matching methods, however, cannot predict the correct concept when the matching string is not listed as a name of the concept in the controlled vocabulary. For example, acronyms of diseases used by authors, such as "FAP" for "Familial Adenomatous Polyposis" and "DM" for "Dystrophia Myotonica", cannot be correctly mapped because these acronyms are not listed as alternative names of the respective concepts in the controlled vocabulary. Correct mapping of these concepts would require information from the context where these mentions are in, or the use of an acronym expansion method, such as incorporating results from Google search. Apart from acronyms not listed in the controlled vocabulary, disease mentions without exact matching concept names cannot be predicted correctly, either. For example, the correct concept for the mentions “inherited disorder” and “allelic disorders” is `Genetic Diseases, Inborn`. These two mentions, however, are not correctly predicted despite being semantically similar to the concept.

7.1.2 Word embedding baselines

The accuracy scores obtained using the embeddings introduced in Section 6.1.2 are shown in Table 7.2. The highest accuracy on the development set 0.714 is obtained using the embeddings from Georgios-Ioannis (2018). More dimensions does not necessarily result in better accuracy scores on this task. The embeddings trained on biomedical texts outperform those trained on general domain texts, possibly due to their ability to better capture the semantics of biomedical terms (Wang et al., 2018).

Table 7.2: Accuracy scores obtained by taking the concept with the highest cosine similarity with a given mention for the development set of the NCBI disease corpus.

Embedding	**Dimension**	**Vocabulary size**	**Development set accuracy**
`emb-Pyysalo`	200	1,000,000	0.557
`emb-Chiu-2`	200	1,000,000	0.596
`emb-Chiu-30`	200	1,000,000	0.600
`emb-Li`	50	309,058	0.652
`emb-Brokos-200`	200	1,000,000	**0.714**
`emb-Brokos-400`	400	1,000,000	0.710
`emb-GoogleNews`	300	1,000,000	0.374
`emb-Mikolov`	300	1,000,000	0.525

7.1.3 TF-IDF baselines

The accuracy scores obtained using the TF-IDF based methods are shown in Table 7.3. Using ngrams (n=1-3) of un-stemmed mentions and concepts resulted in better performance than stemmed ones. The higher accuracy score obtained was 0.711.

Table 7.3: Accuracy scores obtained using cosine similarity of TF-IDF weighted sparse vector representations. All results averaged over five random seeds.

Stemming	**Development set accuracy**
Yes	0.668
No	**0.711**

The best performing exact match and vector space representation methods on the NCBI development set are evaluated on the NCBI test set. The results are shown in Table 7.4. The baselines used in this study are established as the following: For exact match, the accuracy score of 0.460 obtained using the lowercase exact match method on the test set is used. For the word embedding baseline, the accuracy score of 0.628 on the test set is used.

Table 7.4: Accuracy scores on the NCBI test set obtained using selected baseline methods.

Baseline	**Test set accuracy**
Exact match	0.460
Word embedding	0.628

7.2 Selection of embeddings

The accuracy score obtained by models trained using the top three performing word embeddings from Section 7.1.2 is plotted in Figure 7.1. The three embeddings used are `emb-Li`, `emb-Brokos-200`, and `emb-Brokos-400`. The highest accuracy attained within 45 epochs for these embeddings respectively are 0.294, 0.423, and 0.417. While the model trained with 50-dimensional embedding does not outperform the ones trained with 200-dimensional or 400-dimensional embeddings, models trained with 200-dimensional embedding and that with 400-dimensional embedding do not differ much in performance. These results are in line with those from Section 7.1.2. Besides, the observation that results obtained using `emb-Brokos-400` do not differ noticeably from those obtained using `emb-Brokos-200` agrees with the result from Georgios-Ioannis (2018), the creator of these embeddings. Thus, `emb-Brokos-200` is selected as the word embedding used for further experiments.

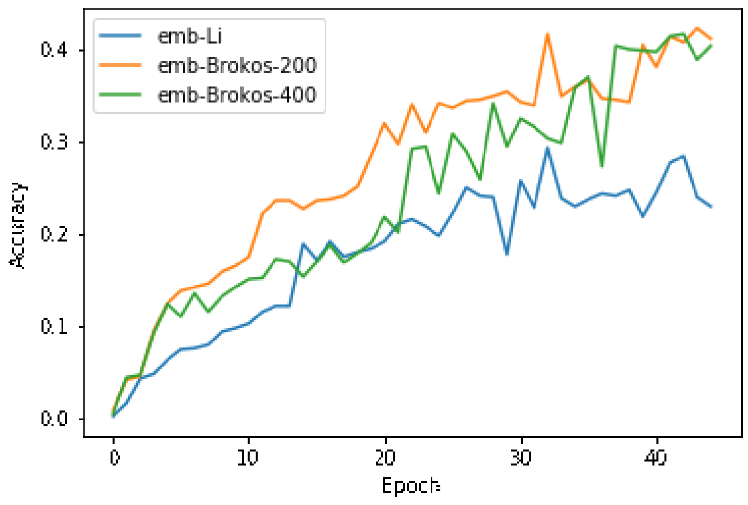


Figure 7.1: Extrinsic evaluation of word embeddings: Accuracy over epochs for models trained using the embeddings `emb-Brokos-200`, `emb-Brokos-400`, and `emb-Li`.

7.3 Pre-training

Preliminary experiments indicated that pre-training on synthetic data may be beneficial to the performance of the models. In the preliminary experiments without any pre-training, accuracy was around 0.03, which was much lower than the lowercase matching baseline

0.460[1]. This indicated that the neural network was not even able to identify exact matches between pairs of mention and candidate. Thus, the first step to improving the performance of the network is to ensure that it is able to predict exact matches. To do this, pre-training of the network is carried out. As introduced in Section 6.3, two different settings of pre-training are tested: (1) pre-training with the task of predicting identical strings, and (2) pre-training with the task to predict both identical strings and synonymous names.

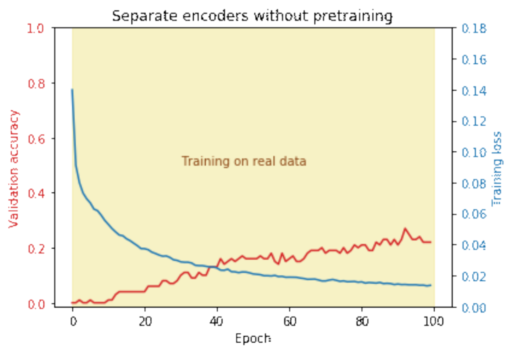
7.3.1 No pre-training

For comparison purposes, a set of experiments with otherwise identical settings as the ones conducted for the exact match pre-training and synonym pair pre-training (described in Section 6.3.1) is carried out without any pre-training. The highest accuracy scores obtained for each architecture are tabulated in Table 7.5. The training loss and validation accuracy on the sampled NCBI development set over epochs is shown in Figure 7.2. The ablation architectures fail to work (highest accuracy obtained 0.04 and 0.00 for separate and shared encoder architectures) without any pre-training, while the full models have performance similar to each other at the end of the training after 100 epochs. The full models could potentially reach even higher accuracy, if the training had not ended due to finishing 100 epochs.

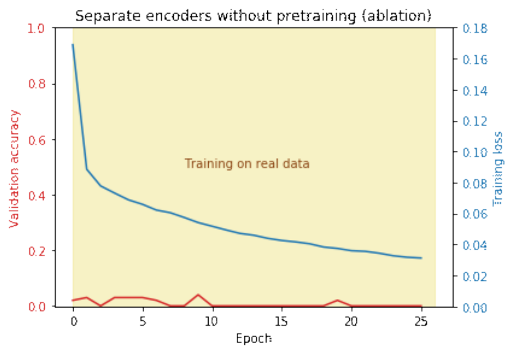
Table 7.5: Highest accuracy obtained on the sampled NCBI development set for different architectures when directly trained on the NCBI training set.

	Full model	**Ablation model**
Separate Encoders	0.27	0.04
Shared Encoder	0.30	0.00

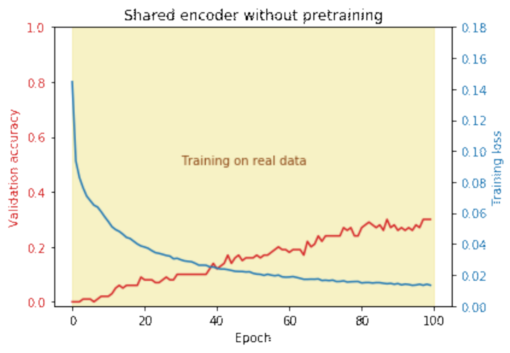
[1]Since the tokens are lowercased in the preprocessing step, the lowercase matching baseline is in effect an exact match baseline for models with preprocessed input



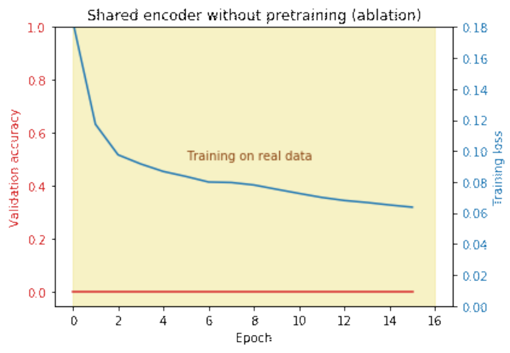
(a)



(b)



(c)



(d)

Figure 7.2: Training loss and validation accuracy on the sampled NCBI development set over epochs for different architectures (separate encoders or a shared encoder, with or without join and hidden layers) directly trained on the NCBI training set.

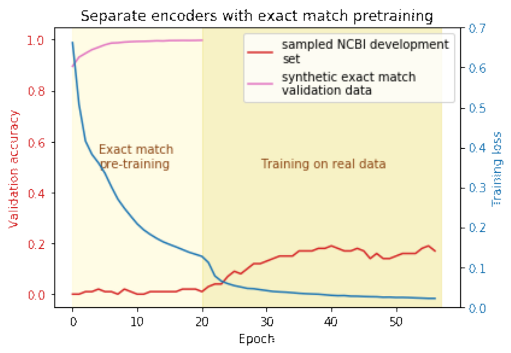
7.3.2 Exact match prediction

Table 7.6 shows the highest accuracy scores reached on the sampled NCBI development set for each of the architecture tested in the pre-training and training stages. The shared encoder architectures achieve higher accuracy than the separate encoder architectures. Both of the full models outperform their ablation counterparts, indicating that the encoded representations of strings carry information useful for similarity ranking not captured by the similarity layer. Interestingly, the exact match pre-training task only seems to benefit the full model with a shared encoder. For the other models, the pre-training may either have been terminated before any effect was seen, or have not worked. This may be due to the pre-training task being too easy, or the pre-training data being too small.

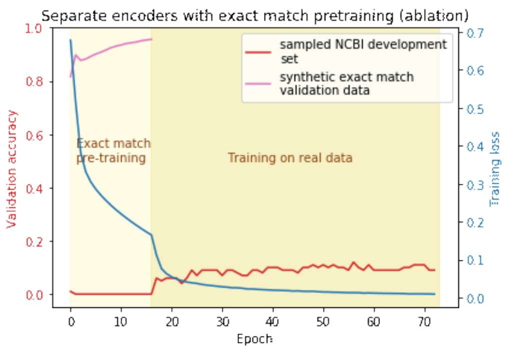
Table 7.6: Highest accuracy obtained during pre-training / after the training completes on the sampled NCBI development set for different architectures when pre-trained on the exact match task.

	Full model	**Ablation model**
Separate Encoders	0.02 / 0.19	0.01 / 0.12
Shared Encoder	0.13 / 0.35	0.00 / 0.20

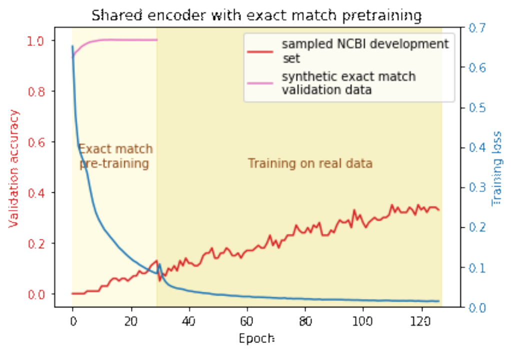
The accuracy and training loss over epochs for all the architectures are shown in Figure 7.3. Figures 7.3a and 7.3b show the accuracy and training loss on sampled NCBI development set over epochs for the full and ablation models with separate encoders. For both of these models, the increase in accuracy score mainly happens after the pre-training, which indicates that pre-training for separate encoder architectures probably does not help with model performance. On the other hand, it could also be the case that the hyperparameters used are not suitable for these architectures, or that more iterations of training are required. Figures 7.3c and 7.3d show the accuracy and training loss on the sampled NCBI development set over epochs for the full and ablation models with a shared encoder. While the accuracy reached by the ablation model with a shared encoder is on par with that of the full model with separate encoders, the latter attained the performance faster. The highest accuracy of 0.20 was reached at epoch 70 (54 epochs after the training on the NCBI training set started) for the ablation model with a shared encoder, whereas the highest accuracy of 0.19 was reached at epoch 40 (21 epochs after switching to the NCBI training set) for the full model with separate encoders. The best performing architecture on this task, however, is the full model with a shared encoder. During the pre-training stage, an accuracy score of 0.13 was reached on the sampled NCBI development set. The highest accuracy of 0.35 was reached at epoch 110 (80 epochs after switching to the NCBI training set).



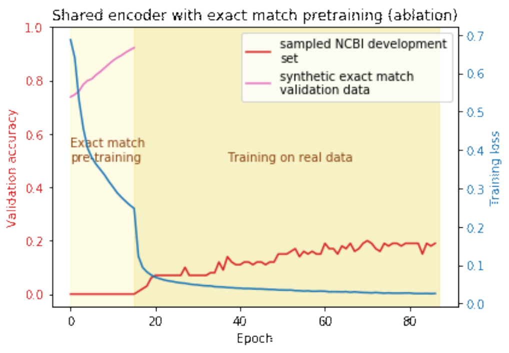
(a)



(b)



(c)



(d)

Figure 7.3: Training loss and validation accuracy on the sampled NCBI development set over epochs for different architectures (separate encoders or a shared encoder, with or without join and hidden layers) pre-trained on synthetic exact-match data before training on the NCBI training set.

To further compare these models, the weights from the similarity layer from the best models are visualized in Figure 7.4. The weights taken from the shared encoder architectures clearly display a diagonal pattern characteristic of an identity matrix. This can be explained by the fact that identical strings have cosine similarity of one, and the cosine similarity operation is equivalent to the operation carried out by the similarity layer when the matrix is an identical matrix. In comparison, the weights taken from the similarity layer of the separate encoder architectures do not display a diagonal pattern. This is probably due to the fact that separate encoders do not generate identical representations for identical strings. When learned by the networks with separate encoders, the equivalence of a cosine similarity operation is thus distributed throughout the layers, if this operation is learned. This may require more training than that required by architectures with a shared encoder, where the learning of the similarity operation is restricted to only the similarity-based ranking module. The grid patterns observed in the ablation architectures may imply that some positions either accept matches in any other positions, or has a high tendency not to accept non-matches. The reason why these patterns are formed is unclear, but it may be related to the positive instances being always exact matches. The two shared encoder models have an inversion of positives and negatives. This is due to the decision layer being one binary classifier, so that positives and negatives can be inverted.

A reason for pre-training failing to improve the performance for the separate encoder full architecture may be related to the number of trainable parameters for the model. The number of trainable parameters for each of the tested architectures are tabulated in Table 7.7. Theoretically, separate encoder architectures have more capacity to learn more complex relationships, as they have twice the amount of trainable parameters at the encoding stage. However, the separate encoder full model fails to do so perhaps because it does not get more training that is required with the increase of number of parameters.

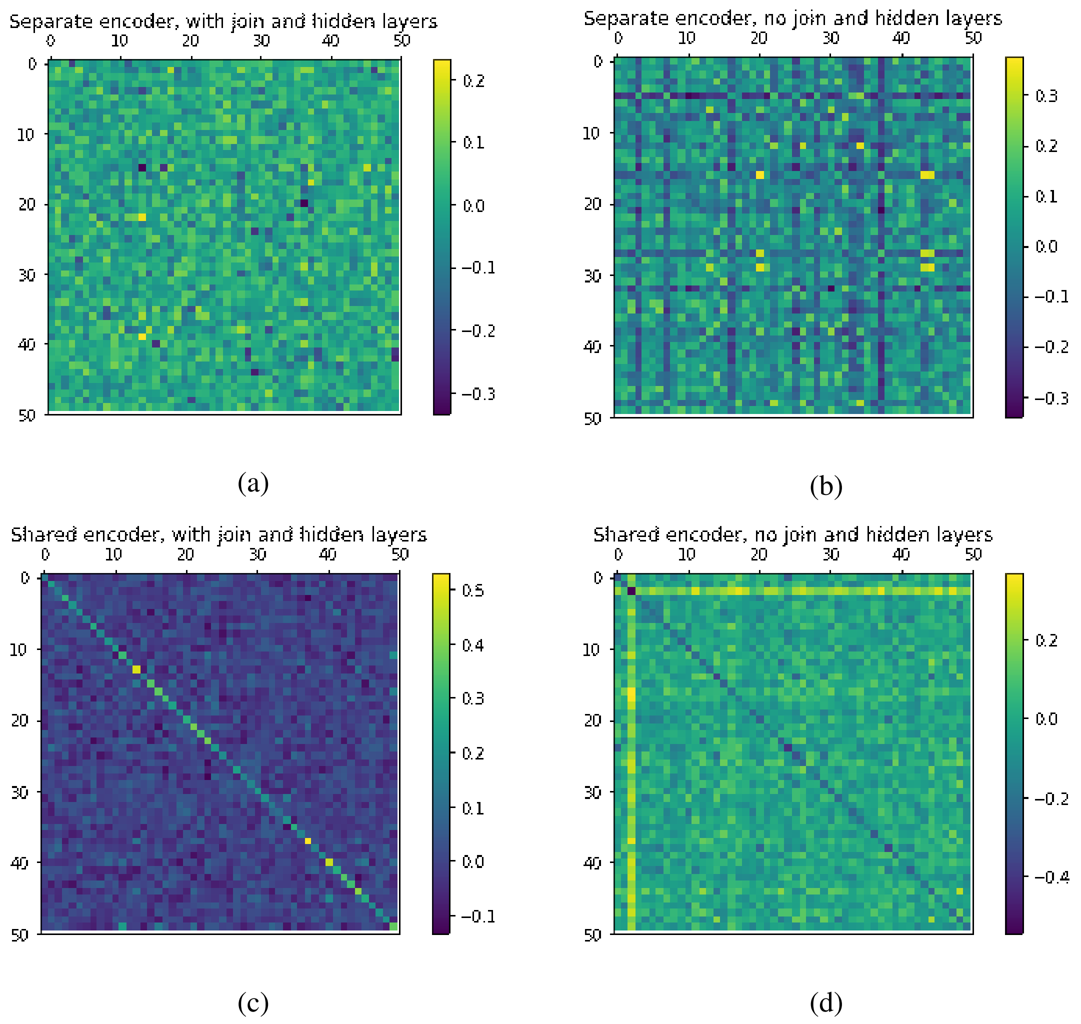


Figure 7.4: Visualization of weights from the similarity layers of pre-trained models. The weights are taken from the best models for each architecture as judged by the accuracy on the NCBI development set.

For error analysis, the predictions made by the best performing architecture (the full model with a shared encoder) at the end of the pre-training and training stages are analyzed. At the end of the pre-training stage, the accuracy reached on the sampled NCBI development set is 0.13. All of the correctly mapped mentions are exact matches to the predicted names, with the only exception being the mention "eye abnormalities", which is correctly normalized to `Eye Abnormality`. This level of performance accounts for only 30% of the total number of exact matches in the set. A reason for the model to fail to

Table 7.7: Number of trainable parameters in each of the architectures tested for exact match pre-training.

Architecture	**No. of parameters in the semantic representation module**	**No. of parameters in the similarity-based ranking module**	**Total no. of parameters**
Separate, Full	60,100	9,093	69,193
Separate, Ablation	60,100	2,502	62,602
Shared, Full	30,050	9,093	39,143
Shared, Ablation	30,050	2,502	32,552

predict exact matches may lie in the way the synthetic data was generated; the negatives in the synthetic training data are sampled only once at the beginning of the training. Consequently, during training, the model does not see the "hard negatives", or the pairs it may predict incorrectly. This is supported by some of the false predictions the model makes at the end of the pre-training stage. Some of the falsely predicted concepts have completely different surface forms as the mention, e.g., the name `Echo Virus Infections` is predicted for the mention "familial breast cancer".

At the end of training, the model attained an accuracy of 0.35, which is still lower than the exact match baseline 0.39. Most of the correct predictions are exact matches, except for a couple of common cases of acronyms in the training set. The most common case is the acronym span "APC", which occurs 96 times in the training set. With this frequent occurrence in the training set, the model correctly maps it to the concept with the preferred name `Adenomatous Polyposis Coli`, even though the acronym is not listed as a synonym in the MEDIC dictionary.

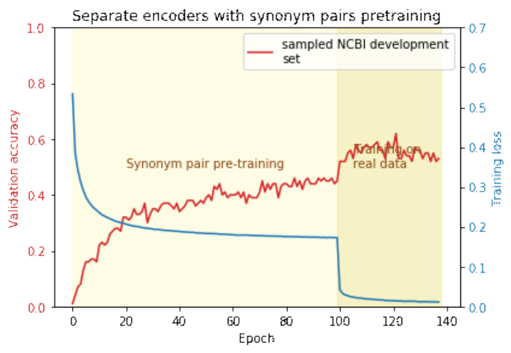
A shared encoder with join and fully connected hidden layer works better than the other architectures under the experimental settings. However, one potential weakness of using the exact match pre-training method on this architecture is that the encoder may not be required to learn to encode the strings. This is because, once the weights of the similarity layer resemble an identity matrix enough, exact matches can be predicted regardless of how the strings are encoded.One solution to this is to pre-train on synthetic data constructed not only using exactly matching pairs, but also synonyms of a concept. Alternatively, the encoder weights could be untied after initial training with tied weights.

7.3.3 Synonym prediction

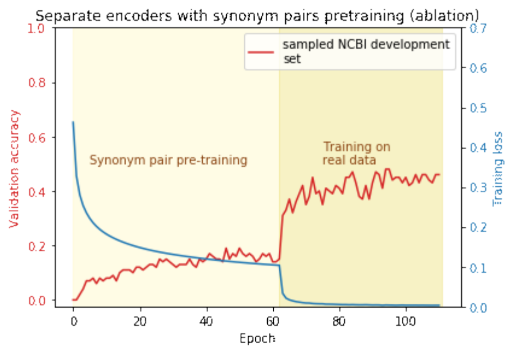
Table 7.8: Highest accuracy obtained during pre-training / after the training completes on the sampled NCBI development set for different architectures when pre-trained on the synonym pair prediction task.

	Full model	**Ablation model**
Separate Encoders	0.46 / 0.62	0.19 / 0.48
Shared Encoder	0.43 / 0.62	0.32 / 0.50

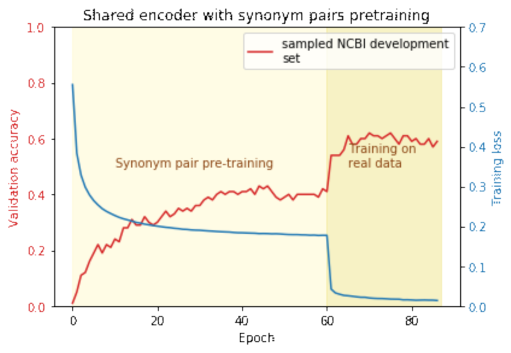
The model is pre-trained on synthetic data constructed on exact match pairs and synonyms. Table 7.8 shows the highest accuracy scores achieved for each of the architectures in the pre-training and training stages. The training loss and validation accuracy on the sampled NCBI development set over epochs for each of the architectures is shown in Figure 7.5. Both full models, with either a shared encoder or separate encoders, achieve the same accuracy of 0.62 at the end of the training. This implies that both of these architectures are capable of learning the mapping of the data to the same degree. The shared encoder architecture, however, is able to learn quicker, as it takes fewer epochs to achieve this performance than the separate encoder full model. This again shows that shared encoder architecture takes less amount of training, possibly due to (1) having half of the



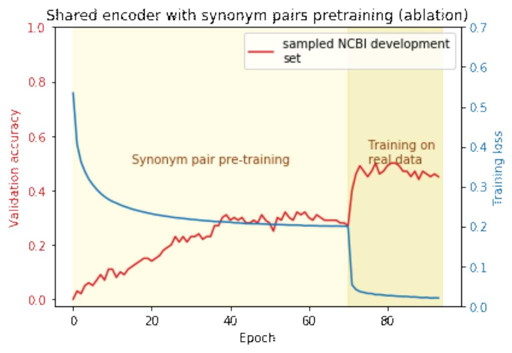
(a)



(b)



(c)



(d)

Figure 7.5: Training loss and validation accuracy on the sampled NCBI development set over epochs for different architectures (separate encoders or a shared encoder, with or without join and fully connected hidden layers) pre-trained on synthetic synonym-pair data before training on the NCBI training set.

amount of trainable parameters for the semantic representation module, and (2) a shared encoder will always encode the same string the same way, thus eliminating the need for the similarity ranking module to learn a mapping of strings encoded by one encoder to those encoded by the other, so that the module can focus instead on learning the mapping of synonyms.

For the ablation models, the shared encoder architecture also achieves better performance than the separate encoder architecture. Unlike the exact match pre-training results for the ablated architectures, which fail to train in the pre-training stage (highest accuracy obtained was 0.01 and 0.00 for the separate and shared encoder architectures, as shown in Table 7.6), the ablated models obtain accuracy scores of 0.19 and 0.32 at the end of synonym pair pre-training. This may be because there is a larger amount of synthetic training data for the synonym pair pre-training than the exact match pre-training. In other words, the similarity matrix alone requires more training data, as there is no join layer to provide more information from the encoded strings. Both ablated models, like the full models, reach about the same level of performance at the end of the training. This shows that, with the join and fully connected hidden layers ablated, the shared and separate encoder architectures are also able to approximate the mapping of the mentions to concepts to about the same level, although the shared encoder architecture requires less training. The weights from the similarity matrix at the end of the synonym pair pre-training are visualized in Figure 7.6. Similar to the weights at the end of exact match pre-training, the weights from the shared encoder architectures display a diagonal pattern characteristic of an identity matrix, whereas the weights from the shared encoder architectures do not. Comparing the weights of the ablation models from the synonym pair pre-training to those from the exact match pre-training (Figure 7.4), the synonym-pair pre-training weights do not have obvious horizontal or vertical streaks as the exact match ones do. This may be because the effect of the streaks cannot be cancelled out for synonym pair pre-training where the positive training examples contain non-identical strings.

The predictions made by the full model with a shared encoder at the end of the pre-training and training stages are analyzed. At the end of the pre-training, the accuracy reaches 0.43, which is higher than 0.39, the exact match baseline on the sampled NCBI development set. Most of the correct predictions have exact matches in the controlled vocabulary. However, some mentions with exact match names in the controlled vocabulary are still incorrectly mapped. For example, the span "cancers" is mapped to `Cancers, Second`, instead of the correct concept `Cancers`. The model also successfully pre-

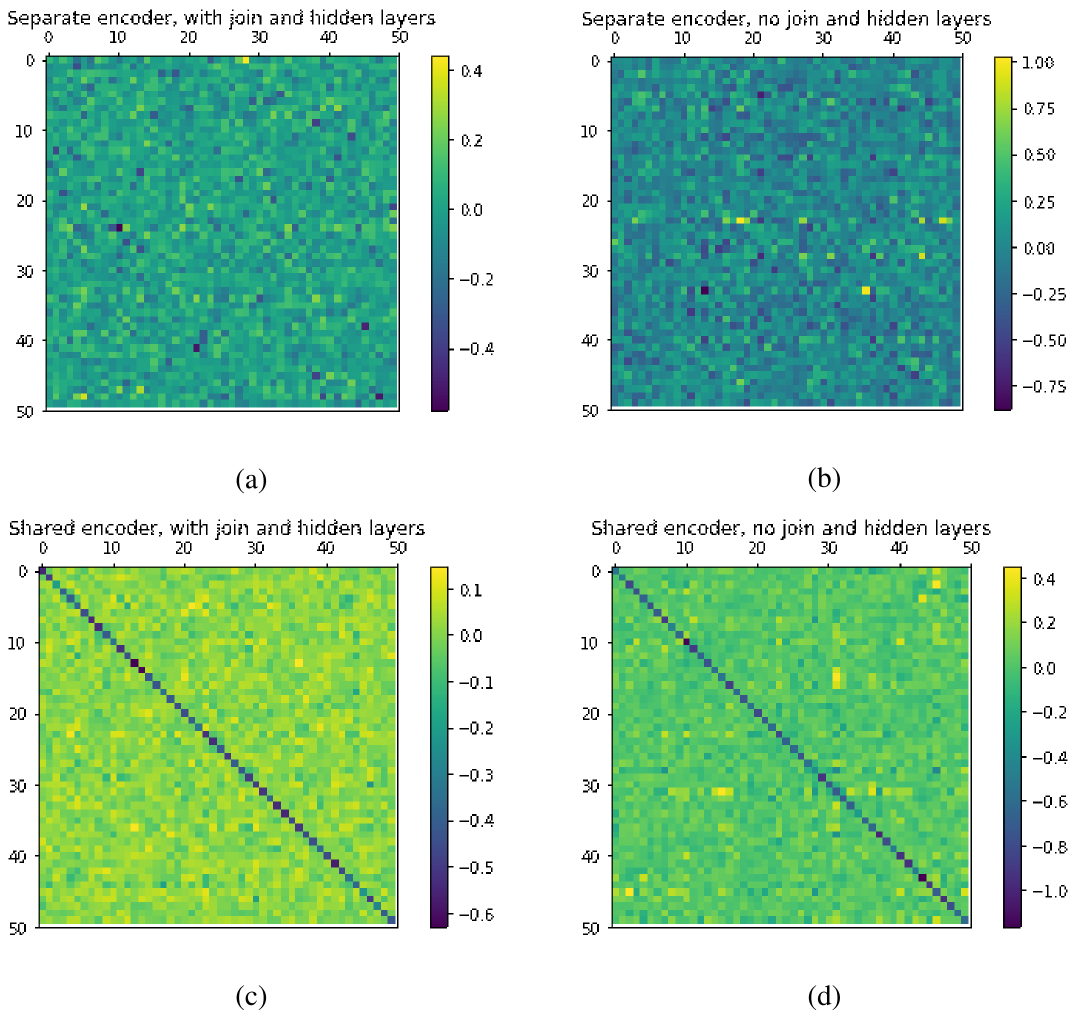


Figure 7.6: Visualization of weights from the similarity layers of models at the end of synonym pair pre-training. The weights are taken from the best models for each architecture as judged by the accuracy on the sampled NCBI development set.

dicts, for some concepts, correct names with shuffled word order from the mention. For example, the mention “cystic kidney disease” is successfully normalized to `Kidney Disease, Cystic`. Interestingly, an exact match alternative exists, but is ranked lowered than the name with shuffled word order. This may be because, in the training data synthesized from synonym pairs, there are more positive instances of shuffled word order than exact matches. This tendency to predict names with shuffled word order decreases as the training data switches to the NCBI training set. This is because, for the real training data, the gold standards of the positive samples shown are selected to be the name with the highest string similarity to the mentions (described in Section 5.3.1). The model, in a couple of instances, correctly predicts correct concept names with partially identical spans. For example, "von Hippel-Lindau disease-associated and sporadic RCCs" is correctly mapped to `von Hippel-Lindau Disease`. This is not always the case, though, as the model fails to map a similar span "von Hippel-Lindau tumor". Apart from complete/-partial lexical match, the model also shows evidence of being able to predict based on semantic meaning. For example, the span "inherited breast cancer" is correctly mapped to the concept name `BREAST CANCER, FAMILIAL BREAST CANCER, FAMILIAL MALE, INCLUDED`, though there is no mention of "inherited" in this name or the synonyms of this concept, nor is this instance in the training set.

At the end of the training, more exact matches are successfully predicted as compared with the predictions at the end of the pre-training. However, not all mentions with exact match concepts are successfully mapped. The model also seems less likely to predict for concepts with shuffled word order. For example, the mention "bacterial infections" is normalized correctly to the concept name `Infections, Bacterial` at the end of the pre-training, though it has an exact match alternative `Bacterial Infections`. At the end of the training, however, the model seems to have forgotten some of its pre-training and incorrectly normalizes the mention to `Opportunistic Infections`. Another instance of the model forgetting its pre-training is the incorrect assignment of `Canavan Disease` to "Krabbe disease" at the end of the training whereas this mention is correctly mapped to `Leukodystrophy, Globoid` at the end of pre-training. This is also a case of the model forgetting about its pre-training, as both `Krabbe Disease` and `Leukodystrophy, Globoid` are listed as synonyms for the concept with the preferred term `Leukodystrophy, Globoid Cell`. One potential solution to this problem of the model forgetting its pre-training would be to mix the pre-training data with the NCBI training set. This, however, remains future work.

In comparison to the exact match pre-training results, the model with separate encoders performs better without pre-training, possibly because (1) the pre-training does not really work for this architecture because it has separate encoders. To predict exact matches, the architecture has to learn to either encode strings identically, or to map two different encodings to each other. This is reflected by the accuracy of only 0.20 obtained by the model at the end of the exact match. (2) The exact match pre-training may have brought the weights to an unfavorable position at the start of training on the NCBI training set. The full model with the shared encoder performs better with pre-training with the current experimental setting of 100 epochs with 15 epochs of early-stopping tolerance. However, there is a chance that the non-pretrained model might have performed better if the training had continued.

Synonym pair pre-training works better than exact match pre-training or no pre-training at all, both in terms of time required to attain a given level of performance for the model and the accuracy the model attains at the end of training. The synonym pair pre-training has three advantages over exact match pre-training. One advantage is that the positive instances contain non-identical pairs, which potentially avoids not properly training the encoder for the shared encoder architecture as discussed in Section 7.3.2. Another advantage is that, with the same number of concepts in the controlled vocabulary, more pre-training instances can be generated: For the exact match prediction task, the number of positive training instances for a concept is equal to the number of concept names listed for the concept; for the synonym pair prediction task, the number of positive training instances for a concept is the combinations of 2-elements with replacement. For example, if a concept has two synonyms in addition to the preferred term, three positive training instances can be generated for the exact match pre-training. For the synonym pair pre-training, however, 6 positive training instances can be generated. This larger amount of pre-training data allows the model to be trained on more instances per epoch, and thus a smaller chance of overfitting at the pre-training stage, which is reflected in the results. In total, with the version of MEDIC vocabulary used, there are 131,270 training instances for exact match pre-training, whereas the synonym pair pre-training has an order of magnitude (1,363,072) more training instances[2]. The third advantage is that synonym pair pre-training provides the encoder with information about abbreviations if those are listed as synonyms, and it also establishes connections between synonyms that have completely different surface forms.

Since models that train more easily are preferred, further experiments will focus on the shared model with the join and fully connected hidden layers pre-trained using synonym pair pre-training. A strategy similar in spirit to the synonym pair pre-training is used by Cho et al. (2017), who augment the training data with different word forms and synonyms in the dictionary. The system of Cho et al. (2017), however, make predictions based on the cosine similarity of the word representations of mentions and concept names, and does not consist of neural network for ranking.

7.4 Similarity measures

In this section, the effect of the similarity measure in the ranking module is investigated. The similarity matrix used for measurement of similarity is replaced with the cosine similarity score of the encoded strings, or completely removed. In addition, an ablation architecture with cosine similarity as similarity measurement that does not have a join layer and fully connected hidden layer is tested. The models are trained under two conditions: either no pre-training or pre-trained on the synonym pair prediction task.

[2]The exact match pre-training data is generated from both the MEDIC dictionary and the training set of the NCBI disease corpus. The synonym pair pre-training data is generated from only the MEDIC dictionary.

The highest accuracy scores obtained by these models are shown in Figure 7.7. For ease of comparison, the accuracy obtained by models with a similarity matrix as the similarity measure (from Sections 7.3.1 and 7.3.3) are also included. The training and validation loss, as well as the accuracy on the sampled NCBI development set over epochs for each of the models tested in this section are plotted in Figure 7.8. The validation data is sampled in a similar fashion as the training data, which is described in Section 5.3. Due to this re-sampling of the validation data for every epoch, the validation loss fluctuates more than the training loss.

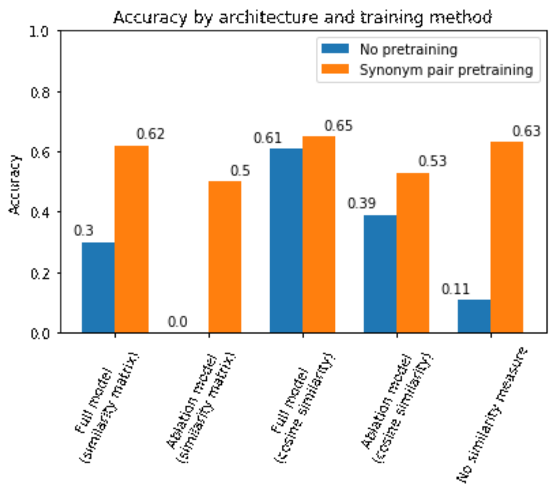
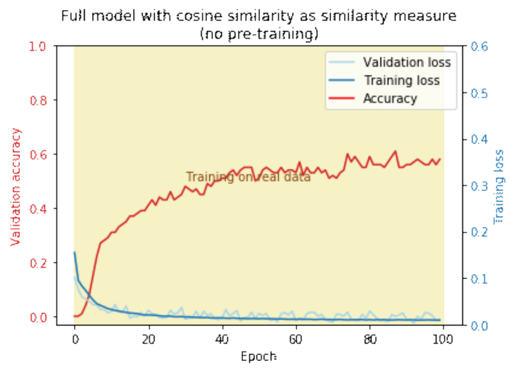
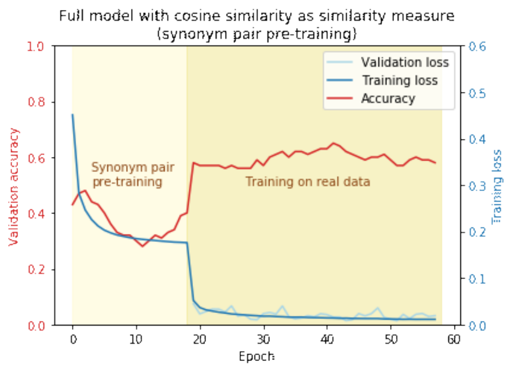


Figure 7.7: Highest accuracy obtained on the sampled NCBI development set for architectures with different similarity measures when directly trained on the NCBI training set or pre-trained on the synonym pair prediction task.

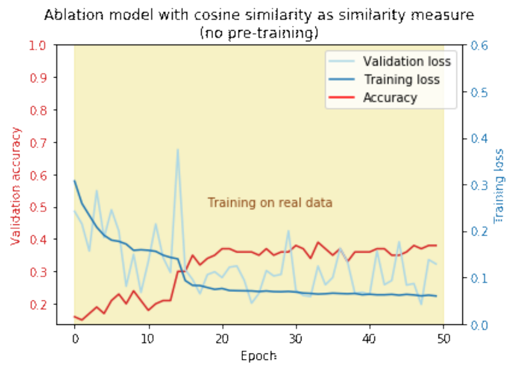
Similar to the results from Section 7.3.3, the comparison of the results from the full model and the ablation model (prediction based only on similarity measure) with cosine similarity as similarity measurement suggests that the encoded strings carry information important to prediction which is lost during similarity score measurement. In Section 7.3.3, the full model with similarity matrix obtains the highest accuracy of 0.62, and the ablation model obtains the highest accuracy of 0.50, while in this experiment, the cosine similarity



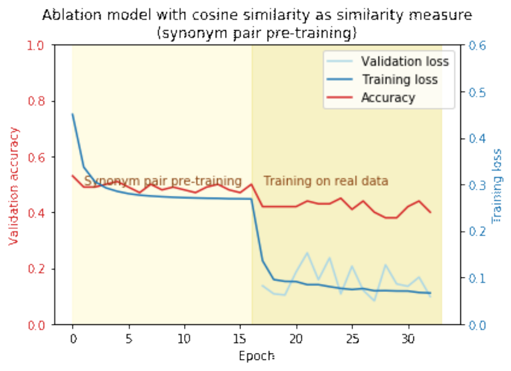
(a)



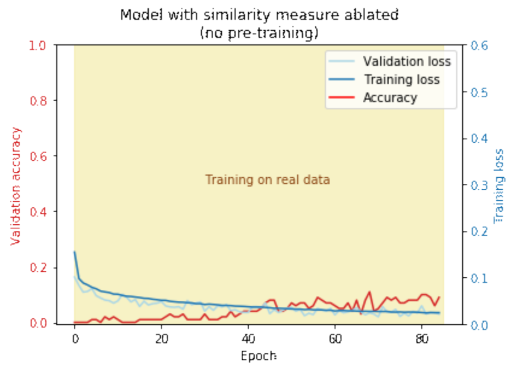
(b)



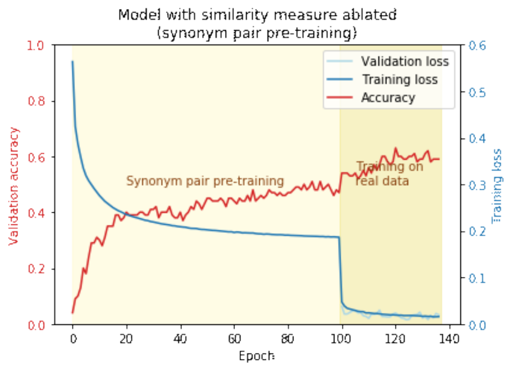
(c)



(d)



(e)



(f)

Figure 7.8: Accuracy, training and validation losses over epochs for different ablation models with shared encoders (model with join and fully connected hidden layer with cosine similarity measurement, model without join and fully connected hidden layer with cosine similarity measurement, or model without any similarity measure but with join and fully connected hidden layer) under different training conditions (either no pre-training or pre-training with synonym pairs).

counterparts of these models obtain accuracy scores of 0.65 and 0.53. Therefore, under the current experimental settings, cosine similarity is a better similarity measure than the similarity matrix.

In terms of the effect of pre-training, the architecture without any string similarity measurement obtains an accuracy of only 0.11 without pre-training, while the same architecture achieves an accuracy of 0.63 with synonym pair pre-training. This again shows that synonym pair pre-training is beneficial to the training of architectures that are harder to train. In addition, all of the models that are pre-trained take less time to achieve their highest score than the non-pretrained ones with the same architecture. This is in agreement with the results from Section 7.3.

Interestingly, the ablation architecture without any similarity measure (illustrated in Figure 5.4c) obtains an accuracy of 0.63, which is only 2 percentage points below the full model with cosine similarity measurement. Since the similarity measure is the only place in the model where word order makes a difference (for CNN filter sizes larger than 1), this small difference in the accuracy scores obtained by these two models may suggest that word order does not make too much of a difference for this particular data set, or that the models do not learn it successfully. This may be due to the fact that, in MEDIC, a lot of names that belong to a concept are just token-shuffled versions of one another, such as the `Neoplasm, Benign` and `Benign Neoplasm` example described in Section 4.1.2. On the other hand, this result is not completely surprising, as bag-of-word models generally work sufficiently well on many tasks. The ablation model, however, takes more than twice the time to train than the full model, so it is much more time-efficient to explicitly use a similarity measure in the ranking module, instead of letting the network take a longer time to learn a similar operation.

Since cosine similarity measure leads to higher accuracy and takes less time to train than the similarity matrix does, further experiments will use this architecture instead of architectures with the similarity matrix.

7.5 Hyperparameter optimization

Hyperparameters are tuned for the selected architecture shown in Figure 5.4a. This architecture has a shared encoder, uses cosine similarity as the similarity measure, concatenates encoded strings and their cosine similarity score, and has a fully connected hidden layer before the decision layer. Several hyperparameters are tuned, viz., the dropout rate of the embedding layer, the number and the sizes of the CNN filters, the activation function used in the CNN layer, and the number of negative samples for each positive instance. Hyperparameter values that lead to better results are fixed while other hyperparameters are further tuned. In this section, the results of this hyperparameter search are reported. All accuracy scores reported are for the best-performing epoch.

7.5.1 Effect of the CNN activation function

The accuracy obtained for models with the tanh or the ReLU function in the CNN layer are plotted in Figure 7.9. For these experiments, 10 filters are used in the CNN layer, 29 negative instances are sampled per positive instance, the dropout rate is set to 0, 0.1, or 0.25, and the size of the CNN filters is set to 2, 3, or 4. From Figure 7.9, it can be seen that there is a trend that the accuracy scores obtained by models with the tanh activation function for the CNN layer are slightly higher than those obtained by models with the ReLU activation. However, due to the small size of the sample, no definitive conclusions can be drawn about the relative performance of these two activation functions. In the following experiments to tune other hyperparameters, the ReLU function is used because it is more computationally efficient. After the values for these other hyperparameters are narrowed down, both the tanh and the ReLU functions are used in the final experiments (reported in Section 7.5.5) to search for better combinations of all the hyperparameters explored.

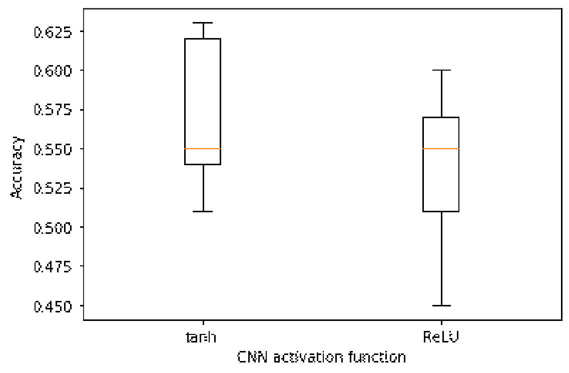
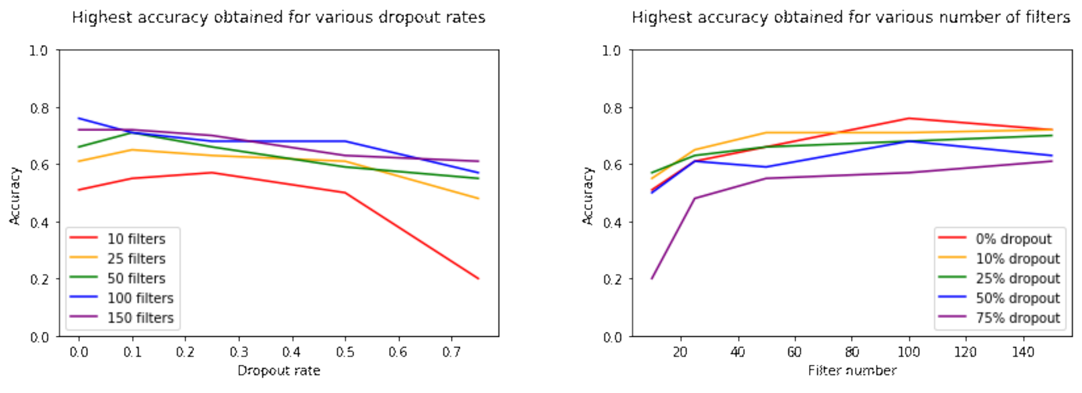


Figure 7.9: Highest accuracy obtained when using the tanh or the ReLU as the CNN activation function.

7.5.2 Effect of the dropout rate and the CNN filter number

The accuracy obtained for varying dropout rates and filter numbers are plotted in Figure 7.10. For these experiments, the CNN filter size is fixed at 1, the ReLU activation is used for the CNN layer, and 29 negative instances are sampled per positive instance. Figure 7.10a shows the change in accuracy as the dropout rate increases for different number of filters. It can be seen that, in the range explored, more filters tend to result in higher accuracy scores, though accuracy scores obtained with 100 and 150 filters do not differ much from one another. This trend may be explained by the fact that filters extract features from the embeddings of strings, and the more filters used in the encoder, the more features extracted in the encoded strings. Consequently, filter number directly relates to how much information is passed on from the semantic representation module to the ranking module.



(a) (b)

Figure 7.10: Highest accuracy scores obtained for varying dropout rates and filter numbers.

The same data is used to plot Figure 7.10b, which shows the change in accuracy as the filter number increases for varying embedding layer dropout rates. It can be seen that lower dropout rates tend to yield better results, especially for larger filter numbers. This may be because smaller dropout rates allow more information to be passed on from the embedding layer to the CNN layer, and this additional information is better processed with more filters.

Under the current experimental settings, higher CNN filter numbers and lower embedding layer dropout rate yield better results. Thus, filter numbers of 100 and 150, and dropout rates of 0 and 0.1, are chosen for further hyperparameter tuning.

7.5.3 Effect of the CNN filter size

The accuracy obtained by models with varying CNN filter numbers (1, 2, 3, and 4) are plotted in Figure 7.11. For these experiments, the CNN filter number is either 100 or 150, the embedding layer dropout rate is either 0 or 0.1, the ReLU activation function is used in the CNN layer, and 29 negative instances are sampled per positive instance. From the figure, it can be seen that no dropout yields slightly better performance than the dropout rate of 0.1. For models with no dropout and 100 filters, performance decreases as the size of the filter increases, whereas for models with the same dropout rate but 150 filters, models with larger filters slightly outperform those with smaller ones. This might be because different filter sizes capture different types of features of the input string, and the data does not contain enough of a certain type of feature, which results in noise or overfitting for larger filter numbers. Along this line of thought, it may be beneficial to use a combination of filter sizes in one single model. However, this may also be just noise, as the numbers are close to one another and only one random seed is used for each model due to limited computational resources. Generally speaking, no visible difference can be inferred about the size and number of filters. Thus, for further experiments, varying CNN filter sizes and numbers are still tested.

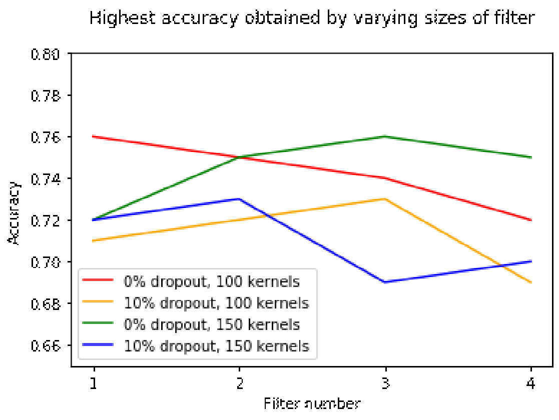


Figure 7.11: Highest accuracy scores obtained for varying filter sizes.

7.5.4 Effect of the number of negative samples

The accuracy obtained for negative sample numbers of 1, 9, 29, 99, and 999 are plotted in Figure 7.12. For the training of these models, no dropout is used for the embedding layer, the CNN layer contains 100 filters of the size of either 1 or 2, and the ReLU activation function is used in the CNN layer. The figure shows that, within the range of negative sample number tested, there is a trend of higher sample number yielding better results.

This trend is more visible for filter size of 2 compared with that of 1. The trend may be due to the fact that larger negative sample numbers better represent the true distribution of the data during training, because the model is given close to 70,000 concept names as candidates during prediction. Although the results show that larger negative sample numbers result in higher accuracy, negative sample number of 99 in addition to that of 999 is used for completeness in further experiments detailed in Section 7.5.5 because the accuracy scores are close to one another.

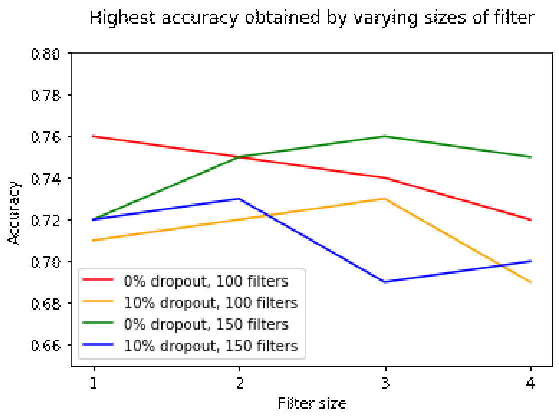


Figure 7.12: Highest accuracy scores obtained for varying number of negative instances.

7.5.5 Evaluation of selected models

Results from the previous sections in this chapter suggest that a larger negative sample number, more filters in the CNN layer, lower dropout rates, and the tanh activation function in the CNN layer may lead to better performing models. The size of the CNN filter, however, does not seem to effect the model performance much. Taking this information into account, another hyperparameter search is conducted, with the negative sample number set to 99 or 999, the size of the CNN filters set to 1, 2, 3, or 4, the number of filters in the CNN set to 100 or 150, and the tanh function used in the CNN layer. The results from this search, along with those from the previous sections in this chapter, are used to select for top performing models for evaluation on the full NCBI development set.

Table 7.9 shows the hyperparameters used for the selected models, as well as the accuracy they obtained on the NCBI development and set. The top performing model has CNN layer with 150 filters of size 3 using the ReLU activation function, and 999 negative instances are sampled per positive instance.

Table 7.9: Hyperparameters of and accuracy obtained by selected models on the NCBI development set.

Number of filters	**Filter size**	**Number of negative instances sampled**	**CNN activation**	**Development set**
100	1	999	tanh	0.802
100	2	999	ReLU	0.832
100	3	99	tanh	0.806
100	4	99	tanh	0.837
100	4	999	tanh	0.837
150	1	999	tanh	0.841
150	3	99	tanh	0.818
150	3	999	tanh	0.811
150	3	999	ReLU	**0.844**
150	4	999	tanh	0.836
150	4	999	ReLU	0.836

7.6 Evaluation on held-out data

The top performing model from Section 7.5.5 is evaluated on the NCBI test set. For comparison, the performance of the second-, third-, and fourth-best models are also reported. Table 7.10 shows the accuracy achieved on the NCBI test set by the top performing models on the development set. The top performing model achieves an accuracy of 0.785 on the test set, outperforming the word embedding baseline of 0.628. This performance, however, is not comparable to the accuracy of 0.861 achieved by `CNN-Li`, or that of 0.878 achieved by the system of Wright et al. (2019).

Table 7.10: Hyperparameters of and accuracy obtained by selected models on the NCBI test set. The accuracy on the development set is included for reference.

Number of filters	**Filter size**	**Number of negative instances sampled**	**CNN activation**	**Development set accuracy**	**Test set accuracy**
100	4	99	tanh	0.837	0.729
100	4	999	tanh	0.837	0.781
150	1	999	tanh	0.841	0.782
150	3	999	ReLU	**0.844**	**0.785**

In this chapter, the baseline of 0.628 was first established by selecting the concept name whose sum of the vector representation of the individual tokens has the highest cosine similarity score with that of a given mention. The embedding used in the rest of the experiments is then selected by an extrinsic evaluation. Due to suboptimal preliminary experiment results that fail to outperform the exact match baselines, pre-training methods are investigated. The synonym pair pre-training task is found to be beneficial to model performance, both in terms of training time required and accuracy attained. In terms of model architecture, a shared encoder architecture is found to outperform a separate one. In addition, taking the cosine similarity score of the encoded strings is found to be a better similarity measure than using a similarity matrix. Finally, with several hyperparameters tuned, a model that outperforms the baseline by 15.7 percentage points is built.

8 Conclusions and future work

This study focused on developing a disease mention normalization system that does not rely on candidate generation and is able to generalize to unseen concepts. To this end, three hypotheses are made with regard to (1) the ability of neural networks to outperform word embedding baselines on this task, (2) the possibility for the normalization system to predict correctly when given the full set of possible concepts, rather than only a heuristically selected subset, and (3) the ability of the neural encoder of the system to generalize to unseen concepts. In light of the experimental results, the following conclusions are drawn:

(1) Neural network-based systems can outperform word embedding baselines. However, with the big number of parameters in neural networks, considerable experimentation and hyperparameter tuning is required for these systems to work better than the baseline methods. Methods such as pre-training using synthetic data can help to improve model performance and shorten the training time required.

(2) The performance of the normalization system does not degenerate when the full set of concepts are given, but is not comparable to those of state-of-the-art systems.

There are some other limitations to this study. The normalization system can predict when given previously unseen concepts. However, the performance of the system under this setting has not been evaluated.

The current implementation of the system is experimental, making prediction on a large set of concepts time-consuming. A substantial speed-up could be achieved by changing some of the operations into Tensorflow tensor operations and executing them on the GPU, where calculating the prediction on a set of concepts can be as efficient as calculating the prediction on a single concept, due to the inherent parallelism.

The system does not have an abstaining mechanism; if there is no correct concept in the given set of concepts, the system does not have the option to output no concept at all. One potential way of addressing this problem is setting a threshold to the predicted value of the selected concept; if the value is low, it suggests a low confidence on the prediction. This solution, however, would require a decision to be made with regard to the threshold value, which requires more experiments.

The system does not predict multiple concepts for one single mention. Multi-concept prediction is challenging because the normalization system assumes that only one correct concept exists in the set of concepts it is given. One potential way of implementing multi-concept prediction would be to set a threshold where any concepts with higher estimated scores are predicted as correct concepts. Another option would be to output the combination of concepts that, when combined, best semantically represent the mention. Both of these two options, however, would require considerable further effort because these approaches of selecting predicted concepts are very different from that used in this study.

There are also more feasible directions for future work. With regard to the current model, hyperparameters can be further tuned, such as using even more negative samples and/or CNN filters, and using a combination of different sizes of filters so that the model captures different types of features. In addition, more architectures can be explored, such as using LSTM as encoders, or using semi-shared encoders for mentions and concepts. With regard to training method, the synonym pair pre-training data can be mixed with the NCBI training set. With regard to the data used, newer versions of MEDIC with more concepts can be used for the NCBI disease corpus. Other normalization data sets, such as the ShARe/CLEF corpus and the Bactria Biotopes data set, could also be used. With regard to string representation, newer text representation methods, such as the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), could be tested. In addition, mention contexts and concept definitions could be included as inputs to the model, as the model only considers mentions and concept names at the moment.

The system described in this thesis is open-source under Apache License 2.0 at `https://github.com/fshdnc/disease_normalization`.

References

R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L.-S. L. Yeh. UniProt: the Universal Protein knowledgebase. *Nucleic acids research*, 32 Database issue:D115–9, 2004.

A. R. Aronson. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. *Proceedings. AMIA Symposium*, pages 17–21, 2001.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

A. Basirat. *Principal Word Vectors*. PhD thesis, Uppsala University, 2017.

S. Bird, E. Klein, and E. Loper. Natural language processing with Python. 2009.

C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *ICML*, 2005.

B. Chiu, G. K. O. Crichton, A. Korhonen, and S. Pyysalo. How to train good word embeddings for biomedical NLP. In *BioNLP@ACL*, 2016.

H. Cho, W. Choi, and H. Lee. A method for named entity normalization in biomedical articles: application to diseases and plants. In *BMC Bioinformatics*, 2017.

F. Chollet et al. Keras. `https://keras.io`, 2015.

A. P. Davis, T. C. Wiegers, M. C. Rosenstein, and C. J. Mattingly. MEDIC: a practical disease vocabulary used at the Comparative Toxicogenomics Database. In *Database*, 2012.

A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegers, T. C. Wiegers, and C. J. Mattingly. The Comparative Toxicogenomics Database: update 2017. In *Nucleic Acids Research*, 2017.

P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals OR*, 134:19–67, 2005.

P. de Matos, A. Dekker, M. Ennis, J. Hastings, K. Haug, S. Turner, and C. Steinbeck. ChEBI: a chemistry ontology and database. In *J. Cheminformatics*, 2010.

L. Deléger, R. Bossy, E. Chaix, M. Ba, A. Ferré, P. Bessières, and C. Nedellec. Overview of the Bacteria Biotope task at BioNLP Shared Task 2016. In *BioNLP*, 2016.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

R. I. Dogan and Z. Lu. An improved corpus of disease mentions in PubMed citations. In *BioNLP@HLT-NAACL*, 2012a.

R. I. Dogan and Z. Lu. An inference method for disease name normalization. In *AAAI Fall Symposium: Information Retrieval and Knowledge Discovery in Biomedical Text*, 2012b.

R. I. Dogan, R. Leaman, and Z. Lu. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. W. Finin. Entity disambiguation for knowledge base population. In *COLING*, 2010.

J. D'Souza and V. Ng. Sieve-based entity linking for the biomedical domain. In *ACL*, 2015.

T. Epelbaum. Deep learning: Technical introduction. *CoRR*, abs/1709.01412, 2017.

A. Ferré, P. Zweigenbaum, and C. Nedellec. Representation of complex terms in a vector space structured by an ontology for a normalization task. In *BioNLP*, 2017.

A. Ferré, L. Deléger, P. Zweigenbaum, and C. Nedellec. Combining rule-based and embedding-based approaches to normalize textual entities with an ontology. In *LREC*, 2018.

M. Francis-Landau, G. Durrett, and D. Klein. Capturing semantic similarity for entity linking with convolutional neural networks. In *HLT-NAACL*, 2016.

B. Georgios-Ioannis. Document reranking with deep learning in information retrieval. Master's thesis, Athens University of Economics and Business, 2018.

M. Gerner, G. Nenadic, and C. M. Bergman. Linnaeus: A species name identification system for biomedical literature. In *BMC Bioinformatics*, 2009.

O. Ghiasvand and R. J. Kate. UWM: Disorder mention extraction from clinical text using CRFs and normalization using learned edit distance patterns. In *SemEval@COLING*, 2014.

W. Golik, M. I. Ur, P. Warnier, and C. Nedellec. Corpus-based extension of termino-ontology by linguistic analysis: a use case in biomedical event extraction. 2011.

J. Hakenberg, M. Gerner, M. Haeussler, I. Solt, C. Plake, M. Schroeder, G. Gonzalez, G. Nenadic, and C. M. Bergman. The GNAT library for local and remote gene mention normalization. In *Bioinformatics*, 2011.

Z. S. Harris. Distributional structure. *WORD*, 10:2-3:146–162, 1954.

F. Hill, K. Cho, and A. Korhonen. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*, 2016.

T. J. P. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clarke, A. Cox, J. A. Cuff, V. Curwen, T. A. Down, R. Durbin, E. Eyras, J. G. R. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehväslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. R. Pocock, S. C. Potter, A. G. Rust, E. Schmidt, S. M. J. Searle, G. Slater, J. A. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. E. Clamp. The Ensembl genome database project. *Nucleic acids research*, 30 1:38–41, 2002.

E. Inan and O. Dikenelli. A sequence learning method for domain-specific entity linking. In *NEWS@ACL*, 2018.

S. Kaewphan, K. Hakala, and F. Ginter. UTU: Disease mention recognition and normalization with CRFs and vector space representations. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014.

I. Karadeniz and A. Özgür. Linking entities through an ontology using word embeddings and syntactic re-ranking. In *BMC Bioinformatics*, 2019.

R. Leaman and Z. Lu. TaggerOne: joint named entity recognition and normalization with semi-Markov models. *Bioinformatics*, 32 18:2839–46, 2016.

R. Leaman, R. I. Dogan, and Z. Lu. DNorm: disease name normalization with pairwise learning to rank. In *Bioinformatics*, 2013.

R. Leaman, C.-H. Wei, and Z. Lu. tmChem: a high performance approach for chemical named entity recognition and normalization. In *J. Cheminformatics*, 2015.

Y. LeCun. Gradient-based learning applied to document recognition. 1998.

H.-C. Lee, Y.-Y. Hsu, and H.-Y. Kao. AuDis: an automatic CRF-enhanced disease normalization in biomedical text. In *Database*, 2016.

H. Li. A short introduction to learning to rank. *IEICE Transactions*, 94-D:1854–1862, 2011.

H. Li, Q. Chen, B. Tang, X. Wang, H. Xu, B. Wang, and D. Huang. CNN-based ranking for biomedical entity normalization. In *BMC Bioinformatics*, 2017.

J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu. BioCreative V CDR task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation*, 2016, 2016.

C. E. Lipscomb. Medical Subject Headings (MeSH). *Bulletin of the Medical Library Association*, 88 3:265–6, 2000.

H. Liu and Y. Xu. A deep learning way for disease name representation and normalization. In X. Huang, J. Jiang, D. Zhao, Y. Feng, and Y. Hong, editors, *Natural Language Processing and Chinese Computing*, pages 151–157, Cham, 2018. Springer International Publishing. ISBN 978-3-319-73618-1.

T.-Y. Liu. Learning to rank for information retrieval. In *SIGIR*, 2010.

Y. Lou, Y. Zhang, T. Qian, F. Li, S. Xiong, and D.-H. Ji. A transition-based joint model for disease named entity recognition and normalization. *Bioinformatics*, 33 15:2363–2371, 2017.

Z. Lu, H.-Y. Kao, C.-H. Wei, M. Huang, J. Liu, C.-J. Kuo, C.-N. Hsu, R. T.-H. Tsai, H.-J. Dai, N. Okazaki, H.-C. Cho, M. Gerner, I. Solt, S. Agarwal, F. Liu, D. Vishnyakova, P. Ruch, M. Romacker, F. Rinaldi, S. Bhattacharya, P. Srinivasan, H. Liu, M. Torii, S. Matos, D. Campos, K. M. Verspoor, K. M. Livingston, and W. J. Wilbur. The gene normalization task in BioCreative III. In *BMC Bioinformatics*, 2011.

D. R. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic acids research*, 39 Database issue:D52–7, 2011.

C. D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. Cambridge, England, 2008. Cambridge University Press.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

F. Mehryary, K. Hakala, S. Kaewphan, J. Björne, T. Salakoski, and F. Ginter. End-to-end system for bacteria habitat extraction. In *Proceedings of the 2017 Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics, 2017.

T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a.

T. Mikolov, W. tau Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, 2013b.

T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

A. A. Morgan, Z. Lu, X. Wang, A. M. Cohen, J. Fluck, P. Ruch, A. Divoli, K. Fundel, R. Leaman, J. Hakenberg, C. Sun, H.-H. Liu, R. Torres, M. O. Krauthammer, W. W. Lau, H. D. Liu, C. Hsu, M. J. Schuemie, K. B. Cohen, and L. Hirschman. Overview of BioCreative II gene normalization. *Genome Biology*, 9:S3 – S3, 2008.

N. Naderi, T. Kappler, C. J. O. Baker, and R. Witte. OrganismTagger: detection, normalization and grounding of organism entities in biomedical documents. *Bioinformatics*, 27 19:2721–9, 2011.

S. J. Nelson, W. D. Johnston, and B. L. Humphreys. Relationships in Medical Subject Headings (MeSH). 2001.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

S. Pradhan, N. Elhadad, W. W. Chapman, S. Manandhar, and G. K. Savova. SemEval-2014 task 7: Analysis of clinical text. In *SemEval@COLING*, 2014.

S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou. Distributional semantics resources for biomedical text processing. 2013.

P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2018.

J. W. Ratcliff and D. Metzener. Pattern matching: The Gestalt approach. *Dr. Dobb's Journal*, page 46, 07 1988.

P. Ristoski and H. Paulheim. RDF2Vec: RDF graph embeddings for data mining. In *International Semantic Web Conference*, 2016.

K. E. Roberts, D. Demner-Fushman, and J. M. Tonning. Overview of the TAC 2017 adverse reaction extraction from drug labels track. In *TAC*, 2017.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24:513–523, 1988.

S. Schulz and R. Cornet. SNOMED CT's ontological commitment. 2009.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681, 1997.

A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 2015.

R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng. Zero-shot learning through cross-modal transfer. In *ICLR*, 2013.

F. M. Suchanek. Yago: a core of semantic knowledge. In *WWW*, 2007.

H. Suominen, S. Salanterä, S. Velupillai, W. W. Chapman, G. K. Savova, N. Elhadad, S. Pradhan, B. R. South, D. L. Mowery, G. J. F. Jones, J. Leveling, L. Kelly, L. Goeuriot, D. Martínez, and G. Zuccon. Overview of the ShARe/CLEF eHealth Evaluation Lab 2013. In *CLEF*, 2013.

K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

B. Tang, Y. Wu, M. Jiang, J. C. Denny, and H. Xu. Recognizing and encoding discorder concepts in clinical text using machine learning and vector space model. In *CLEF*, 2013.

H. ter Horst, M. Hartung, and P. Cimiano. Joint entity recognition and linking in technical domains using undirected probabilistic graphical models. In *LDK*, 2017.

M. Tiftikci, H. Sahin, B. Büyüköz, A. Yayikçi, and A. Özgür. Ontology-based categorization of bacteria and habitat entities using information retrieval techniques. In *BioNLP*, 2016.

R. Usbeck, M. Röder, A.-C. N. Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, P. Ferragina, C. Lemke, A. Moro, R. Navigli, F. Piccinno, G. Rizzo, H. Sack, R. Speck, R. Troncy, J. Waitelonis, and L. Wesemann. GERBIL: General entity annotator benchmarking framework. In *WWW*, 2015.

H. M. Wallach. Conditional random fields: An introduction. 2004.

Y. Wang, S. Liu, N. Afzal, M. Rastegar-Mojarad, X. Wang, F. Shen, P. R. Kingsbury, and H. Liu. A comparison of word embeddings for the biomedical natural language processing. *Journal of biomedical informatics*, 87:12–20, 2018.

C.-H. Wei and H.-Y. Kao. Cross-species gene normalization by species inference. In *BMC Bioinformatics*, 2011.

C.-H. Wei, H.-Y. Kao, and Z. Lu. SR4GN: A species recognition software tool for gene normalization. In *PloS one*, 2012.

C.-H. Wei, H.-Y. Kao, and Z. Lu. GNormPlus: An integrative approach for tagging genes, gene families, and protein domains. In *BioMed research international*, 2015.

D. Wright, Y. Katsis, R. Mehta, and C.-N. Hsu. Normco: Deep disease normalization for biomedical knowledge base construction. In *AKBC 2019*, 2019.

Y. Zhang, J. Wang, B. Tang, Y. Wu, M. Jiang, Y. Chen, and H. Xu. UTH_CCB: A report for SemEval 2014 - task 7 analysis of clinical text. In *SemEval@COLING*, 2014.

S. Zwicklbauer. *Robust entity linking in heterogeneous domains*. PhD thesis, Universität Passau, 2017.

S. Zwicklbauer, C. Seifert, and M. Granitzer. From general to specialized domain: Analyzing three crucial problems of biomedical entity disambiguation. 2015.