Yleiset menetelmät tekstinlouhinnassa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Joulukuu 2024
Atte Lietzén

TURUN YLIOPISTO
Tietotekniikan laitos

Atte Lietzén: Yleiset menetelmät tekstinlouhinnassa

TkK-tutkielma, 24 s.
Tietotekniikka
Joulukuu 2024

Tarve käsitellä jatkuvasti kasvavaa määrää digitaalisia tekstejä niiden analysoimiseksi on suuri. Tähän tarkoitukseen on kehitetty ja sovellettu suuri määrä erilaisia menetelmiä, joiden yleisyyden arviointi ja toiminnan tarkastelu on tärkeää.
Tavoitteena tässä tutkielmassa on selvittää mitä yleisiä menetelmiä tekstinlouhinnassa esiintyy sekä tarkastella niiden toimintaa esimerkkien kautta.
Tutkielma toteutettiin kirjallisuuskatsauksena, jolla pyrittiin selvittämään sekä mitkä menetelmät ovat yleisiä, että miten nämä menetelmät toimivat tekstinlouhinnassa. Tuloksista selviää, että yleisiä menetelmiä ovat hierarkkinen klusterointi, tiedon haku tf-idf:ää käyttäen sekä Named-Entity Recognition (NER). Näitä menetelmiä sovelletaan hyvinkin erilaisiin käyttökohteisiin toisistaan merkittävästikin eroavilla tavoilla. Menetelmien toiminta oli usein riippuvaista oikein valitusta käyttökohteesta, pohjamenetelmän oikeanlaisesta käytöstä sekä uusien tapojen kuten muuntajakielimallien hyödyntämisestä.

Asiasanat: tekstinlouhinta, klusterointi, tiedon haku, tf-idf, tiedon poiminta, NER,

Sisällys

1 Johdanto **1**

2 Taustoitus **3**

2.1 Tekstinlouhinta yleisesti 3

2.2 Tekstinlouhinnan käyttökohteet 4

2.3 Tekstin esiprosessointi 5

3 Menetelmien perusteet **6**

3.1 Klusteroivat menetelmät yleisesti 7

3.1.1 Klusterointi yleisesti 7

3.1.2 Hierarkkisen klusteroinnin perusteet 7

3.2 Tiedon haun menetelmät yleisesti 8

3.2.1 Bag-of-words yleisesti 8

3.2.2 tf-idf -perusteet 9

3.3 Tiedon poiminnan menetelmät yleisesti 10

3.3.1 Sanakirja-menetelmät yleisesti 10

3.3.2 Named-Entity Recognition -perusteet 11

4 Menetelmien toiminta tekstinlouhinnassa **12**

4.1 Hierarkkinen klusterointi 12

4.2 Tiedon haku tf-idf:ää käyttäen 16

4.3 Named-Entity Recognition 20

5 Yhteenveto **24**

Lähdeluettelo **25**

Kuvat

3.1 Yleinen klusteroinnin prosessi, muokattu lähteestä [5] 7
3.2 Yksinkertainen tf-idf kuvaus, muokattu lähteestä [10] 9

1 Johdanto

Tekstin määrä digitaalisessa muodossa kasvaa nopeasti ja siten tarve louhia, tutkia ja analysoida kaikkea tuotettua tekstiä on noussut tärkeäksi. Tekstit voivat olla tieteellisiä artikkeleita, esseekirjoituksia tai lyhyitä tilanneselostuksia sekä ne voivat olla millä tahansa olemassa olevalla kielellä tehtyjä. Nämä seikat johtavat vaatimuksiin käsitellä tekstimäärää entistä tehokkaammin hyödyntäen uusimpia menetelmiä ja tekniikoita. [1] Viime aikoina tapahtuneet kehitykset koneoppimisessa ja varsinkin syväoppimisessa ovat vieneet tekstinlouhintaa eteenpäin merkittävästi ja mahdollistaneet suuremman tekstimäärän käsittelyn ja paremmat tulokset tekstiä analysoidessa. [2]

Tämän tutkielman tarkoitus on tutkia valikoituja yleisiä menetelmiä tekstinlouhinnassa. Menetelmien valikointi tehtiin laajasti hakien niistä tietokannoista, missä tekstinlouhinnasta on runsaasti kirjallisuutta saatavilla. Valitut menetelmät eivät ole absoluuttisesti yleisimpiä tekstinlouhinnassa käytettyjä, mutta ne ovat laajasti käytettyjä. Tutkimuskysymykset ovat seuraavat:

- TK1: Mitkä ovat kirjallisuuskatsauksen perusteella yleisiä menetelmiä tekstinlouhinnassa?
- TK2: Miten ensimmäisen tutkimuskysymyksen menetelmät toimivat tekstinlouhinnassa?

Tiedonhaku toteutettiin ACL Anthology-, Web of Science-, IEEE Xplore- ja Google Scholar-tietokannoissa. Menetelmien kartoitukseen käytettiin hakulausekkei-

ta "text mining AND (common methods OR usual methods)"ja "text mining AND (common applications OR widely-used applications)". Kun menetelmät oli kartoitettu, haettiin jokaisesta menetelmästä konkreettisia sovelluksia ja esimerkkejä hakulausekkeilla "[menetelmä] application OR [menetelmä] method", "text mining AND [menetelmät] AND application", "[menetelmä] in text mining"ja "[menetelmä] in text analysis". Luvussa 3 käydään läpi tulosten suodattamista ja suodattamisen kriteerejä.

Tämän johdantoluvun jälkeen toisessa luvussa taustoitetaan tutkielman aihetta käymällä läpi tekstinlouhintaa yleisesti ja määrittämällä mitä se on tässä työssä, mihin sitä voidaan käyttää sekä käydään rajoitetusti läpi tekstin esiprosessointia. Kolmannessa luvussa taustoitetaan valittuja yleisiä menetelmiä käymällä läpi menetelmien perusmuotoja ja niiden perusteita. Neljännessä luvussa käydään läpi menetelmiä tarkemmin ja kirjallisuuden otettujen konkreettisten esimerkkien avulla tutkitaan menetelmien toimintaa tekstinlouhinnassa. Viimeinen eli viides luku on yhteenveto tutkielmasta.

2 Taustoitus

Tässä luvussa taustoitetaan tekstinlouhintaa käymällä sitä läpi yleisellä tasolla, sekä käymällä läpi joitain sen käyttökohteita. Lisäksi tekstin esiprosessointia käydään pintapuolisesti läpi, koska se on tekstinlouhinnan prosessin kannalta tärkeä osa.

2.1 Tekstinlouhinta yleisesti

Tekstinlouhinta (engl. *text mining*) sisältää laajan joukon menetelmiä tekstillisen datan analysointiin tietokoneavusteisesti. On kehittynyt erilaisia suuntauksia sekä oppeja, jonka seurauksena selkeää ja yksitulkintaista määritelmää tekstinlouhinnalle ei ole olemassa. Eri suuntauksia ja oppeja yhdistää kuitenkin tietokoneavusteinen luonnollisen kielen muodossa olevan tekstin prosessointi ja analyysi. Tekstinlouhintaa käytetään esimerkiksi, kun halutaan saada joukosta tieteellisiä artikkeleita eniten esiintyvät termit ja niiden kontekstit tai esimerkiksi kun halutaan luokitella sosiaalisen median julkaisuja niiden tunnesisällön perusteella. Tekstinlouhinta voidaan jakaa alana useaan osaan. Menetelmät voidaan jakaa pääosin kahteen osaan: sanakirja-pohjaisiin ja algoritmisiin eli koneoppiviin.

Sanakirja-pohjaiset menetelmät perustuvat sanojen esiintymistiheyden mittaamiseen ja analyysin tekemiseen käyttäen ennalta määritettyjä sanajoukkoja eli sanakirjoja. Tällaisia menetelmiä käytetään yleisesti tiedon poiminnassa (engl. *information extraction*), jossa tarkoituksena on löytää yleistä tietoa tai suhteita suuresta joukosta dataa, kuten dokumenteista.

Koneoppivat menetelmät jaetaan kahteen joukkoon: ohjattuihin (luokitteleviin) ja ohjaamattomiin (klusteroiviin) menetelmiin. Ohjatut menetelmät perustuvat merkkien eli tägien generointiin tekstistä ja luokitteluun niiden perusteella käyttäen aiempia tägejä sekä tekstilliseen kontekstiin perustuvia luokitteluja. Ohjattuja menetelmiä käytetään yleisesti tiedon haussa (engl. *information retrieval*), jossa tarkoituksena on löytää tekstistä tai muusta datasta tiettyjä ennalta määritettyjä asioita ja jättää mahdollisesti huomiotta osia dataa. Ohjaamattomat menetelmät perustuvat valvomattomiin algoritmeihin, jotka ryhmittelevät tekstiä perustuen tekstin osien sisäiseen samanlaisuuteen. [1]

Termi 'tekstinlouhinta' tarkoittaa tässä työssä luonnollisen kielen mukaisen tekstin käsittelyä jollain menetelmällä niin, että tekstistä saadaan hyödyllistä ja tarkoituksenmukaista informaatiota. Täten 'tekstinlouhinnan menetelmät' tarkoittaa kaikkia sellaisia sanakirja-pohjaisia ja koneoppivia (sekä ohjattuja että ohjaamattomia) menetelmiä, joita voidaan käyttää luonnollisen tekstin analyysissa saamaan sellaista informaatiota mikä on saatavissa ja haluttua. Kaikki ne menetelmät, joita tässä työssä käsitellään, ovat tekstinlouhinnan menetelmiä, koska niitä voi soveltaa aiemmin kuvatulla tavalla.

2.2 Tekstinlouhinnan käyttökohteet

Tekstinlouhinnalla, niin kuin se aiemmin määritettiin, on laaja joukko käyttökohteita. Seuraavaksi käydään läpi joitain yleisiä käyttöalueita tekstinlouhinnan eri menetelmille.

Tekstin yhteenveto: selkeän yleiskatsauksen muodostaminen suuresta dokumentista tai dokumenttikokoelmasta. *Ekstraktiivinen* yhteenveto koostuu alkuperäisen tekstin informaatioyksiköistä. *Abstraktiivinen* yhteenveto voi sisältää synteettisiä informaatioyksiköitä mitä alkuperäisessä tekstissä ei esiinny. [3]

Sentimenttianalyysi: Tekstin sentimentin eli esimerkiksi tekstin positiivisuu-

den tai negatiivisuuden löytäminen esimerkiksi käyttäen sanakirja-menetelmiä. [1] Sentimenttianalyysia voidaan hyödyntää esimerkiksi verkkomainonnassa käyttäen hyväksi tuotearvosteluja ja käyttäjien mielipiteitä. [3]

Tekstivirrat: Sosiaalinen media ja sosiaaliset verkostot tuottavat suuren määrän dataa ja tekstiä jatkuvana virtana useista eri aiheista käyttäjien toimesta. Tämän sisällön louhinta on hyvin arvokasta. Dynaamisten palveluiden tekstivirtojen louhinta vaatii kuitenkin ei-standardoidun kielen ymmärtämistä. [3]

2.3 Tekstin esiprosessointi

Ennen varsinaisen louhinnan menetelmien soveltamista, on välttämätöntä esiprosessoida teksti. Esiprosessoinnin toteuttaminen niin, että sovellusalue ja -menetelmä otetaan huomioon, on kriittistä, sillä esiprosessointi voi viedä jopa 80 % koko prosessin työstä. Se, minkälaista esiprosessointia käytetään, riippuu vahvasti louhinnan tulosten aiotusta käyttökohteesta. [4] Tässä työssä ei käsitellä tarkemmin esiprosessoinnin tekniikoita tai menetelmiä. Ne kuitenkin mainitaan sillä ne ovat olennainen osa laajempaa prosessia. Seuraavaksi käydään läpi joitain yleisiä tekniikoita esiprosessointiin:

Tokenisaatio (engl. *tokenization*): Merkkijonojen ja -yhdistelmien hajottaminen useaan osaan eli *tokeniin*, jättäen mahdollisesti joitain merkkejä pois. Tokenien listaa käytetään sitten varsinaisten menetelmien kanssa. [3]

Suodattaminen (engl. *filtering*): Tekstin suodattaminen poistamalla joitain sanoja, joista ei saa paljoa sisältötietoa tekstistä. Näitä sanoja ovat esimerkiksi prepositiot ja mahdollisesti tekstissä useasti esiintyvät sanat. [3]

Juurittaminen (engl. *stemming*): Sanojen juurien (*stem*) löytäminen johdetuista ja taivutetuista sanoista käyttäen kieliriippuvaisia menetelmiä. Juurittaminen mahdollistaa sanojen ryhmittämisen yksiköihin tekstin yksinkertaistamiseksi. [3]

3 Menetelmien perusteet

Tässä työssä tarkastellaan kolmea erilaista yleistä menetelmää, joita käytetään tekstinlouhinnassa niin kuin tekstinlouhinta on aiemmin työssä määritetty. Tarkasteltavien menetelmien yleisyyden määritys absoluuttisesti ei ole mahdollista, sillä täydellisen kirjallisuuskatsauksen ja -analyysin tekeminen ei ole tässä työssä mahdollista.

Tarkasteltavat menetelmät on valittu sen perusteella, kuinka yleisiksi ne tai niistä johdetut soveltavat menetelmät osoittautuivat kirjallisuuskatsauksessa. Kirjallisuuskatsaukseen käytettiin seuraavia tietokantoja: Web of Science, ACL Anthology, Google Scholar ja IEEE Xplore. Syynä näiden käyttöön oli niiden paras kattavuus tekstinlouhinnan ja sen menetelmien osalta. Eniten näistä käytettiin Web of Science - ja ACL Anthology -tietokantoja. Ehtona yleisyydelle oli, että menetelmä esiintyi sellaisenaan tai sovellettuna vähintään sadoissa eri tuloksissa ja että esitetty käyttökohde noudatti tässä työssä esitettyä tekstinlouhinnan määritelmää. Tarkemmin käytetyistä hakulausekkeista on kirjoitettu johdannossa.
Menetelmiksi on valittu seuraavat:

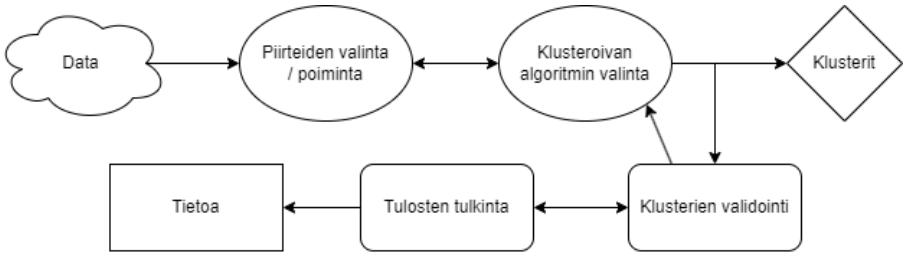
- Hierarkkinen klusterointi
- Tiedon haku tf-idf:ää käyttäen
- Named-Entity Recognition

Tässä luvussa esitellään näitä menetelmiä yleisellä tasolla. Luvussa 4 käydään läpi menetelmien toimintaa tekstinlouhinnassa tarkemmin.

3.1 Klusteroivat menetelmät yleisesti

3.1.1 Klusterointi yleisesti

Klusterointi eli ohjaamaton luokittelu tarkoittaa rajallisen ja merkitsemättömän datan luokittelua diskreetteihin osajoukkoihin eli klustereihin käyttäen datan yksiköiden keskinäistä samanlaisuutta. Tavoitteena klusteroinnissa on, että kun data on jaettu klustereihin käyttäen jotain samanlaisuuden mittaa, on klusterin yksiköiden keskinäinen samanlaisuus suurempi kuin samanlaisuus muiden klustereiden yksiköihin. Yksiköiden sekä kokonaisten klusterien samanlaisuus ja erilaisuus on aina selkeästi mitattavissa ja matemaattisesti esitettävissä. [5] Kuvassa 3.1 esitetään yleistetty klusteroinnin prosessi alusta loppuun.



Kuva 3.1: Yleinen klusteroinnin prosessi, muokattu lähteestä [5]

3.1.2 Hierarkkisen klusteroinnin perusteet

Klusterointia ja varsinkin hierarkkista klusterointia käytetään laajasti eri sovellusalueilla ja se on koneoppimisen ja tekstinlouhinnan yleisimpiä käytettyjä menetelmiä. [5] Hierarkkisen klusteroinnin perusteena on sisäkkäisten klusterien muodostama puu, jossa klustereilla on keskinäinen hierarkia. Hierarkkiseen klusterointiin on kaksi lähestymistapaa: agglomeratiivinen ja divisiivinen. Agglomeratiivisessa tavassa aloitetaan pienistä klustereista ja muodostetaan suurempia klustereita, kunnes on vain yksi kaiken datan kattava klusteri. Divisiivessä tavassa lähtökohtana on kaiken datan kattava klusteri, joka jaetaan pienempiin ja jakamista jatketaan kaikille klus-

tereille, kunnes jokaisessa on yksi datayksikkö. Molemmissa tavoissa lopputuloksena on datan klusteroinnin rakennetta kuvaava puukaavio. Näistä tavoista agglomeratiivinen on yleisempi hierarkkisen klusteroinnin sovelluksissa. [6]

Klustereiden toisistaan erottamisessa olennaista on niiden erojen löytäminen. Kaksi yleistä ja yksinkertaista tapaa siihen on yksittäinen linkki (engl. *single linkeage*) ja täydellinen linkki (engl. *complete linkeage*). Yksittäisessä linkissä kahden klusterin etäisyys määritetään niiden lähimpänä olevien yksiköiden etäisyytenä. Täydellinen linkki toimii päinvastoin, ja siinä etäisyys on klusterien kaukaisimpien yksiköiden etäisyys toisistaan. Muitakin menetelmä on, muun muassa keskiarvon linkki (engl. *average linkeage*) sekä mediaanin linkki (engl. *median linkeage*). [5]

3.2 Tiedon haun menetelmät yleisesti

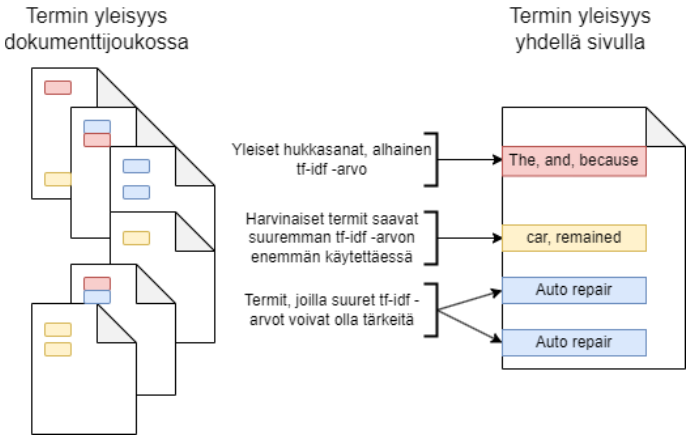
Tiedon haku on noussut tärkeäksi osaksi tekstin ja muun datan louhintaa. Tavoitteena tiedon haussa on tunnistaa ja hakea tietoa, mikä on tehdyn haun kannalta tärkeää. Tiedon hakuun tarkoitetut menetelmät ovat perinteisesti perustuneet tekstin ja haun vertaamiseen ja arviointiin. Viime aikoina, koneoppivat ja varsinkin syväoppivat menetelmät ovat vieneet tiedon haun menetelmien käyttö eteenpäin ja parantaneet vanhemmissa lähestymistavoissa olleiden ongelmien ratkaisua. [7]

3.2.1 Bag-of-words yleisesti

Bag-of-words on yleisesti käytetty ja sovellettu ohjattu eli luokitteleva menetelmä. Menetelmän ydinidea on kvantisoida datasta otettuja pisteitä, kuten tekstistä sanoja tai kuvasta objekteja, luokittelua varten. [8] Tekstiin soveltaessa lasketaan arvokkaaksi määritettyjen sanojen tai sanojen yhdistelmien esiintymismäärät tekstissä. Näin saadaan kartoitettua teksti moniulotteiseen ja rajoitettuun vektoriavaruuteen. Riippuen siitä, mikä käyttökohde on sekä muista kontekstiriippuvaisista asioista, ku-

ten vektoriulottuvuuksien määrä, voidaan luokitteluun käyttää erilaisia algoritmeja. Esimerkkejä näistä ovat esimerkiksi naiivi-Bayes-luokittelija, logistisen regression mallit sekä erilaiset tukivektorikoneet. Näitä algoritmeja ei tarkemmin tässä työssä käydä läpi. [9] Lisäksi on huomattava, että syväoppivat lähestymistavat ovat vähentäneet vanhempien algoritmien ja lähestymistapojen käyttöä. [7].

3.2.2 tf-idf -perusteet



Kuva 3.2: Yksinkertainen tf-idf kuvaus, muokattu lähteestä [10]

Tf-idf (*term frequency - inverse document frequency*) on laajennettu versio Bag-of-words -menetelmästä ja tarkemmin sanottuna harvasta (engl. *sparse*) Bag-of-words -menetelmästä. [11] Johtuen yksinkertaisimpien Bag-of-words mallien rajoituksista, on tf-idf yksi yleisimpiä käytettyjä yleisyyspainottavia menetelmiä. [10] Perusideana on, että sanojen esiintymismääriä (engl. *term frequency*) tarkastellessa otetaan huomioon sanojen esiintymismäärät koko tekstissä käyttäen niiden käänteistä dokumentin yleisyyttä (engl. *inverse document frequency*). Tällöin pystytään identifioimaan ne sanat, jotka vääristävät tuloksia koko tekstille. [11] Sanat, jotka ovat yleisimpiä, kuten artikkelit, prepositiot tai muuten sattumalta yleiset mutta analyysin kannalta merkityksettömät sanat, menettävät siis merkitystään, kun taas ne sanat, jotka esiintyvät harvoin tai vain kerran, saavat suuremman painoarvon.

Näin saadaan piirre-esitys tekstistä, jota voidaan sitten syöttää esimerkiksi klusteroiville menetelmille. [10] Kuvassa 3.2 esitetään yksinkertainen kuvaus tf-idf-arvon määrittämisestä termeille.

3.3 Tiedon poiminnan menetelmät yleisesti

Tiedon poiminta ja sen menetelmät ovat nousseet tärkeäksi osaksi tekstinlouhintaa varsinkin analysoitavien tekstimäärien lisääntyessä. Tavoitteena tiedon poiminnassa on tunnistaa tekstistä tai muusta datasta yleisempää tietoa tai sisäisiä suhteita, eli se eroaa tiedon hausta siinä, että lähtökohtaisesti ei olla hakemassa tiettyjä, ennalta määriteltyjä asioita vaan tavoitteena saada laajemmin tietoa irti tekstistä. Esimerkiksi voidaan analysoida uutistekstejä, jotka käsittelevät samaa tapahtumaa ja poimia tietoa niin, että löydetään yhteyksiä tekstien välillä mitkä eivät olisi tulleet ilmi yksittäistä uutista käsiteltäessä. Tiedon poiminnan menetelmiä on kehitetty jo 1970-luvulta, mutta viime aikoina koneoppiminen ja syväoppiminen on vienyt menetelmien eteenpäin merkittävästi ja mahdollistanut laajempien tekstijoukkojen louhintaa [12]

3.3.1 Sanakirja-menetelmät yleisesti

Sanakirja-menetelmät ovat sellaisia menetelmiä, missä hyödynnetään jo valmiiksi tehtyjä sanakirjoja, mitkä sisältävät käyttökohteeseen sopivaa dataa eli sanoja ja niiden taivutusmuotoja. Nämä sanakirjat voivat olla yleisiä ja julkisesti saatavilla olevia, tai sitten ne voivat olla mittatilaustyönä tehtyjä vain mahdollisesti yksittäistä käyttökohdetta varten. [1] Tällaiset menetelmät ovat löytäneet laajasti käyttökohteita esimerkiksi psykologian ja kielitieteen alalta, missä niitä voidaan käyttää tekstien tunneanalyysissa eli sentimentaalisessa analyysissä tai kielitieteellisessä analyysissä tekstin sanojen ja sanakirjan sanojen taivutusta vertailemalla. Erilaisia sanakirja-

menetelmiä voidaan myös käyttää useiden muiden menetelmien, kuten erilaisten klusterointien tai Bag-of-words -sovellusten, kanssa esimerkiksi sentimenttianalyysin toteutuksessa. [13]

3.3.2 Named-Entity Recognition -perusteet

Named-Entity Recognition (NER) tarkoittaa sellaisia menetelmiä, missä on painotettuna tekstin nimetyt entiteetit (engl. *named entity*), jotka riippuvat tekstin kontekstista. Nimetty entiteetti voi olla esimerkiksi paikka, ihmisen nimi tai muu tietyn konseptin mukainen asia. Lisäksi NER:issä sanojen järjestyksellä on väliä (vaikkakin painotus on vain nimetyissä entiteeteissä), sillä sanojen merkitys voi riippua koko lauseista. Esimerkiksi 'Washington' voi tarkoittaa henkilöä tai useampaa paikkaa, jolloin pitää tarkastella sanan esiintymisen kontekstia. [14] Olemassa olevat NER-menetelmät voidaan jakaa kahteen osaan: sääntöpohjaisiin ja koneoppiviin. Sääntöpohjaiset menetelmät noudattavat perinteisiä sanakirja-menetelmiä eli vaativat ihmisen käsityönä tehtyjä rajallisia ja tarkkoja sanakirjoja. Tällaisten menetelmien toteuttaminen vaatii siis sovellusalan asiantuntijuutta. Koneoppivat menetelmät taas perustuvat koneoppivan järjestelmän automaattiseen sääntöjen ja kaavojen tunnistukseen. Tällaiset menetelmät vaativat kuitenkin suuret määrät käsin merkittyä dataa, jonka tekeminen ja saaminen voi olla hankalaa ja kallista. [15] NER ja sen sovellukset ovat laajasti käytettyjä, koska ne soveltuvat muun muassa sentimenttianalyysiin ja entiteettiin tehtyjen viittauksien tarkasteluun. Lisäksi kone- ja syväoppivan kieliteknologian kehitykset ovat vieneet NER:iä merkittävästi eteenpäin. [14]

4 Menetelmien toiminta tekstinlouhinnassa

Seuraavaksi tarkastellaan luvussa 3 käytyjen menetelmien toimintaa tarkemmin tekstinlouhinnassa. Tarkastelu tehdään käymällä jokaisesta menetelmästä läpi niiden toimintaa sekä konkreettisia esimerkkejä, jonka jälkeen niistä tehdään yhteenvedot.

4.1 Hierarkkinen klusterointi

Yleistettynä hierarkkinen klusterointi voidaan esittää prosessina, jossa joukko $X = \{x_1, x_2, x_3..., x_n\}$ jaetaan joukkoon $A = \{a_1, a_2, a_3...a_m\}$, joka toteuttaa ehdon $a_i \cap a_j = \emptyset$, eli joukon A klusterit ovat keskenään niin erilaisia, että ne eivät jaa mitään keskenään tarkasteltavien ominaisuuksien osalta ja ovat riittävän erilaisia muodostaakseen klustereita. Tämä joukkojen jakaminen tehdään useaan kertaan niin monta kertaa, että on saatu muodostettua puu. [6]

Klusterien muodostamisessa käytetään datan yksiköiden etäisyyksiä, jotka kuvaavat niiden keskinäistä samanlaisuutta ja siten yksiköiden jakoa klustereihin. Muodostamisessa voidaan käyttää samanlaisuutta sellaisenaan eli klusteroimalla sellaiset yksiköt, joilla on keskenään pienin etäisyys, tai vaihtoehtoisesti käyttää yksiköiden erilaisuutta, jota voidaan pitää myös 'vahvempana' etäisyytenä. Käytännössä näitä etäisyyksiä mitataan arvioimalla niitä vektorien avulla, jotka muodostetaan tarkastelemalla haluttuja ominaisuuksia klustereissa. Esimerkiksi Bag-of-Words -

menetelmää käyttämällä saadaan mallinnettua teksti vektoriavaruudessa vektoreina, jolloin pystytään tekemään vertailuja tekstin osien välillä käyttäen näitä vektoreita.

Esimerkki samanlaisuutta käyttävästä tavasta on erityisesti dokumenttien louhinnassa paljon käytetty kosini-samanlaisuus (engl. *cosine similarity*), joka on määritelty kahdelle vektorille x_a, x_b seuraavasti:

$$s = (x_a, x_b) = cos(\theta) = \frac{x_a \cdot x_b}{\|x_a\| \cdot \|x_b\|}$$

Esimerkki erilaisuutta käyttävästä tavasta on yleinen Minkowskin etäisyys (jota voidaan soveltaa myös samanlaisuutta käyttävästi), joka on määritetty kahdelle vektorille x_a, x_b seuraavasti:

$$L_p(x_a, x_b) = (\sum_{i=1}^{n} \|x_{i,a} - x_{i,b}\|^p)^{1/p}, \forall p \geq 1, p \in \mathbb{Z}^+$$

Minkowskin etäisyydessä n on vektorien $x_a, x_b ... x_n$ kokonaismäärä ja p on järjestysluku. [16] Erilaisia tapoja etäisyyksien mittaamiseen on useita, mutta tässä työssä niitä ei tarkastella yksityiskohtaisesti. Olennaista on, että tekstinlouhintaan soveltamisessa yhtä oikeaa tapaa ei ole, vaan käyttökohde ja tavoite on määrittävä tekijä.

Seuraavaksi tarkastellaan alan kirjallisuuden perusteella joitain sovelluksia ja sovelluskohteita hierarkkiselle klusteroinnille sekä tarkastellaan käytön tuloksia kirjallisuuden perustella. Lisäksi näistä esimerkeistä hierarkkisen klusteroinnin käytöstä tekstinlouhinnassa tehdään havaintoja ja huomioita.

Chrupala [17] esittää hierarkkisen klusteroinnin sovelluksen eri kielien sanatyyppien (esim. adjektiivit, adverbit, verbit) klusterointiin käyttäen Jensen-Shannon-poikkeavuutta klusterien etäisyyden mittaamiseen. Sovellus toimii myös täysin erillisten klusterien määrittämiseen mahdollisimman pienellä tiedon menetyksellä. Sovelluksessa puun solmut sisältävät normalisoimattoman arvon, joka on yhdistelmä sanatyyppiä sekä luokkaa mihin sana kuuluu. Jensen-Shannon -poikkeavuutta so-

velletaan, kun solmuja yhdistetään, mutta uuden solmun arvo on aina normalisoimaton. Tämä sovellus sanatyyppin analyysiin hierarkkisella klusteroinnilla toimii paremmin kuin sen vertailukohta, joka käytti enemmän abstraktoivaa tekstin piirreoppimista. Tekijä huomauttaa, että menetelmät on yksinkertainen ja tehoton jos sanatyyppejä on useita satoja. Chrupałan sovellus on siis suhteellisen hyvin toimiva esimerkki hierarkkisen klusteroinnin soveltamisesta rajoitetulla määrällä sanojen luokkia ja tyyppejä.

Can ja Manandhar [18] esittävät hierarkkisen klusteroinnin sovelluksen sanojen pienimpien osien eli morfeemien klusterointiin niiden funktioiden perusteella. Sovelluksessa toteutetaan alhaalta ylös rakennettava puu, jossa solmujen arvot riippuvat morfeemista itsestään, sanan juuresta, sanan muista morfeemeista sekä muista useista muista kielitieteellisistä asioista. Käytännössä muuttujia solmujen saamien arvojen taustalla on siis paljon. Etäisyyksiin sovelluksessa käytetään minimoitavaa Kullback-Leiblerin-poikkeavuutta sekä keskiarvon linkkiä. Sovelluksen testien perusteella se toimi suhteellisen hyvin valmiiksi tehdyssä morfologian testissä. Huomattavaa on kuitenkin, että sovellus toimii vaihtelevasti riippuen niin kielestä kuin morfeemien tyypistä, eli tällainen hierarkkisen klusteroinnin sovellus ei ole yleispätevä.

Chali ja Noureddine [19] esittävät kaksi hierarkkisen klusteroinnin sovellusta dokumenttien ryhmittelyyn, joihin he viittaavat ryhmittelevänä (engl. *grouping*) ja ketjuttavana (engl. *chaining*) algoritmina. Ryhmittelevä algoritmi perustuu klusterointiin vaiheittain niin, että klusterien sisäinen semantiikkaan ja tekstin samanlaisuuteen perustuva vahvuus on mahdollisimman suuri, eli puu rakennettaan 'siirtäen' dataa klusterista toiseen, jotta klusterien vahvuus on mahdollisimman suuri. Ketjuttava algoritmi perustuu ennalta määritetyn kynnysarvon käyttöön niin, että tekstit, joiden läheisyys on kynnysarvon yläpuolella, klusteroidaan yhteen, kun taas ne jotka ovat lähellä kynnysarvoa ylittämättä sitä, toimivat uuden klusterin alkuna,

jolloin klusterit voidaan 'ketjuttaa' keskenään hierarkkisesti. Semanttisen yhteyden samanlaisuuden testeissä molemmat algoritmit osoittautuivat paremmiksi kuin vertailukohteena ollut EM-algoritmi ja ne saavuttivat yli 90 % tarkkuuden. Sanojen samanlaisuuden testissä oli vain ryhmittelevä algoritmi, joka toimi huomattavasti paremmin kuin vertailukohteina olleet k-means- ja EM-algoritmi. Nämä hierarkkista klusterointia soveltavat algoritmit osoittavat, että erilaiset sovellukset voivat toimia perinteisimpiä ja yleisimpiä algoritmeja vastaan, mutta ne voivat olla hyvinkin käyttörajotteisia, sillä ketjuttava algoritmi oli täysin sopimaton toiseen testiin.

Chen, Tseng ja Laing [20] esittävät lähestymistavan dokumenttien ryhmittelyyn hierarkkisen klusteroinnin avulla käyttäen 'sumeaa' assosioiden louhintaa. 'Sumea' (engl. *fuzzy*) viittaa tässä kontekstissa tekstiin, jolta puuttuu selkeä ennalta määritelty rakenne. Tässä sovelluksessa tekstin sanoille lasketaan sanan esiintymismääriin perustuvia sumeusarvoja, joiden perusteella muodostetaan kohdejoukkoja (engl. *itemset*), jotka määrittävät ehdokasklustereita (sekä vanhempisolmuja ja lapsisolmuja). Tämän jälkeen keskenään liian samanlaiset ehdokasklusterit yhdistetään eli käytännössä poistetaan joitain ehdokasklustereita. Sitten kaikki vanhemmattomat ehdokasklusterit poistetaan. Kun puuta on 'siistitty', rakennetaan lopullinen puu ylhäältä alas. Sovellus toimii vertailukohtiinsa nähden paremmin testeissä, tosin erot kaikkiin muihin menetelmään eivät ole suuria. Tämä sovellus on esimerkki, että hierarkkisessa klusteroinnissa klusterien muodostaminen hieman monimutkaisemmin ja niiden 'siistiminen' voi olla hyödyllistä. Lisäksi tekstin osien esiintymismäärien tarkastelu voi olla hierarkkiseen klusterointiin yhdistettynä tällaisella tavalla edullista lopputuloksen kannalta.

Tekstinlouhinnassa hierarkkisella klusteroinnilla on siis monia sovelluskohteita. Kuten edellä esitetyistä alan kirjallisuuden esimerkeistä nähdään, sitä voidaan käyttää hyvinkin erilaisissa käyttökohteissa, kuten sanojen tyyppien, sanojen yksittäisten osien sekä dokumenttien klusterointiin. Can ja Manandhar [18] sekä Chrupala

[17] esittävät miten valmiita tapoja etäisyyksien laskemiseen voi käyttää onnistuneesti. Toisaalta Chen, Tseng ja Laing [20] osoittaa täysin räätälöidyn laskemistavan toimivuuden ja Chali ja Noureddine [19] osoittavat muidenkin kuin yksinkertaisten etäisyyksien laskennan mahdollisuuden toimia hyvin. Näistä konkreettisista esimerkeistä näkyy myös, että klusterien puun rakentaminen pitää toteuttaa käyttötapauslähtöisesti, sillä sekä agglomeratiiviset että diviiviset sovellukset toimivat käyttötapauksissaan.

4.2 Tiedon haku tf-idf:ää käyttäen

Kuten aiemmin todettu, tf-idf arvioi sanan merkitystä suoraan suhteessa siihen, kuinka usein se esiintyy tekstissä, mutta käänteisesti kuinka usein se esiintyy yleisesti dokumenteissa. tf (engl. *term frequency*) eli sanan yleisyys lasketaan seuraavasti:

$$tf_{i,j} = \frac{N_{i,j}}{\sum_k N_{k,j}}$$

missä $N_{i,j}$ on sanan i yleisyys tekstissä d_j ja k on muiden sanojen yleisyys samassa tekstissä. idf (engl. *inverse document frequency*) eli käänteinen yleisyys kaikissa dokumenteissa lasketaan seuraavasti:

$$idf_i = \log \frac{|D|}{|j : t_i \in d_j|}$$

missä $|D|$ on kaikkien dokumenttien määrä ja $|j : t_i \in d_j|$ tarkoittaa niiden dokumenttien määrää, jotka sisältävät sanan i. Lopullinen tf-idf -arvo sanalla saadaan siis seuraavasti:

$$tf - idf_{i,j} = tf_{i,j} \times idf_i$$

Jos sanalla on siis korkea tf -arvo ja matala idf -arvo, on sana kriittinen. Tällöin sana esiintyy siis kaikissa dokumenteissa harvoin ja ei ole yleinen, mutta esiintyy

tutkitussa tekstissä paljon, joten sillä on todennäköisemmin merkitystä tiedon haun kannalta. Jos taas idf-arvo on korkea (riippumatta tf -arvon suuruudesta), sana esiintyy yleisesti kaikissa dokumenteissa, jolloin tf-idf-arvo on lähellä nollaa ja sana todennäköisemmin ei ole haun kannalta kriittinen. [21]

Edellä esitetyt kaavat tf-idf-arvon selvittämiseen ovat yksinkertaisia esimerkkejä, ja käytännön sovelluksissa esiintyy usein muokattuja tai laajennettuja laskemistapoja arvojen selvittämiseen.

Seuraavaksi tarkastellaan alan kirjallisuuden perusteella joitain sovelluksia ja sovelluskohteita tf-idf:ille sekä tarkastellaan käytön tuloksia kirjallisuuden perustella. Lisäksi näistä esimerkeistä tf-idf:n käytöstä tekstinlouhinnassa tehdään havaintoja ja huomioita.

Gebre, Zampieri, Wittenburg et al. [22] esittävät tf-idf-sovelluksen äidinkielen tunnistamiseen yhdistäen lineaarisia luokittelijoita tf-idf:n käyttöön. Tavoitteena sovelluksessa on yrittää tulkita ei-englanninkielisen kirjoittajan äidinkieli englanninkielisen esseen perusteella. tf-arvon laskemista varten sanaksi lasketaan n-grammit, sanat ja puheosat. Lisäksi tf-arvo, tai tässä tapauksessa tf-paino, lasketaan kaavalla $wf_{i,j} = 1 + log(tf_{i,j})$. idf-arvo lasketaan aiemmin esitetyllä tavalla. Kolmea lineaariluokittelijaa käytettiin: tukivektorikone, logistinen regressio ja perseptroni. Tulosten perusteella tämä tf-idf-sovellus toimii parhaiten tukivektorikoneen ja logistisen regression kanssa, hieman huonommin perseptronin kanssa. Lisäksi selviä eroja ilmentyi, sillä paras tarkkuus oli saksankielisille (95 %) ja huonoin hindinkielisille (72 %), eli ero parhaan ja huonoimman välillä tarkkuudessa oli 23 %. Tulokset myös osoittavat, että tehtävä suoritettiin paremmalla tarkkuudella tf-idf-painoja käyttämällä. Tämä sovellus toimii paremmin kuin sen aiemmin tehdyt vertailukohdat, mutta sen suurehkot erot eri kielien tunnistamisessa jättävät parantamisen varaa. Lisäksi sovellus on yli 10 vuotta vanha, joten huonommin tunnistettujen kielien kannalta parempia jatkokehitelmiä sovelluksesta voi olla olemassa. Sovellus osoittaa

kuitenkin verrattain yksinkertaisen menetelmän toimivuuden.

Chen [23] esittää laajentavan tf-idf-sovelluksen avainsanojen poimintaan laajoista tekstijoukoista, jota testataan ilmastonmuutokseen liittyvistä artikkeleista kootulla korpuksella. Tässä sovelluksessa käytetään NLP-sovellusta, joka tokenisoi kaikki sanat tekstissä ja muokkaa samalla kaikki isot kirjaimet pieniksi, jonka jälkeen se laskee sanojen esiintymismäärät kaikissa teksteissä. Sitten tieto esiintymismääristä muokataan taulukkomuotoon, johon tallennetaan tieto, kuinka monessa tekstissä sana esiintyy ja kuinka monta kertaa kussakin tekstissä. Näin kaikki relevantti tieto on valmiiksi laskettavassa muodossa. tf-, idf- ja tf-idf-arvot lasketaan kaikki tämän NLP-koneen tuloksesta aiemmin esitetyllä tavalla. Se mikä erottaa tämän sovelluksen aiemmista, on NLP-kone itse, sillä perinteiset korpusohjelmistot keräävät eivät kerää kaikkea tietoa samaan aikaan vaan ne 'hyppivät' edestakaisin datassa, jonka lisäksi ne vaativat enemmän aktiivista toimintaa käyttäjiltä verrattuna NLP-koneeseen, joka tekee kaikin itse automaattisesti. Verrattuna perinteiseen menetelmään, jossa käytettiin LLT (engl. *Log-Likelihood Test*) -algoritmia, NLP-konetta käyttänyt sovellus toimi paremmin yleisten sanojen eliminoinnissa, piirteiden valinnan joustavuudessa sekä adaptiivisuudessa suhteessa vaihtoehtoisiin tutkimustavoitteisiin. Tämä sovellus on hyvä esimerkki siitä, miten muuttamatta aiemmin esitettyjä määritelmiä tf-idf-arvon laskemiselle, voidaan tekstinlouhintaa sillä kuitenkin tehostaa hyödyntämällä parempia kieliteknologian malleja ennen kuin menetelmä itse laskee arvoja ja tuloksia.

Alva Principe, Chiarini ja Viviani [24] esittävät tf-idf-sovelluksen, joka soveltaa semanttisen konstekstin huomioon ottamista käyttämällä piileviä kontekseteja (*engl. latent concepts*). Tätä sovellusta kutsutaan termillä lcf-idf (engl. *latent concept frequency - idf*) ja sitä sovelletaan pitkien dokumenttien luokittelutehtävässä. Tavoitteena on yhdistää kaksisuuntaisiin muuntajiin (engl. *transformer*) perustuvan BERT-kielimallin kontekstin ymmärtämisen kyvyn ja tf-idf-menetelmän

todetun tehokkuuden. Ensimmäinen vaihe sovelluksessa on esikoulutetun kielimallin (BERT:in) käyttö dokumenttien tekstin tokenisointiin niin, että jokaiselle sanalle saadaan asiayhteysvektori (engl. *contextualized word embedding*), joten samalle sanalle voi olla eri asiayhteysvektori riippuen kontekstista. Moniulotteiset asiayhteysvektorit redusoidaan ja näihin yksinkertaisimpiin vektoreihin sovelletaan kuvausfunktiota, joilla saadaan piileviä konteksteja. Toisessa vaiheessa jokainen sana-token-dokumentti käännettään uudeksi piilevien kontekstien dokumentiksi. Kolmannessa vaiheessa varsinaisesti rakennetaan dokumentin kuvaus käyttäen lcf-idf:ää. Käytännössä sen sijaan, että laskettaisiin sanan yleisyys ja verrattaisiin sitä käänteiseen yleisyyteen korpuksessa, lasketaan piilevän kontekstin yleisyys dokumentissa verrattuna koko korpukseen. Testaustulosten perusteella lcf-idf toimii selkeästi paremmin pitkien dokumenttien luokittelussa kuin perinteinen tf-idf, sekä pääasiallisesti paremmin kuin BERT-kielimallit. Tämä sovellus osoittaa, miten muuntajiin perustuvien kielimallien yhdistämisen muokattuun tf-idf-menetelmään voi antaa merkittävästi parempia tuloksia, varsinkin pitkien ja yhtenäisten tekstien tapauksissa. Toisaalta sovelluksen prosessin kolmas vaihe osoittaa, että pohjimmiltaan perinteisen tf-idf-menetelmän periaatteita noudattava sovellus voi parantaa tuloksia ilman suuria muutoksia itse prosessiin, vaan muutokset voi kohdistaa lähinnä esiprosessointiin.

Tf-idf on siis hyödyllinen menetelmä tekstinlouhinnassa. Kuten edellä esitetyistä esimerkkisovelluksista näkee, on menetelmän käytölle soveltuvat tehtävät keskenään kovinkin erilaisia. Sitä voidaan käyttää niin avainsanojen poimintaan, dokumenttien luokitteluun kuin syvempään analyysiin kirjoittajasta. Gebre, Zampieri, Wittenburg et al. [22] esittävät miten syvempää analyysia voi tehdä ilman valtavia laajennuksia valmiiseen menetelmään. Chen [23] taas esitää miten yhdistämällä 'perinteinen' tf-idf-menetelmä uutta kieliteknologiaa hyödyntävään esiprosessointiin, voidaan siitä saada merkittävästikin lisäarvoa. Alva Principe, Chiarini ja Viviani [24] puolestaan esittävät miten muuntajakielimallin sekä muokatun tf-idf:n avulla voidaan viedä näi-

tä yhdistämällä tekstinlouhintaa eteenpäin merkittävästi. Nämä konkreettiset esimerkit osoittavat, miten tf-idf soveltuu erilaisiin tehtäviin ja miten koneoppiminen, ja kielimallit ovat vieneet sen käyttöä eteenpäin ja parantaneet tuloksia, mutta myös miten verrattain yksinkertainen pohjamenetelmä on toimiva.

4.3 Named-Entity Recognition

Named-Entity Recognition eli NER on siis nimettyjen ja tavoitteen kannalta merkityksellisten entiteettien tunnistamista tekstistä. Nämä entiteetit voivat olla mitä tahansa tarkoitukseen sopivia asioita, kuten esimerkiksi henkilöitä, paikkoja, lääketieteellisiä operaatioita tai proteiineja. NER:iä ja sen sovelluksia käytetään pitkälti tiedon poiminnassa sekä esimerkiksi kysymyksiin vastaamisessa ja aihemallinnuksessa. Kuten muitakin tekstinlouhinnassa käytettäviä menetelmiä, on kone- ja syväoppiminen vienyt NER:iä merkittävästi eteenpäin ja vähentänyt tarvetta alan asiantuntijuudelle, vaikka menetelmän perusperiaatteet ovat samanlaiset. [25]

NER-menetelmät jaetaan siis kahteen osaan: perinteisiin (sääntöpohjaisiin) sekä koneoppiviin. Perinteisissä menetelmissä yleinen haaste on tarve alan asiantuntijuudelle alakohtaisen tai vielä tarkemmin rajatun sanakirjan muodostamiselle. Lisäksi haastetta voi aiheuttaa alojen monikielinen tai synonyymejä sisältävä sanasto. [15] Haasteena kone- ja syväoppivissa menetelmissä on kuitenkin käsin merkityn datan saaminen tarvituissa määrissä, sekä testaamista varten saatavan ei-merkityn datan keräys. Lisäksi haasteena voi olla miten resursseja on saatavilla entiteetin kontekstien tarkasteluun (esimerkiksi kuinka monta sanaa ennen ja jälkeen) sekä minkälainen luokittelu on sopiva. [12]

Seuraavaksi tarkastellaan alan kirjallisuuden perusteella joitain sovelluksia ja sovelluskohteita NER:ille sekä tarkastellaan käytön tuloksia kirjallisuuden perustella. Lisäksi näistä esimerkeistä NER:n käytöstä tekstinlouhinnassa tehdään havaintoja ja huomioita.

Katona ja Farkas [26] esittävät NER-sovelluksen kliinisten tekstien analyysiin, joka louhii teksteistä mainintoja vaivoista potilaiden kotiuttamisteksteistä. Ensimmäisenä tekstit normalisoidaan poistamalla merkit, kuten pisteet, pilkut, kysymysmerkit jne. Toisena luodaan uusi sanakirja käyttäen UMLS-sanakirjaa, joka sisältää lääketieteellisiä vaivoja, sekä MetaMap-merkintätyökaluja, joka osaa tunnistaa UMLS:n sanoja tekstistä. Kolmantena käytetään aiempaa NER-sovellusta, Illinois NER:iä, joka sisältää valmiita tapoja entiteetin kontekstin analysointiin. Käyttäen tätä NER-sovellusta, analysoidaan jokaisen merkityn entiteetin ympäristö neljän muun läheisimmän merkityn entiteetin avulla, jolloin saadaan tietoa tekstin piirteistä. Sovelluksen tulokset ovat pääsääntöisesti hieman paremmat tai yhtä hyvät kuin yksinkertaisemmat ja vaihtoehtoiset sovellukset. Sovelluksen ongelmia ovat muun muassa sanojen monitulkintaisuus, tuntemattomat vaivat sekä epäkoherentit lauseet, mitkä kaikki heikensivät testauksen arvoja. Tämä sovellus osoittaa miten valmiiden sanakirjojen soveltava käyttö on toimivaa eikä omaa tarvitse aina tehdä, sekä miten aiempi NER-sovellus voi olla myös käyttötarkoitukseen sopiva. Toisaalta sovellus osoittaa minkälaisia ongelmia NER-menetelmissä voi esiintyä, jos esimerkiksi sanakirja ei ole täydellinen tai teksti ei ole oletetun laatuista.

Kim, Lee, So et al. [27] esittävät BERN-nimisen NER-sovelluksen biolääketieteeseen, joka hyödyntää muuntajiin perustuvaa BioBERT-mallia, joka on aiempi BERT-kielimallia hyödyntävä NER-sovellus, jolla on kyky tunnistaa ja löytää uusia entiteettejä. BERN:in tarkoituksena on yrittää ratkaista vajaiden sanakirjojen ja sanojen monitulkinnaisuuden ongelmia. BioBERT käy läpi tekstin ja laskee kaikelle tekstin sisällölle annettavien merkkien todennäköisyydet, joiden perusteella BioBERT pyrkii määrittelemään nimetyt entiteetit tekstistä. Tämä johtaa käytännössä entiteettien päällekkäisyyksiin esimerkiksi sanojen monitulkintaisuuden tai taivutusmuotojen takia. Tämän ongelman ratkaisemiseksi BERN hyödyntää ennalta määrättyjä sääntöjä. Ensin päätetään entiteettien päällekkäisyyden suhteellisen määrän

perusteella, että onko tarvetta analysoida päällekkäisten ryhmää tarkemmin, ja jos ei ole, niin kaikki entiteetit merkitään. Jos taas tarvetta on, analysoidaan päällekkäisyyksiä entiteettien mutaatioiden välillä, ja jos sitä esiintyy, merkitään sekä mutaatiot että mutatoitumattomat entiteetit. Jos sitä taas ei esiinny, merkitään vain ei-mutatoitumattomat entiteetit. Tällä päätösprosessilla saadaan 'jalostettua' BioBERT:in tuloksia ja tunnistettua sekä poimittua huomattavasti tarkemmin nimettyjä entiteettejä tekstistä. Käytön tulokset osoittavat, että BERN toimii suhteellisen hyvin biolääketieteellisessä tekstinlouhinnassa, tosin joidenkin tyyppisten entiteettien tunnistus toimi paremmin kuin toisten, sillä esimerkiksi taudit eroteltiin ja tunnistettiin paremmin kuin geenit ja proteiinit. Tämä sovellus osoittaa, miten kehittyneemmät kielimallit voivat viedä NER:n soveltamista merkittävästi eteenpäin, mutta myös mitä ongelmia tuloksissa voi olla (päällekkäisyys), sekä miten niitä voida suhteellisen yksinkertaisilla manuaalisesti tuotetuilla säännöillä kuitenkin ratkaista ja varmistaa tulosten laatu.

Huang, He, Yang et al. [28] esittävät materiaalitieteisiin tarkoitetun NER-sovelluksen, jossa perinteinen sekvenssileimaus, missä tarkastellaan sekvenssejä eli kontekseja missä entiteetit esiintyvät ja niiden perusteella tunnistetaan niitä, korvataan koneellisella luetunymmärtämisellä (engl. *machine reading comprehension*) jotta ratkaistaan esimerkiksi entiteettien monitulkintaisuuden ja semanttisten yhteyksien aiheuttamia haasteita perinteisille koneoppiville menetelmille. Sovelluksen 'selkärankana' käytetään BERT- ja MatsciBERT-kielimalleja. Yksinkertaistetusti, koneellista luetunymmärtämistä varten merkatun NER-koulutusdatan perusteella luodaan entiteettikohtaisesti hakulause perustuen sanoihin ja tekstinpätkiin, jotka liittyvät kyseiseen entiteettiin. Tässä sovelluksessa apuna käytettiin manuaalisen annotoinnin tekijöille tehtyjä ohjeita datan annotointiin. Käyttäen muuntajakielimalleja, arvioidaan hakulauseille alku- ja loppupisteet, jotka voivat olla päällekkäisiä, jonka jälkeen arvioidaan hakulausekkeiden sisällä nimettyjen entiteettien sijainnit. Testauksessa

sovellus osoittautuu aiempia sovelluksia paremmaksi, tosin tarkkuusprosentit eivät parane valtavia määriä. Tämä sovellus osoittaa vaihtoehtoisten kontekstitutkimisen tapojen toimivuuden NER:issä kun käytetään muuntajakielimalleja. Testauksen perusteella tämä sovellus pystyi varsinkin ratkaisemaan monitulkintaisuuden ongelmia verrattuna aiempiin sovelluksiin. Toisaalta malli on materiaalitieteisiin suunnattu ja ei toimi muilla aloilla. Lisäksi huomattava on myös tarve suhteellisen suuresta määrästä annotointiohjeita, ei vain annotoitua dataa.

NER-menetelmän sovelluksille on siis keskenään hyvinkin erilaisia käyttökohteita. Oikein käytettynä, on NER hyvin hyödyllinen menetelmä tekstinlouhinnassa ja tarjoaa uusia mahdollisuuksia soveltaa viime aikoina kehitettyjä tekniikoita. Katona & Farkas esittävät suhteellisen yksinkertaisen ja valmiisiin ratkaisuihin perustuvan toimivan NER-sovelluksen, joka kuitenkin myös näyttää yleisiä ongelmia. Kim, Lee, So et al. [27] esittävät koneoppivan NER-sovelluksen, jolla pyritään ratkaisemaan yksinkertaisella jaottelulla niitä ongelmia, mitä muun muassa Katona ja Farkas [26] sovelluksessa ilmeni. Huang, He, Yang et al. [28] esittävät erilaista lähestymistapaa käyttävän sovelluksen, joka pyrkii hyödyntämään viimeisintä kieliteknologian kehitystä parantamaan selkeästi tuloksia aiempiin verrattuna, tosin suurella esikoulutuksen kustannuksella. Nämä konkreettiset esimerkit osoittavat joitain ongelmia, mitä NER-sovelluksissa tekstinlouhinnassa esiintyy yleisesti, mutta myös ratkaisuja niihin ongelmiin, käyttäen sekä viime aikoina yleistynyttä teknologiaa sekä yksinkertaisempia tapoja.

5 Yhteenveto

Tässä työssä tutkittiin, minkälaisia yleisiä menetelmiä tekstinlouhinnassa käytetään, sekä miten nämä menetelmät toimivat tekstinlouhinnassa.

TK1 oli selvittää kolme alan aineistoon tehtävän katsauksen perusteella yleistä tekstinlouhinnassa käytettyä menetelmää. Tutkitun aineiston perusteella nämä olivat hierarkkinen klusterointi, tiedon haku tf-idf:ää käyttäen sekä Named-Entity Recognition (NER). Näitä menetelmiä esiintyi sellaisenaan ja sovellettuna useissa sadoissa läpikäydyissä alan artikkeleissa, jotka myös viittasivat aiempiin menetelmien sovelluksiin. Tällä perusteella niitä pidettiin yleisinä. TK2 oli selvittää miten nämä ensimmäisen tutkimuskysymyksen perusteella valitut menetelmät toimivat tarkemmin tekstinlouhinnassa. Tämä tehtiin käymällä läpi menetelmien toimintaa tarkemmin läpi sekä ottamalla menetelmistä tehtyjä sovelluksia alan kirjallisuudesta tarkasteluun ja arviointiin. Näiden sovellusten perusteella menetelmien todettiin toimivan hyvin tekstinlouhinnassa, jos käyttökohde oli valittu oikein, pohjamenetelmää osattiin hyödyntää ja käytettiin uudempia tekniikoita kuten muuntajakielimalleja. Menetelmät kärsivät kaikki myös joistain niille tyypillisistä ongelmista, jota sovellukset yrittivät kuitenkin ratkoa.

Jatkotutkimuksena voitaisiin tutkia alan aineistoa ja kirjallisuutta vielä laajemmin, jotta saadaan laajempi ja tarkempi kuva kaikkien menetelmien yleisyydestä. Lisäksi alan nopean kehityksen takia, olisi hyvä tutkia uusimpia sovelluksia ja miten ne vertautuvat toisiinsa käytännön testeissä.

Lähdeluettelo

[1] D. Antons, E. Grünwald, P. Gichy ja T. O. Salge, "The application of text mining methods in innovation research: current state, evolution patterns, and development priorities", *R&D Management*, vol. 50, nro 3, 2020. DOI: 10.1111/radm.12408.

[2] H. Liang, X. Sun, Y. Sun ja Y. Gao, "Text feature extraction based on deep learning: a review", *EURASIP Journal on Wireless Communications and Networking*, 2017. DOI: 10.1186/s13638-017-0993-1.

[3] M. Allahyari, S. Pouriyeh, M. Assefi et al., "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques", 2017. DOI: 10.48550/arXiv.1707.02919.

[4] H. Ahonen, O. Heinonen, M. Klemettinen ja A. I. Verkamo, "Applying Data Mining Techniques in Text Analysis", *Tech. rep. C-1997-23*, 1997.

[5] X. Rui ja D. C. Wunsch, "Survey of Clustering Algorithms", *IEEE Transactions on Neural Networks*, vol. 16, nro 3, s. 645–678, 2005. DOI: 10.1109/TNN.2005.845141.

[6] S. Patel, S. Sihmar ja A. Jatain, "A Study of Hierarchical Clustering Algorithms", *2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, s. 537–541, 2015.

[7] K. A. A. Hambrade ja H. Proenca, "Information Retrieval: Recent Advances and Beyond", *IEEE Access*, vol. 11, s. 76 581–76 604, 2023. DOI: 10.1109/ACCESS.2023.3295776.

[8] Y. Zhang, J. Rong ja Z. Zhi-Hua, "Understanding bag-of-words model: a statistical framework", *International Journal of Machine Learning and Cybernetics*, vol. 1, s. 43–52, 2010. DOI: 10.1007/s13042-010-0001-0.

[9] P. Cichosz, "Bag of Words and Embedding Text Representation Methods for Medical Article Classification", *International Journal of Applied Mathematics and Computer Science*, vol. 33, nro 4, s. 603–621, 2023. DOI: 10.34768/amcs-2023-0043.

[10] A. Jalilifard, V. Fernandes Caridá, A. Fernandes Mansano, R. S. Cristo ja F. Penhorate Carvalho da Fonseca, "Semantic Sensitive TF-IDF to Determine Word Relevance in Documents", DOI: 10.48550/arXiv.2001.09896.

[11] Z. Xu, M. Chen, F. Sha ja K. Q. Weinberger, "An alternative text representation to TF-IDF and Bag-of-Words", DOI: 10.48550/arXiv.1301.6770.

[12] Z. Hong, L. Ward, K. Chard, B. Blaiszik ja I. Foster, "Challenges and Advances in Information Extraction from Scientific Literature: a Review", *JOM*, vol. 73, s. 3383–3400, 11 2021. DOI: 10.1007/s11837-021-04902-9.

[13] M. Wankhade, A. C. Sekhara Rao ja C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges", *Artificial Intelligence Review*, vol. 55, s. 5731–5780, 2022. DOI: 10.1007/s11837-021-04902-9.

[14] Z. Nasars, S. W. Jaffry ja M. K. Malik, "Named Entity Recognition and Relation Extraction: State-of-the-Art", *ACM Computing Surveys*, vol. 54, 1 2021. DOI: 10.1145/3445965.

[15] S. Gao, O. Kotevska, A. Sorokine ja J. B. Christian, "A pre-training and self-training approach for biomedical named entity recognition", *Plos One*, vol. 16, 2 2021. DOI: 10.1371/journal.pone.0246310.

[16] M. Fionn ja P. Contreras, "Algorithms for hierarchical clustering: An overview", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2012. DOI: 10.1002/widm.53.

[17] G. Chrupala, "Hierarchical clustering of word class distributions", *NAACL-HLT Workshop on the Induction of Linguistic Structure*, s. 100–104, 2012.

[18] B. Can ja S. Manandhar, "An Agglomerative Hierarchical Clustering Algorithm for Labelling Morphs", *Proceedings of Recent Advances in Natural Language Processing*, s. 129–135, 2013.

[19] Y. Chali ja S. Noureddine, "Document Clustering with Grouping and Chaining Algorithms", *Second International Joint Conference on Natural Language Processing*, s. 280–291, 2005. DOI: 10.1007/11562214_25.

[20] C.-L. Chen, F. S. Tseng ja T. Laing, "Mining fuzzy frequent itemsets for hierarchical document clustering", *Information Processing and Management*, vol. 46, s. 193–211, 2 2010. DOI: 10.1016/j.ipm.2009.09.009.

[21] Y. Wang, "Research on the TF–IDF algorithm combined with semantics for automatic extraction of keywords from network news texts", *Journal of Intelligent Systems*, vol. 33, 1 2024. DOI: 10.1515/jisys-2023-0300.

[22] B. G. Gebre, M. Zampieri, P. Wittenburg ja T. Heskes, "Improving Native Language Identification with TF-IDF Weighting", *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, s. 216–223, 2013.

[23] L.-C. Chen, ”An extended TF-IDF method for improving keyword extraction in traditional corpus-based research: An example of a climate change corpus”, *Data & Knowledge Engineering*, vol. 153, 2024. DOI: 10.1016/j.datak.2024.102322.

[24] R. A. Alva Principe, N. Chiarini ja M. Viviani, ”An LCF-IDF Document Representation Model Applied to Long Document Classification”, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, s. 1129–1135, 2024.

[25] V. Yadav ja S. Bethard, ”A Survey on Recent Advances in Named Entity Recognition from Deep Learning models”, 2019. DOI: 10.48550/arXiv.1910.11470.

[26] M. Katona ja R. Farkas, ”SZTE-NLP: Clinical Text Analysis with Named Entity Recognition”, *Proceedings of the 8th International Workshop on Semantic Evaluation*, s. 615–618, 2014. DOI: 10.3115/v1/S14-2108.

[27] D. Kim, J. Lee, C. H. So et al., ”A Neural Named Entity Recognition and Multi-Type Normalization Tool for Biomedical Text Mining”, *IEEE Access*, vol. 7, s. 73 729–73 740, 2019. DOI: 10.1109/ACCESS.2019.2920708.

[28] Z. Huang, L. He, Y. Yang et al., ”Application of machine reading comprehension techniques for named entity recognition in materials science”, *Journal of Cheminformatics*, vol. 16, 1 2024. DOI: 10.1186/s13321-024-00874-5.