Opinnäytetyö (AMK)

Tietojenkäsittely

2020

Emil Nuutinen

LUONNOLLISEN KIELEN KÄSITTELY: SEMANTTISEN SAMANKALTAISUUDEN MITTAAMINEN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely

2020 | 38 sivua

Emil Nuutinen

LUONNOLLISEN KIELEN KÄSITTELY: SEMANTTISEN SAMANKALTAISUUDEN MITTAAMINEN

Opinnäytetyön tavoitteena oli tehdä toimeksiantajalle soveltava tutkimus, jossa tutkittiin kielimallien eroja tarkkuudessa ja käytettävyydessä semanttisen samankaltaisuuden mittaamisessa. Tutkimuksen tuloksia oli tarkoitus hyödyntää tulevia projekteja suunnitellessa.

Opinnäytetyössä tutkittiin kahden eri soveltavan tutkimuksen pohjalta viiden eri kielimallin toimintaa sekä käytettävyyttä semanttisen samankaltaisuuden mittaamisessa. Yhdessä toimeksiantajan kanssa tutkimuksen kohteeksi valittiin TF-IDF, fastText, LASER, Sentence-BERT sekä USE kielimallit.

Ensimmäinen tutkimus tehtiin STS Benchmarkin lausepareilla SemEval-2017 STS tehtävän kriteerein, mutta tutkimuksen tulokset eivät olleet luotettavia. Toiseen tutkimukseen luotiin omat testilausekkeet, joiden tarkoitus oli tuoda esille kielimallien välisiä eroja kielen ymmärtämisessä.

Vaikka USEn DAN-versio osoittautui tämän tutkimuksen testeissä parhaaksi kompromissiksi laskenta-ajan ja tarkkuuden perusteella, niin käytettävissä olevalla koulutusdatalla on kuitenkin paljon suurempi vaikutus kuin valitulla kielimallilla.

ASIASANAT:

luonnollisen kielen käsittely, semanttinen samankaltaisuus, tf-idf, fastText, laser, bert, use

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology

2020 | 38 pages

Emil Nuutinen

NATURAL LANGUAGE PROCESSING: MEASURING SEMANTIC SIMILARITY

The aim of the thesis was to conduct an applied research for the client, in which the differences in language models in terms of accuracy and usability in measuring semantic similarity were investigated. The results of the study were to be used in planning on future projects.

Based on two different applied studies, the thesis examined the operation of five different language models and their usability for measuring semantic similarity. Together with the client, TF-IDF, fastText, LASER, Sentence-BERT and USE language models were selected for the study.

The first study was conducted with the STS Benchmark sentence pairs according to the SemEval-2017 STS task criteria, but the results of the study were not reliable. For the second study, separate test sentences were created to highlight the differences in language comprehension between the language models.

Although the DAN version of USE proved to be the best compromise in terms of computation time and accuracy in the tests of this study, the available training data still has a much greater impact than the chosen language model.

KEYWORDS:

natural language processing, semantic similarity, tf-idf, fasttext, laser, bert, use

SISÄLTÖ

KAAVAT

KUVAT

KOODIT

TAULUKOT

KÄYTETYT LYHENTEET TAI SANASTO

Anaconda	tieteelliseen ohjelmointiin luotu ohjelmointiympäristö (Anaconda Invidual Edition 2020)
BiLSTM	kaksisuuntainen LSTM, Bi-Directional Long Short Term Memory (Papers With Code 2020b)
BoW	malli, jossa teksti mielletään sanoja sisältäväksi pussiksi, Bag of Words
CBOW	jatkuva Bag of Words malli, Continuous-Bag-Of-Words (Mikolov ym. 2013)
DAN	neuroverkko, joka käyttää sanavektoreiden keskiarvoja ja syöttää ne lineaaristen kerrosten läpi, Deep Averaging Network
Jupyter Notebook	interaktiivinen muistikirja, joka voi sisältää tekstiä, koodia, yhtälöitä ja visualisaatioita (Jupyter Notebook 2020)
korpus	kokoelma, joukko tekstejä tai dokumentteja
LSTM	RNN, joka pyrkii korjaamaan häviävän gradientin ongelmaa kahdella muistissa olevalla tilalla, Long Short Term Memory (Papers With Code 2020a)
n-grammi	n-merkin pituinen peräkkäinen jatkuva jakso tekstissä
RNN	takaisinkytketty neuroverkko, Recurrent Neural Network (Recurrent Neural Network 2020)
skip-grammi	n-grammi, joka voi sisältää tyhjiä merkkejä jakson sisällä (Mikolov ym. 2013)
STS Benchmark	semanttista samankaltaisuutta mittaava tutkimus, Semantic Textual Similarity Benchmark (STS Benchmark 2020)
TF-IDF	malli, joka kuvaa termin ilmestymistä dokumentissa suhteessa termin ilmestymiseen koko korpuksessa, Term Frequency-Inverse Document Frequency (scikit-learn 2020b)

1 JOHDANTO

Luonnollisen kielen käsittely on kokenut suuren harppauksen 2010-luvun aikana. Laskentatehon kasvun ja suurien datamäärien myötä uudet Attention- ja Transformer-arkkitehtuuriin perustuvat mallit ovat kasvattaneet kielimallien tarkkuutta ja ymmärrystä uudelle tasolle. Mallien kasvaessa toisaalta myös mallin käyttäminen vaatii enemmän laskentatehoa ja muistia. Kielimallin rakenteen monimutkaistuessa mallin antamien tulosten arviointi ja testaaminen myös vaikeutuu.

Tämän opinnäytetyön tavoitteena oli tehdä toimeksiantajalle soveltava tutkimus, jossa tutkitaan kielimallien eroja tarkkuudessa ja käytettävyydessä semanttisen samankaltaisuuden mittaamisessa. Tutkimuksen tuloksia on tarkoitus hyödyntää tulevissa projekteissa, täten tutkimusstrategiaksi valikoitui toimintatutkimus.

Opinnäytetyön luvussa 2 määritellään mitä semanttisen samankaltaisuuden mittaamisella tarkoitetaan. Luvussa 3 tutustutaan viiden yhdessä toimeksiantajan kanssa valitun mallin toimintaan semanttisen samankaltaisuuden näkökulmasta. Malleiksi valittiin TF-IDF, fastText, LASER, BERT (Sentence-BERT) ja USE. Teoriaosuudessa perehdytään eri mallien rakenteeseen sekä heikkouksiin ja vahvuuksiin. Teoriaosuus myös pohjustaa työssä tehdyn soveltavan tutkimuksen.

Lukuun 4 on koottu soveltavan tutkimuksen kulkua, käytettyjä tutkimusmetodeja ja niistä saatuja tuloksia. Soveltavan tutkimuksen ensimmäisessä osassa vertaillaan mallien tarkkuutta yleisesti käytetyillä STS Benchmarkin lausepareilla. Soveltavan tutkimuksen toisessa osassa malleja testataan itse suunnitelluilla testilausekkeilla, joiden tarkoitus on haastaa mallien semanttista ymmärrystä sekä auttaa selvittämään mallien vahvuuksia ja heikkouksia.

Kaikki testit on toteutettu Jupyter Notebook-tiedostoissa Python-ohjelmointikielellä. Kaikki testattavat lauseet ovat englanninkielisiä.

2 SEMANTTINEN SAMANKALTAISUUS

Semanttisella samankaltaisuuden mittaamisella vertaillaan dokumenttien, lauseiden tai sanojen välistä etäisyyttä niiden merkityksen tai semantiikan perusteella. Semanttisen samankaltaisuuden mittaus koostuu kahdesta pääelementistä, sana- ja lausevektoreista sekä kosinietäisyydestä. Kielimallien tehtävä on luoda semanttisesti merkittäviä sana- ja lausevektoreita, joiden samankaltaisuutta kyetään mittaamaan kosinisamankaltaisuudella. (Maind ym. 2012)

2.1 Sana- ja lausevektorit

Sanavektorit ovat luonnollisen kielen käsittelyn metodi esittää sanat numeraalisina vektoreina. Sanojen muuttaminen numeraalisiksi vektoreiksi sallii kielimallien laskea niiden semanttisuutta tarkemmin ja tehokkaammin. Monet kielimallit myös luovat lausevektoreita laskemalla lauseen sisällä olevien sanavektoreiden normeeratun keskiarvon. (Mikolov ym. 2013)

Jokaiselle sanalle luodaan sitä edustava vektori. Kielimallin rakenne määrää, miten tämä vektori luodaan. Tämän vuoksi eri kielimallien väleillä voi olla suuriakin eroja sanavektoreiden koossa ja tarkkuudessa. Sanavektorit tallentavat sanan merkitystä myös sen käyttöympäristön perusteella toisin kuin esimerkiksi yksinkertaisempi Bag of Words-metodi.

Yksi sanavektoreiden heikkouksista on se, että monimerkityksiset ja homonyymiset sanat menettävät merkitystään, koska sanat esitetään vain yhdellä vektorilla. Uudemmat kielimallit kuitenkin upottavat vektoreihin enemmän tietoa ja sen vuoksi säilyttävät enemmän semanttista tietoa sanoista kuin yksinkertaisemmat kielimallit.

2.2 Kosinisamankaltaisuus

Yleisin tapa mitata kahden sanan semanttista samankaltaisuutta on mitata niiden sanavektoreiden kosinisamankaltaisuutta.

Kosinisamankaltaisuus saadaan laskemalla kahden vektorin kosinikulma. Jos kosinikulma on 0 astetta eli vektorit ovat vaakasuorassa toisiinsa nähden niin

samankaltaisuusarvo on 1 ja jos kosinikulma on 90 astetta eli vektorit ovat kohtisuorassa toisiinsa nähden niin samankaltaisuusarvo on 0. (Kaava 1.)

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \, \sqrt{\sum_1^n b_i^2}}$$

Kaava 1. Kosinisamankaltaisuuden kaava. (Cosine Similarity 2020)

Cosθ siis saadaan laskemalla vektoreiden **a** ja **b** pistetulo ja jakamalla se niiden L^2 normeeratulla pistetulolla. (Kaava 1.)

Toisinaan semanttisen samankaltaisuuden mittaamisessa käytetään myös kosinietäisyyttä, joka saadaan vähentämällä kosinisamankaltaisuus yhdestä eli 1 – kosinisamankaltaisuus. Esimerkiksi Scipy-ohjelmointikirjasto käyttää tätä versiota. (Cosine distance 2020)

3 TESTATUT KIELIMALLIT

Tässä osiossa käydään läpi testattujen kielimallien toimintatapoja, jotta voimme ymmärtää niiden antamien tuloksien eroja soveltavassa tutkimuksessa.

3.1 TF-IDF (Term Frequency-Inverse Document Frequency)

Ensimmäinen läpikäytävistä malleista on TF-IDF.

TF-IDF on statistinen metodi, jonka tavoite on laskea termin tärkeysarvo korpuksessa.TF-IDF arvo kasvaa verrannollisesti sen perusteella, kuinka monta kertaa termi esiintyy lauseessa tai dokumentissa ja kompensoituu sen perusteella, kuinka monta kertaa termi esiintyy koko korpuksessa. TF-IDF koostuu vain kahdesta osasta, TF eli Term Frequency ja IDF eli Inverse Document Frequency.

Term Frequency mittaa kuinka usein sana tai termi toistuu dokumentissa, toisin sanoen sillä mitataan sanan tai termin frekvenssiä.

TF saadaan jakamalla termin ilmaantumislukumäärä dokumentissa dokumentin kokonaistermimäärällä.

Toisin sanoen TF mittaa sanan ilmaantumistiheyttä dokumentissa.

IDF mittaa termin merkitystä dokumentissa/korpuksessa. (Kaava 2.)

$$\mathrm{idf}(t) = \log \frac{1+n}{1+\mathrm{df}(t)} + 1$$

Kaava 2. IDF. (scikit-learn 2020b)

IDF-kaavassa t on termi, n on dokumenttien määrä ja df(t) kuvastaa dokumenttien määrää, jotka sisältävät termin t. Yhden lisääminen osoittajaan ja nimittäkään poistaa mahdollisuuden nollalla jakamiseen ja täten tasoittaa IDF-n liikettä. (Kaava 2.)

Edelliset yhdistämällä saadaan aikaiseksi TF-IDF, joka ottaa huomioon sekä termin ilmaantuvuuden dokumentissa, että sen ilmaantuvuuden koko korpuksessa. (Kaava 3.)

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t)$$

Kaava 3. TF-IDF. (scikit-learn 2020b)

Kuten kaavassa 2, termiä kuvataan kirjaimella t. Dokumentin kokonaistermimäärää kuvataan kirjaimella d.

Jos termi esiintyy usein dokumentissa, mutta harvoin koko korpuksessa niin TF-IDF saa suuremman arvon kuin jos termi esiintyy usein dokumentissa ja usein koko korpuksessa.

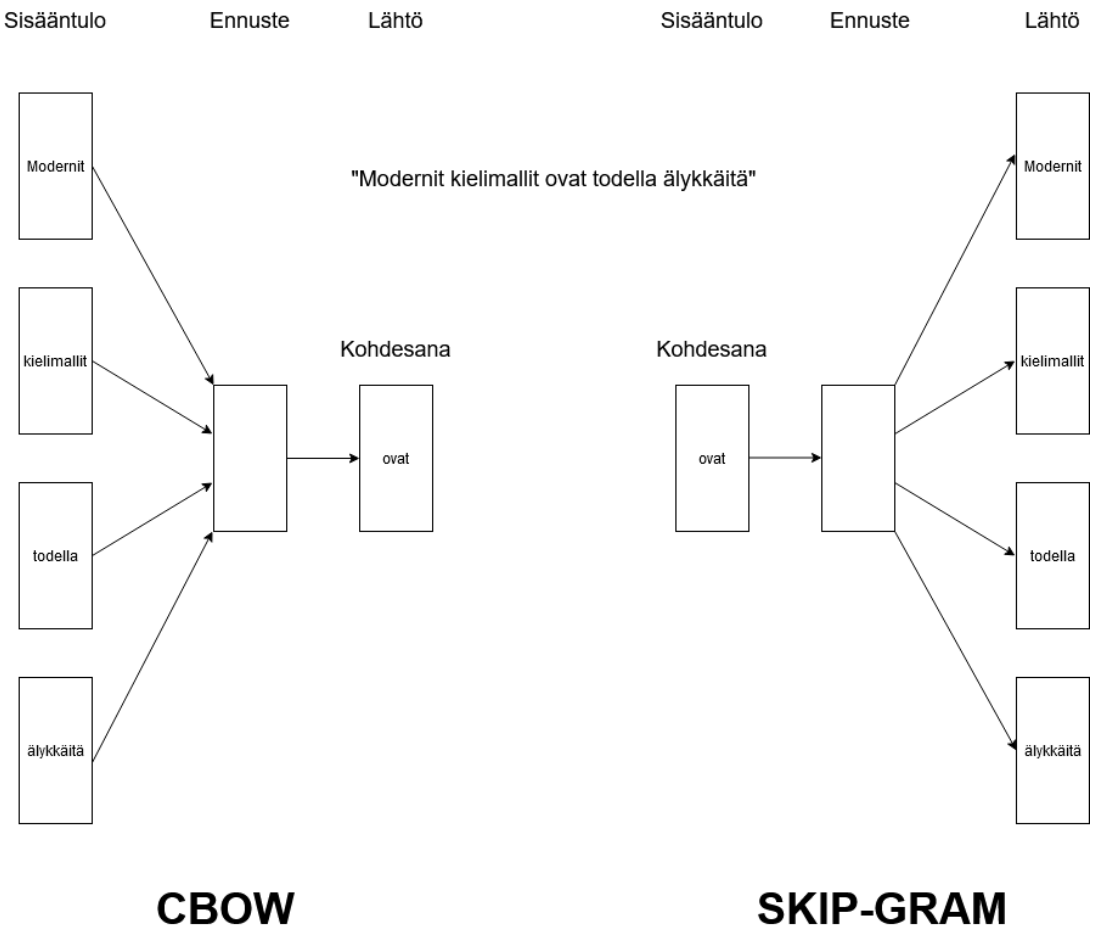
Toisin sanoen suuri TF-IDF tarkoittaa, että sanan merkitys kyseisessä dokumentissa on suuri.

3.2 fastText

fastText on Facebook Research -tiimin kehittämä lähinnä tekstin klassifiointiin suunniteltu monikielinen kirjasto (fastText 2020). Se pohjautuu perinteiseen word2vec-malliin (Mikolov ym. 2013). fastText laskee sanarepresentaatiot joko skipgrammi-mallilla tai CBOW-mallilla (continuous-bag-of-words). fastText on suunniteltu ajettavaksi vain prosessorilla eikä näytönohjaimella kuten muut modernit syväoppimismallit. fastTextin filosofia on tuoda huippuluokan suorituskykyä marginaalisella teholla syväoppimismalleihin verrattuna.

Skipgrammi-malli pyrkii ennustamaan ympäröivät sanat kohdesanan avulla. CBOW-malli pyrkii ennustamaan kohdesanan ympäröivien sanojen perusteella (Kuva 1).

Ennustamiseen fastText käyttää joko softmax-funktiota tai hierarkiallista softmax-funktiota. Normaalia softmax-funktiota käytetään pienempien korpuksien kanssa, mutta hierarkiallisella softmax-funktiolla saavutetaan parempi suorituskyky isojen korpuksien kanssa.



Kuva 1. CBOW ja SKIPGRAM. (Mikolov ym. 2013)

Sanavektoria luodessa fastText luo sanoille sanatason n-grammin lisäksi kirjaintason n-grammien representaation. Esimerkiksi sanan "jouluaatto" representaatio n-grammin pituuden ollessa 3 on "jouluaatto, jou, oul, ulu, lua, uaa, aat, att, tto". Tämän vuoksi fastText kykenee luomaan sanavektoreita myös sanoille, joita ei ole koulutukseen käytetyssä korpuksessa.

fastText myös luo sanavektoreista normeerattuja lausevektoreita. Implementaation rakenteesta johtuen lausevektoreissa ei oteta huomioon sanajärjestystä.

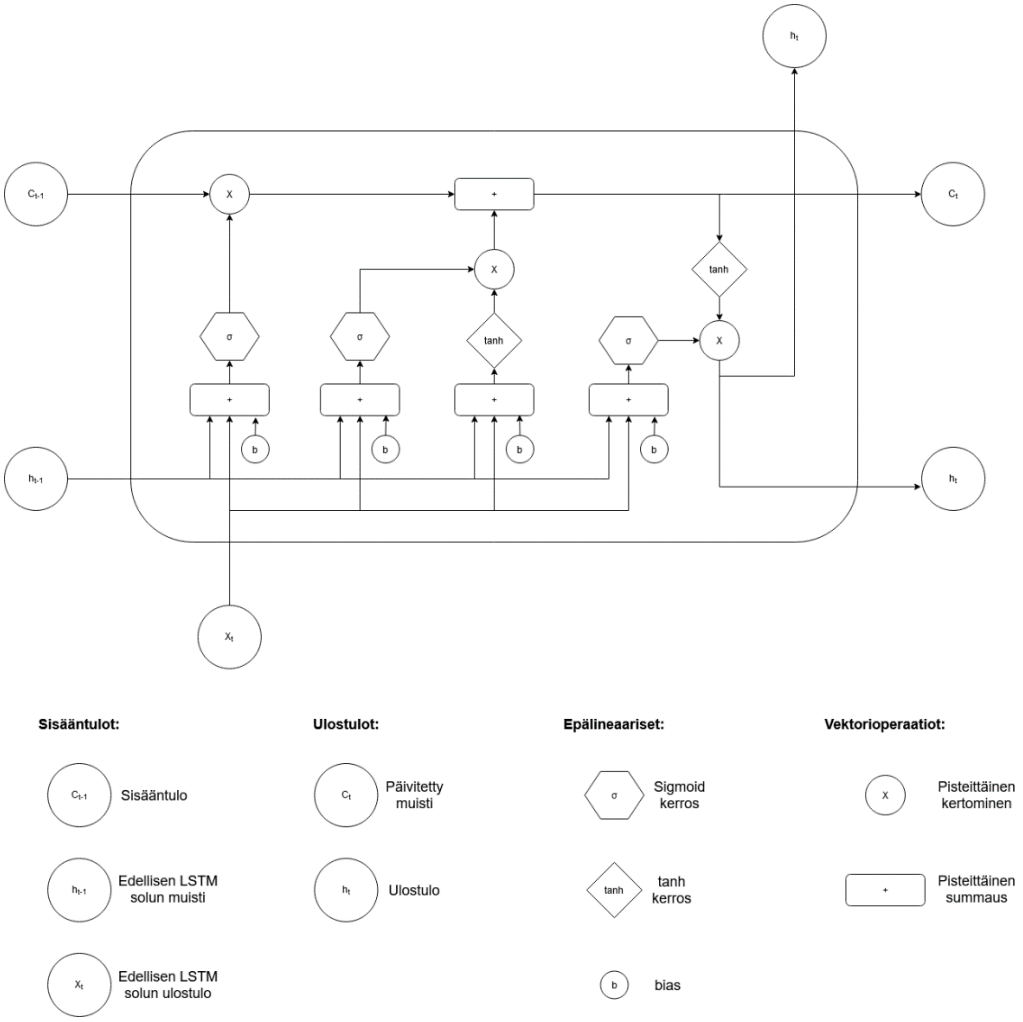
fastText-kirjasto sisältää useita esikoulutettuja sanavektoreita kuten tulevassa soveltavassa tutkimuksessa käytetyt 2 miljoonaa 300-ulotteista Common Crawl -korpuksella koulutettua sanavektoria.

fastTextin pohjana toimineesta word2vec-mallista löytää tarkempi kuvaus (Mikolov ym. 2013) julkaisusta.

3.3 LASER (Language-Agnostic SEntence Representations)

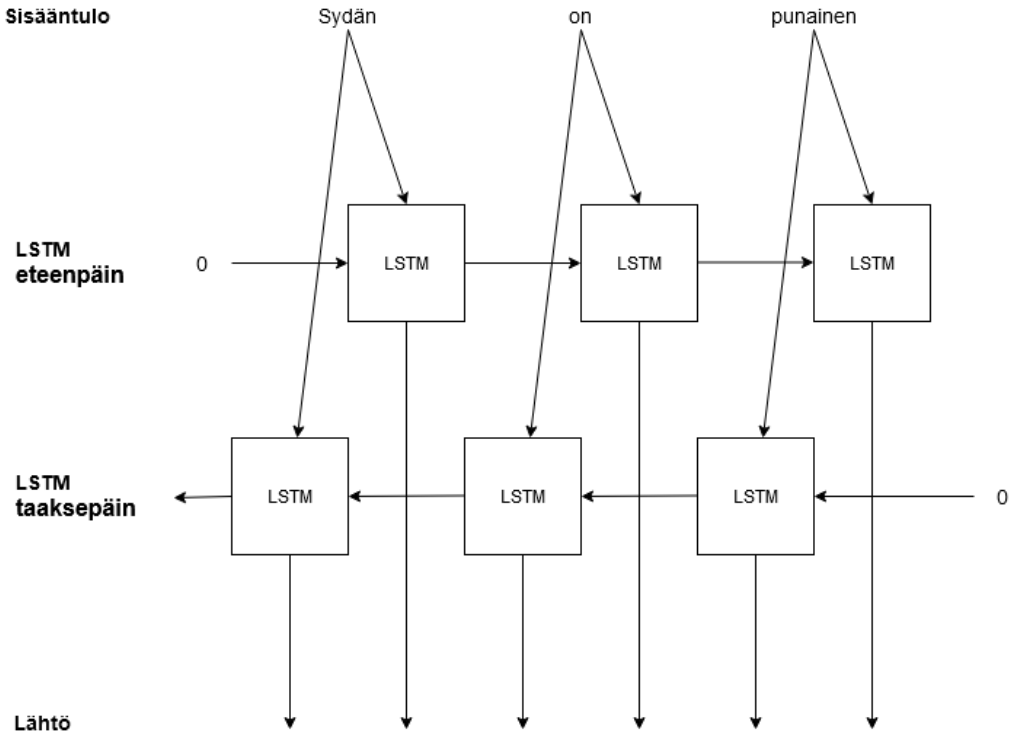
LASER on myös Facebook Research -tiimin kehittämä monikielinen kirjasto lausevektoreiden laskemiseen ja käyttämiseen. LASER on toteutettu BiLSTM neuroverkolla. (LASER 2020)

RNN on neuroverkko, jossa edellisen vaiheen tulos syötetään seuraavan vaiheen syötteeksi. Näin ollen sillä on ”muisti”, joka pitää sisällään edellisten vaiheiden informaation. RNN ei kuitenkaan toimi hyvin, jos käsitellään pitkiä lauseita, koska sen on vaikea muistaa pitkäaikaisia riippuvuuksia häviävän gradientin vuoksi. LSTM korjaa tämän ongelman pitämällä muistissaan kaksi tilaa. Toinen muuttuu vain vähän prosessin aikana (pitkä muisti) kun taas toinen muuttuu enemmän (lyhyt muisti) (Kuva 2).



Kuva 2. LSTM neuroverkon toimintaperiaate. (Papers With Code 2020a)

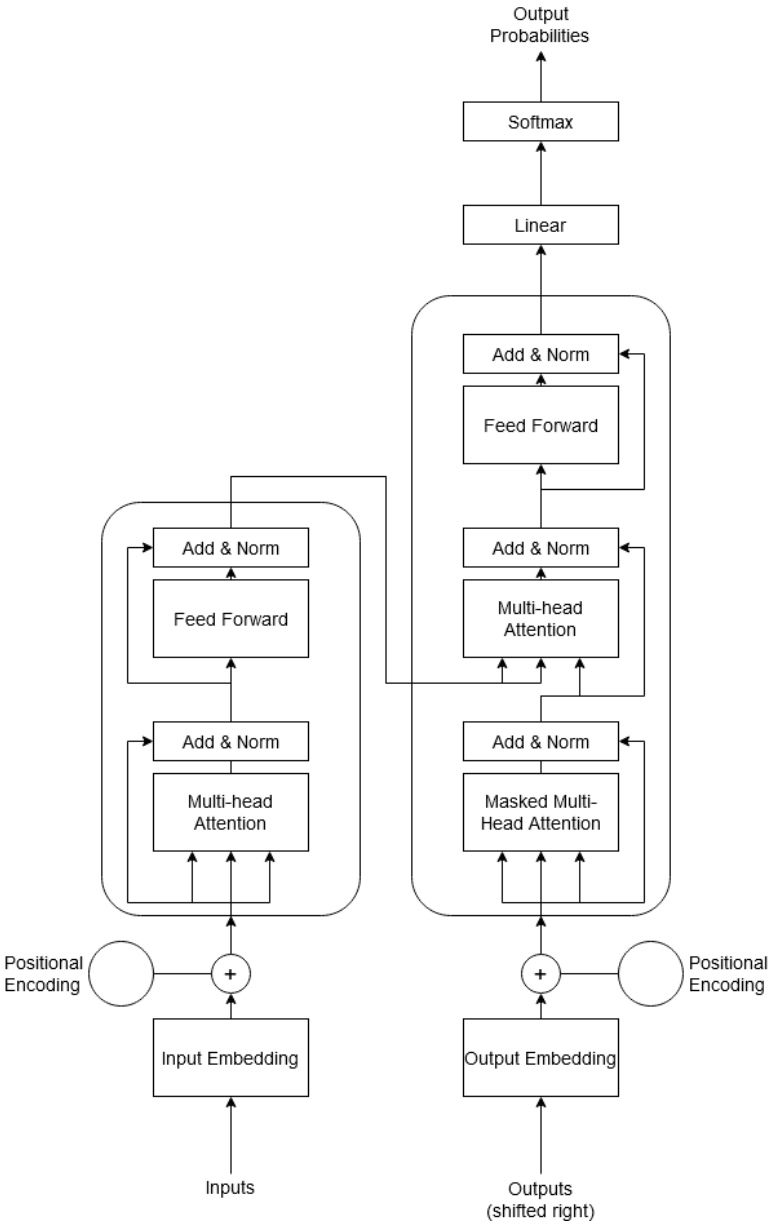
BiLSTM on muunnos LSTM:stä, joka yhdistää kaksi LSTM-neuroverkkoa, joista toinen käsittelee syötteen eteenpäin ja toinen taaksepäin. Tällä tavoin BiLSTM ymmärtää termin kontekstin paremmin kuin tavallinen LSTM-neuroverkko (Kuva 3).



Kuva 3. Yksinkertaistettu kuvaus BiLSTM-neuroverkon toimintaperiaatesta (Papers With Code 2020b).

3.4 BERT (Bidirectional Encoder Representations for Transformers)

BERT on Google Research -tiimin kehittämä Transformer-arkkitehtuuriin perustuva kielimalli (Devlin ym. 2019). Esimerkiksi Googlen hakukone käyttää BERT-mallia. Transfomer-arkkitehtuuri on täysin erilainen kuin aikaisemmat RNN-neuroverkkoon perustuvat mallit. Siinä missä RNN käsitteli syötteet peräkkäin, kykenee Transformer-arkkitehtuuri käsittelemään syötteitä rinnakkain (Kuva 4).



Kuva 4. Transformer arkkitehtuurin rakenne yksinkertaistettuna (Vaswani ym. 2017).

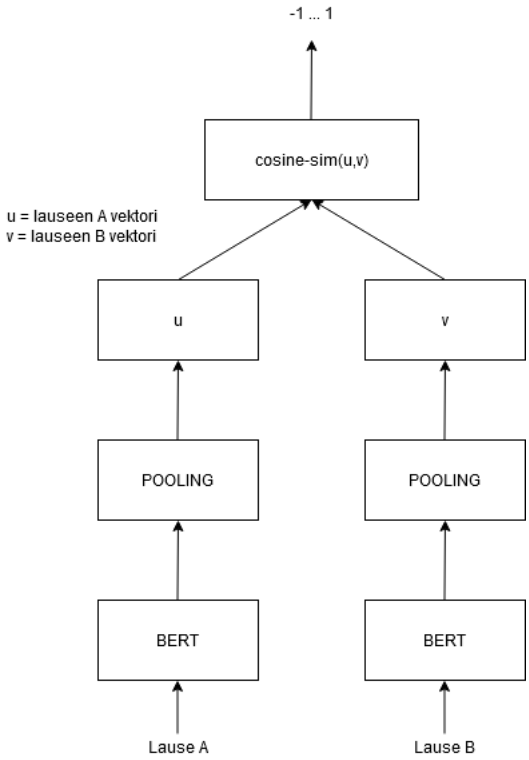
Vastaavasti kuin BiLSTM yhdisti kaksi LSTM neuroverkkoa, joista toinen käsitteli syötteen eteenpäin ja toinen taaksepäin, yhdistää BERT kaksi Transformer-neuroverkkoa, joista toinen käsittelee syötteen eteenpäin ja toinen taaksepäin (Devlin ym. 2019).

BERTin monimutkaisuuden takia sen tarkan toimintatavan selittäminen on tämän raportin laajuuden ulkopuolella. BERTin tutkimuspaperi (Devlin ym. 2019) kuvailee sen toimintaa tarkemmin. Transformer-arkkitehtuurin tarkka kuvaus löytyy ”Attention is all you need”-julkaisusta (Vaswani ym. 2017).

Sentence-BERT

Sentence-BERT on UKP Labin BERTistä kehittämä kielimalli, joka käyttää siamilais- ja triplet-neuroverkkoja semanttisesti merkittävien lausevektoreiden luomiseen. Rakenteensa ansioista Sentence-BERT luo lausevektorit paljon nopeammin kuin tavallinen BERT ja täten soveltuu tämän tutkimuksen tarkoitukseen paremmin kuin normaali BERT.

Sentence-BERT lisää pooling-operaation BERTin tulosteeseen luodakseen semanttisesti merkittäviä kiinteäulotteisia lausevektoreita, joita sitten on helppo vertailla esimerkiksi kosinisamankaltaisuudella (Kuva 5).



Kuva 5. Esimerkki kosinisamankaltaisuuden mittaamisesta Sentence-BERTillä (Reimers & Gurevych 2020).

Sentence-BERTin rakenteesta sekä tuloksista löytyy lisää tietoa ”Sentence-BERT: Sentence embeddings using siamese BERT-networks”-julkaisusta (Reimers & Gurevych 2020).

3.5 USE (Universal Sentence Encoder)

USE on toinen Google Research -tiimin kehittämä kielimalli lausevektoreiden luomiseen. USEsta on saatavilla kaksi eri versiota. Toinen on BERTin tavoin Transformer-arkkitehtuuriin perustuva malli ja toinen on DAN-arkkitehtuuriin perustuva malli.

Transformer-versio pohjautuu samaan Transformer-arkkitehtuuriin kuin BERT (Kuva 4). Sentence-BERTin tavoin se luo semanttisesti merkittäviä kiinteäulotteisia lausevektoreita. Tranformer-arkkitehtuuriin perustuva versio on DAN-versiota tarkempi, mutta vaatii huomattavasti enemmän laskenta-aikaa ja muistia.

DAN-versio on yksinkertainen neuroverkko, joka laskee keskiarvon sana- ja bi-grammivektoreista, jotka sitten syötetään 4-kerroksisen myötäkytketyn syvän neuroverkon läpi jonka tuloksena on lopullinen lausevektori (Cer ym. 2018).

4 TUTKIMUKSEN TOTEUTUS JA TULOKSET

4.1 Kielimallien asennus

Kaikki testit luotiin Python-ohjelmointikielellä Jupyter Notebook-tiedostoihin. Tiedostot luotiin niin, että ne voidaan ajaa millä tahansa tietokoneella tai pilvipalvelulla, jossa on Anaconda-ohjelmointiympäristö asennettuna. Anacondan ulkopuoliset ohjelmointikirjastot asennetaan Pythonin pip-asennustyökalulla (Koodi 1).

```
In [ ]: # LASER
        !pip install laserembeddings # https://github.com/yannvgn/laserembeddings
        !python -m laserembeddings download-models

        # SENTENCE-BERT
        !pip install transformers # https://github.com/huggingface/transformers
        !pip install -U sentence-transformers # https://github.com/UKPLab/sentence-transformers

        # UNIVERSAL SENTENCE ENCODER
        !pip install tensorflow
        !pip install tensorflow_hub

        # FASTTEXT
        !pip install fasttext
```

Koodi 1. Kielimallien asennus.

TF-IDF-metodi sisältyy scikit-learn-kirjastoon, joka taas sisältyy Anacondaan. Muita kielimalleja varten jouduttiin tekemään lisäasennuksia sekä lataamaan esikoulutettuja kielimalleja.

4.2 Kielimallien implementaatio

Olemassa olevien kirjastojen ansiosta kielimallien implementaatio onnistui varsin helposti. Kaikkien kielimallien implementaatio luotiin yhteen models.py-tiedostoon, jota Jupyter Notebookissa kutsumalla kielimalleja voitiin ajaa. Kaikki tarvittavat kirjastot tuodaan models.py-tiedostoon ja nimetään yleisen nimeämistavan mukaisesti (Koodi 2).

```
from sklearn.feature_extraction.text import TfidfVectorizer
import fasttext
import fasttext.util
import tensorflow_hub as hub
import pandas as pd

from laserembeddings import Laser
from sentence_transformers import SentenceTransformer
```

Koodi 2. models.py-tiedoston importit.

Kielimalleja ensimmäisen kerran ajaessa ne tarvittaessa initialisoidaan. Initialisoinnin yhteydessä kielimalli joko koulutetaan tai esikoulutetut kielimallit ladataan. Suurimmat kielimallit olivat yli 5 gigatavun kokoisia, joten ensimmäinen ajokerta on suhteellisen hidas.

4.2.1 TF-IDF

TF-IDF-implementaatio vaati eniten työtä, koska se esikoulutettiin itse STS Benchmarkin training-datalla (Koodi 3).

```
class Tfidf:

    def __init__(self):
        self.initialized = False

    def initialize(self):

        if self.initialized:
            return

        train_df = pd.pandas.read_table(
            '../sts_similarities/stsbenchmark/sts-train.csv',
            error_bad_lines=False,
            skip_blank_lines=True,
            usecols=[5, 6],
            names=["s1", "s2"])

        train_sentences = []

        for row in train_df.itertuples(index=False):
            train_sentences.extend((str(row[0]), str(row[1])))

        self.vectorizer = TfidfVectorizer(
            analyzer='char_wb', ngram_range=(3, 5))
        self.vectorizer.fit(train_sentences)

        self.initialized = True

    def get_sentence_vec(self, sentences):

        self.initialize()

        sentence_vectors = self.vectorizer.transform(
            sentences).toarray().tolist()

        return sentence_vectors
```

Koodi 3. TF-IDF-implementaatio.

Training-data luettiin Pandas-kirjastoa käyttäen taulukkoon, josta sitten luotiin scikit-learn kirjaston TfidfVectorizer-metodilla TF-IDF-sanavektorit 3–5 pituisilla n-grammeilla (scikit-learn 2020b).

4.2.2 fastText

fastText lataa sanavektorit ensimmäisen ajokerran yhteydessä. fastTextin esikoulutetut englanninkieliset sanavektorit olivat rakenteensa takia kaikista esikoulutetuista kielimalleista kooltaan suurimmat (noin 5,4 gigatavua) (Koodi 4).

```
class FastText:
    def __init__(self):
        self.initialized = False

    def initialize(self):

        if self.initialized:
            return

        fasttext.util.download_model('en', if_exists='ignore')  # English
        self.ft = fasttext.load_model('cc.en.300.bin')

        self.initialized = True

    def get_sentence_vec(self, sentences):

        self.initialize()

        sentence_vectors = [self.ft.get_sentence_vector(x) for x in sentences]

        return sentence_vectors
```

Koodi 4. fastText-implementaatio.

4.2.3 LASER

LASER-lausevektorit ladataan asennuksen yhteydessä, joten LASER-implementaatio ei vaatinut alustamista (Koodi 5).

```
class LaserModel:

    def get_sentence_vec(self, sentences):

        laser = Laser()
        sentence_embeddings = laser.embed_sentences(sentences, lang='en')

        return sentence_embeddings
```

Koodi 5. LASER-implementaatio.

LASER on monikielinen kielimalli, ja implementaation yhteydessä tulee muistaa valita käytettävä kieli.

4.2.4 Sentence-BERT

fastTextin tavoin Sentence-BERT lataa sanavektorit ensimmäisen ajokerran yhteydessä (Koodi 6).

```
class SentenceBert:
    def __init__(self):
        self.initialized = False

    def initialize(self):

        if self.initialized:
            return

        self.model = SentenceTransformer('bert-large-nli-stsb-mean-tokens')

        self.initialized = True

    def get_sentence_vec(self, sentences):

        self.initialize()

        sentence_embeddings = self.model.encode(sentences)

        return sentence_embeddings
```

Koodi 6. Sentence-BERT-implementaatio.

Tutkimuksessa käytettiin suurinta semanttisen samankaltaisuuden mittaamiseen koulutettua Sentence-BERT-mallia.

4.2.5 USE

USEn lausevektorit kuuluvat tensorflow hub -kirjastoon, joka lataa lausevektorit välimuistiin. Tästä syystä esimerkiksi kirjoittajan tietokoneella mallit ladattiin joka kerta kun Jupyter Notebook-tiedosto oli käynnistetty uudestaan (Koodi 7).

```
class Use:
    def __init__(self):
        self.initialized = False

    def initialize(self):

        if self.initialized:
            return

        module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
        self.model = hub.load(module_url)

        self.initialized = True

    def get_sentence_vec(self, sentences):

        self.initialize()

        sentence_embeddings = self.model(sentences)

        return sentence_embeddings
```

Koodi 7. USE-implementaatio.

Tätä tutkimusta varten käytettiin USEn DAN-versiota.

4.3 STS Benchmark

STS Benchmark on kokoelma lauseita vuosilta 2012–2017. Lauseet koostuva kuvaotsikoista, uutisotsikoista sekä keskustelupalstan kirjoituksista. STS Benchmark pitää sisällään 8 628 lauseparia, jotka on jaettu koulutus-, kehitys- ja testauskokoelmiin. Testi suoritetaan STS Benchmarkin dev-lausepareilla, joita on 1 500kpl.

Kielen asiantuntijat ovat arvioineet jokaisen lauseparin samankaltaisuuden välille 0–5 (Taulukko 1).

Taulukko 1. Lauseparien samankaltaisuuden kriteerit (Cheng & Zhou 2018)

Pisteet	Kriteeri
0	Lauseet ovat täysin erilaiset.
1	Lauseet ovat erilaiset, mutta samasta aiheesta.
2	Lauseet ovat erilaiset, mutta niissä on samoja yksityiskohtia.
3	Lauseet ovat samantyylisiä, mutta niissä on merkittäviä asiaeroja.
4	Lauseet ovat samantyylisiä, mutta niissä on merkityksettömiä asiaeroja.
5	Lauseet ovat merkitykseltään täysin vastaavia.

4.3.1 Kielimallien testaaminen STS Benchmarkin lauseilla

Malleja testataan STS Benchmarkin lauseilla laskemalla jokaisen lauseparin kosinisamankaltaisuus ja vertaamalla sitä ammattilaisten antamiin pisteisiin Pearsonin korrelaatiokertoimella (Pearson Correlation 2020). Mitä suuremman korrelaation malli saa, sitä paremmin sen antamat arvot vastaavat ammattilaisten antamia pisteitä. Tuloksia julkaistaessa kielimallin pisteet ilmoitetaan muodossa Pearsonin korrelaatiokerroin × 100. (Cheng & Zhou 2018)

Soveltavaa tutkimusta varten jokaiselle mallille luotiin oma Jupyter Notebook, jossa käydään läpi ensin kielimallin toimintaa yksittäisillä lauseilla ja tämän jälkeen testataan sen suorituskykyä STS Benchmarkin kehityslauseilla.

4.3.2 Testattujen kielimallien tulokset

Ennen tulosten julkistamista on hyvä käydä läpi testattujen kielimallien koulutus.

Koulutimme itse TF-IDF-mallin scikit-learn-kirjaston metodilla STS Benchmarkin training-lauseilla käyttäen 3–5 pituisia n-grammeja. fastTextille käytimme kahden miljoonan 300-ulotteisen sanavektorin mallia, joka on koulutettu Common Crawl -korpuksella. LASER on koulutettu Tatoeba-korpuksella, joka sisältää noin 112 000 lausetta 112:lla eri kielellä. Sentence-BERT sekä USE käyttävät molemmat koulutusdatassaan STS Benchmarkin training-lauseita.

Kielimallien epätasavertaisen koulutuksen takia tämän testin tulokset ovat kyseenalaisia ja puoltavat TF-IDF-, Sentence-BERT sekä USE-kielimalleja. Testistä kuitenkin nähdään selkeästi kielimallien vaatima laskenta-aika 1 500 lauseparin lausevektoreiden luomiseen. Testit suoritettiin tietokoneella, jossa on Nvidia GeForce GTX 1060 3GB-näytönohjain, Intel i5-6500-prosessori sekä 32 GB DDR4-muistia (Taulukko 2).

Taulukko 2. STS Benchmarkin tulokset. Pisteet ovat muodossa Pearsonin korrelaatiokerroin × 100.

Malli	Pisteet	Aika
TF-IDF	76,9	2,8 s
fastText	55,3	0,1 s
LASER	69,4	19,3 s
Sentence-BERT	87,4	67,3 s
USE (DAN)	80,1	5,3 s
USE (Transformer)	83,1	78,1 s

4.4 Testilausekkeet

STS Benchmark -testi ei antanut selkeää kuvaa mallien välisistä heikkouksista ja vahvuuksista, joten teimme lisätestejä itseluoduilla testilausekkeilla. Testit tehtiin englannin kielellä ja lauseet luotiin itse. Täten kielimallien esikoulutuksella ei ollut yhtä puoltavaa vaikutusta kuin STS Benchmark-testissä.

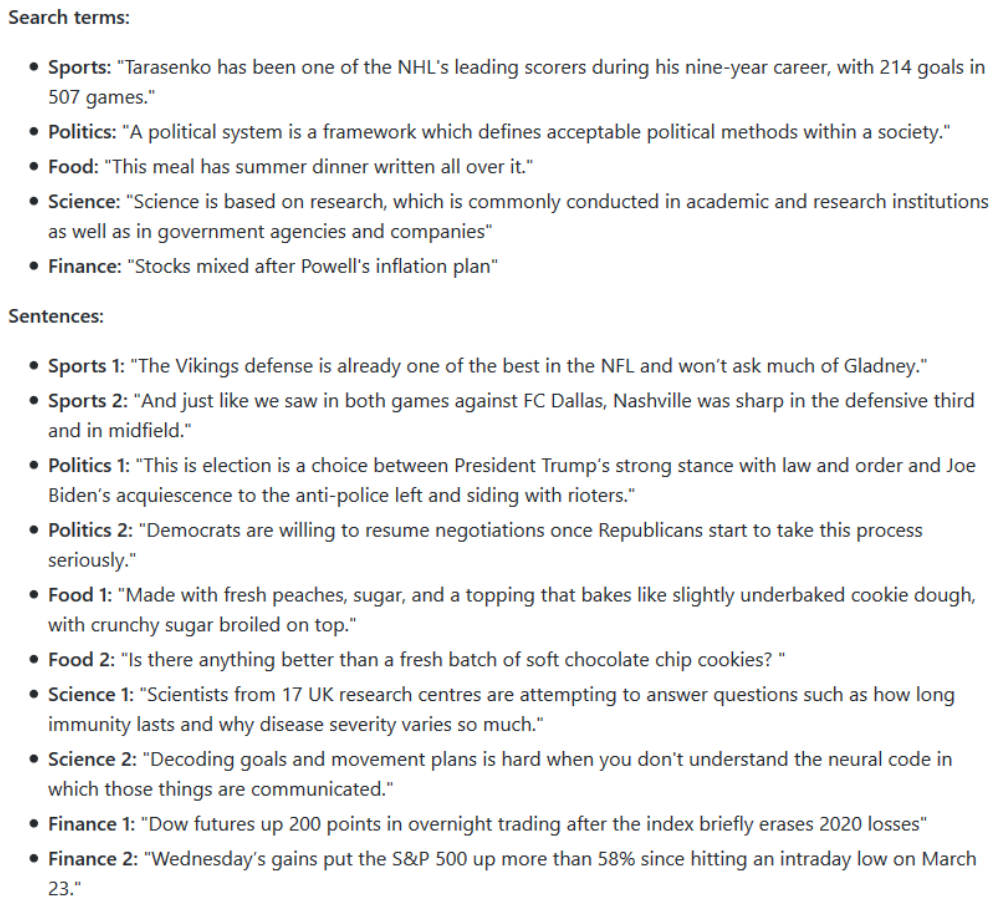
Kaikki testit tehtiin vertailemalla kohdelauseita keskenään kosinisamankaltaisuudella ja luomalla vertailuista lämpökartat. Testeissä luotiin ensin lämpökartat, joiden alueena oli kosinisamankaltaisuus välillä 0–1, mutta osaan testeistä luotiin selkeyden vuoksi lisänä lämpökartat, joiden alueena oli kosinisamankaltaisuus min-max. Tämä tehtiin sen vuoksi, että osassa testeistä kosinisamankaltaisuuden erot olivat niin pieniä, että niitä oli vaikea erottaa lämpökartasta täydellä 0–1 alueella (Koodi 8).

```
import seaborn as sns


from sklearn.metrics.pairwise import cosine_similarity

def plot_similarity(labels_y, labels_x, features_y, features_x):
    corr = cosine_similarity(features_y, features_x)
    sns.set(font_scale=1.2)
    g = sns.heatmap(
        corr,
        xticklabels=labels_x,
        yticklabels=labels_y,
        vmin=0,
        vmax=1,
        cmap="YlOrRd")
    g.set_xticklabels(labels_x, rotation=90)
    g.set_yticklabels(labels_y, rotation=0)
```

Koodi 8. Lämpökarttojen implementaatio, jossa alueena kosinisamankaltaisuus 0–1.

4.4.1 Semanttinen haku

Semanttista hakua varten valittiin viisi hakulauseketta sekä 10 lausetta, joihin haku kohdistettaisiin. Kaikki lauseet otettiin sen hetkisistä uutisartikkeleista, jotta ne vastaisivat mahdollisimman hyvin todellisuutta. Lauseita valittiin viidestä eri kategoriasta: sports, politics, food, science sekä finance (Kuva 6)



Search terms:

- **Sports:** "Tarasenko has been one of the NHL's leading scorers during his nine-year career, with 214 goals in 507 games."
- **Politics:** "A political system is a framework which defines acceptable political methods within a society."
- **Food:** "This meal has summer dinner written all over it."
- **Science:** "Science is based on research, which is commonly conducted in academic and research institutions as well as in government agencies and companies"
- **Finance:** "Stocks mixed after Powell's inflation plan"

Sentences:

- **Sports 1:** "The Vikings defense is already one of the best in the NFL and won't ask much of Gladney."
- **Sports 2:** "And just like we saw in both games against FC Dallas, Nashville was sharp in the defensive third and in midfield."
- **Politics 1:** "This is election is a choice between President Trump's strong stance with law and order and Joe Biden's acquiescence to the anti-police left and siding with rioters."
- **Politics 2:** "Democrats are willing to resume negotiations once Republicans start to take this process seriously."
- **Food 1:** "Made with fresh peaches, sugar, and a topping that bakes like slightly underbaked cookie dough, with crunchy sugar broiled on top."
- **Food 2:** "Is there anything better than a fresh batch of soft chocolate chip cookies? "
- **Science 1:** "Scientists from 17 UK research centres are attempting to answer questions such as how long immunity lasts and why disease severity varies so much."
- **Science 2:** "Decoding goals and movement plans is hard when you don't understand the neural code in which those things are communicated."
- **Finance 1:** "Dow futures up 200 points in overnight trading after the index briefly erases 2020 losses"
- **Finance 2:** "Wednesday's gains put the S&P 500 up more than 58% since hitting an intraday low on March 23."

Kuva 6. Semanttisen haun lausekkeet.

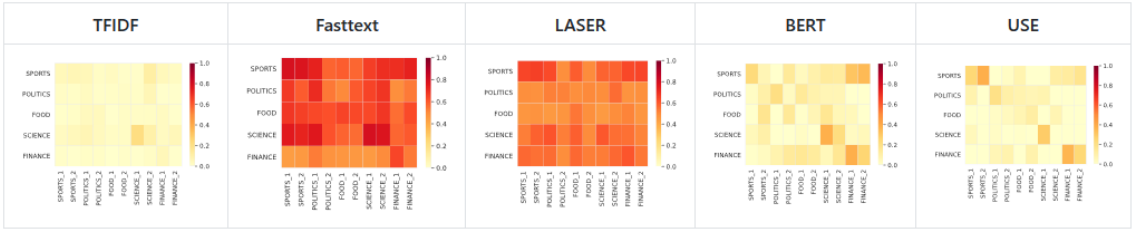
Testin tarkoituksena oli yrittää haastaa malleja löytämään yhteys saman kategorian hakulausekkeen ja kohdelausekkeiden välille. Lauseissa ei toistunut samoja sanoja eivätkä ne olleet välttämättä edes samasta aiheesta, joten mallin on ymmärrettävä avainsanojen yhteys hakulausekkeen ja kohdelauseiden välillä.

Semanttisen haun tulokset:

Testissä haettiin lämpökartalle selkeää diagonaalista linjaa vasemmasta yläkulmasta oikeaan alakulmaan. Tällöin malli yhdistää oikeat hakulausekkeet oikeisiin kohdelauseisiin.

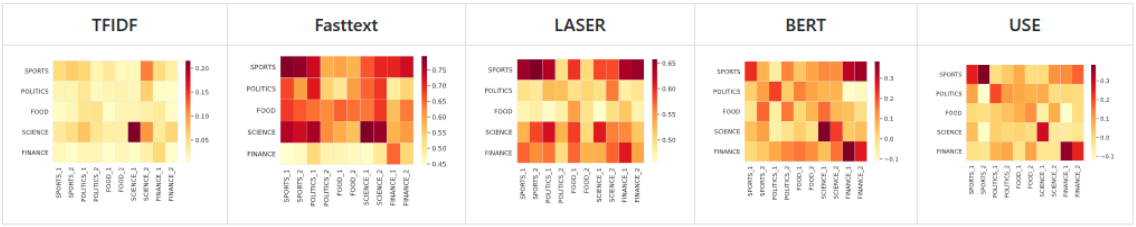
Semanttisen haun tuloksia tutkittaessa huomattiin heti, että lämpökartan käyttäessä kosinisamankaltaisuusasteikkoa 0–1 ei saada selkeää kuvaa mallin erottelukyvystä. Siitä kuitenkin huomattiin, että fastText ja LASER antavat muita malleja suurempia samankaltaisuusarvoja kaikille lauseille (Taulukko 3)

Taulukko 3. Semanttisen haun tulokset alueella kosinisamankaltaisuus 0–1.



Kun samankaltaisuusasteikko muutettiin asteikolle min-max, niin alettiin näkemään tuloksia. Jokainen malli tunnisti ainakin osan kategorioista oikein, mutta Sentence-BERT ja USE toimivat selkeästi tuloksekkaammin verrattuna muihin malleihin (Taulukko 4)

Taulukko 4. Semanttisen haun tulokset alueella kosinisamankaltaisuus min-max.



4.4.2 Synonyymilausekkeet

Synonyymilausekkeet ovat lauseita, jotka kirjoitetaan eri tavalla, mutta tarkoittavat samaa (Kielitiede: synonyymi 2020).

Testiä varten luotiin 5 synonyymilauseparia, joita vertailtiin keskenään (Kuva 7)

- **S1_a:** "The need for software developers has gone up by 50% in 5 years"
- **S1_b:** "The demand for programmers has doubled during the last five years"
- **S2_a:** "Personal computers entered the market in 1977"
- **S2_b:** "PCs came into shops in the late seventies"
- **S3_a:** "Symptoms of influenza include fever and nasal congestion."
- **S3_b:** "A stuffy nose and elevated temperature are signs you may have the flu."
- **S4_a:** "He has tons of stuff to throw away."
- **S4_b:** "He needs to get rid of a lot of junk."
- **S5_a:** "Her life spanned years of incredible change for women as they gained more rights than ever before."
- **S5_b:** "She lived through the exciting era of women's liberation."

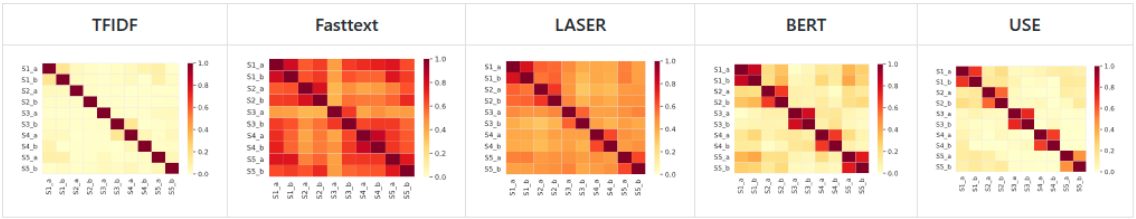
Kuva 7. Synonyymilausekkeet.

Synonyymilausekkeiden tulokset:

Synonyymilausekkeiden testissä haettiin myös vasemmasta yläkulmasta oikeaan alakulmaan kulkevaa linjaa. Erona kuitenkin se, että tavoitteena oli luoda neljän laatikon ryppäitä, jotka edustavat synonyymipareja.

Tässä testissä nähtiin taas selkeä etu Sentence-BERT ja USE mallien eduksi. LASER erottelivat oikeat lauseet toisistaan, mutta hieman heikommin kuin Sentence-BERT ja USE. fastText erotteli kunnolla vain osan lausepareista. TF-IDF ei tunnistanut kuin kaksi synonyymiparia (Taulukko 5)

Taulukko 5. Synonyymilausekkeiden tulokset alueella 0–1.



4.4.3 Homonyymilausekkeet

Homonyymilausekkeet ovat lauseita, joilla on samanlainen tai samankaltainen kirjoitusasu, mutta eri tarkoitus. (Kielitiede: homonyymi 2020)

Testiä varten luotiin 5 homonyymilauseparia (Kuva 8)

- **H1_a:** "She lies on the couch"
- **H1_b:** "She lies to the coach"
- **H2_a:** "Train muscles twice a week"
- **H2_b:** "Train departs twice a week"
- **H3_a:** "I want to book a room"
- **H3_b:** "I want to read a book in a room"
- **H4_a:** "These plants are huge" (plant = flower etc)
- **H4_b:** "Tesla plants are huge" (plant = factory)
- **H5_a:** "I saw a man"
- **H5_b:** "A man has a saw"

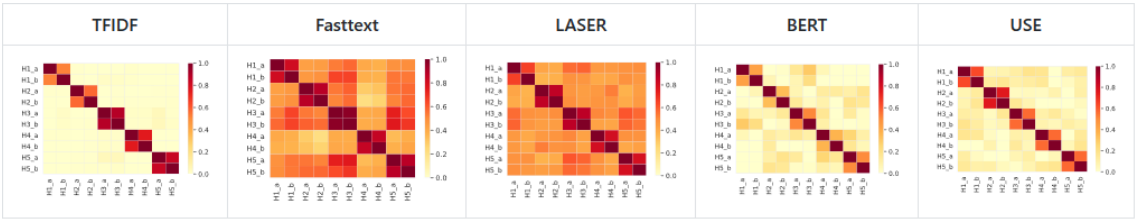
Kuva 8. Homonyymilausekkeet.

Homonyymilausekkeiden tulokset:

Homonyymilausekkeiden testissä haettiin vastaavaa tulosta kuin synonyymilausekkeissa. Erona oli kuitenkin se, että parien kesken tulisi olla selkeä erottelu kosinisamankaltaisuudessa.

fastTextiä lukuun ottamatta jokainen malli erotteli jokaisen lauseen oikein. Kuitenkin jälleen kerran Sentence-BERT ja USE toimivat muita malleja tehokkaammin lauseiden merkitystä erotellessa (Taulukko 6)

Taulukko 6. Homonyymilausekkeiden tulokset alueella 0–1.



4.4.4 Homonyymit hakulausekkeilla

Homonyymien vertaaminen keskenään ei antanut selkeää kuvaa mallien erottelukyvystä, joten lisäsimme ylimääräisen testin. Testiin kehitettiin viisi hakulauseketta, jotka kohdentavat jompaakumpaa homonyymilauseparin lausetta tarkoittamatta kuitenkaan täysin samaa asiaa kuin kohdelause. Tällä tavoin pystyimme haastamaan kielimallin luonnollisen kielen ymmärryksen, koska pelkkä semanttinen ja syntaksinen tieto ei välttämättä paljasta oikeaa lausetta (Kuva 9.)

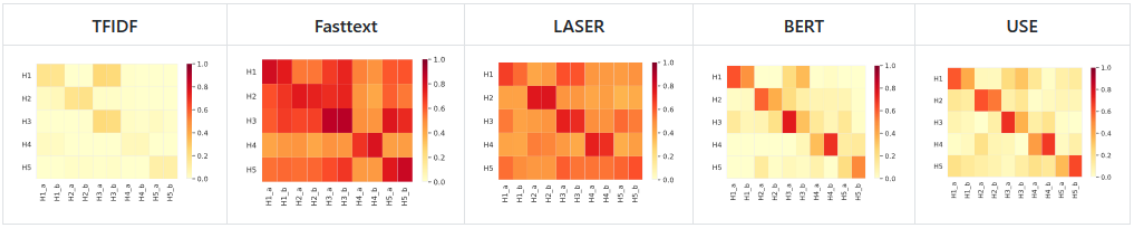
- **H1:** "She is resting in the living room" (targeting H1_a)
- **H2:** "Go to the gym two times per week" (targeting H2_a)
- **H3:** "I want a reservation from a hotel" (targeting H3_a)
- **H4:** "Electric car factories are big" (targeting H4_b)
- **H5:** "A man is cutting a tree" (targeting H5_b)

Kuva 9. Homonyymien hakulausekkeet.

Homonyymien haun tulokset:

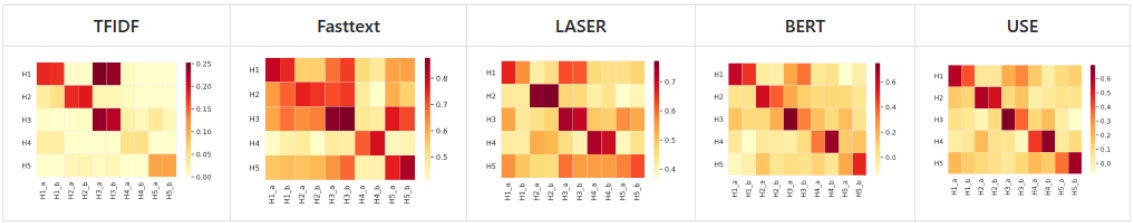
Homonyymien haun kanssa huomattiin sama kuin semanttisen haun kanssa, että lämpökartan käyttäessä kosinisamankaltaisuusasteikkoa 0–1 ei saada selkeää kuvaa mallin erottelukyvystä (Taulukko 7)

Taulukko 7. Homonyymien haun tulokset alueella 0–1.



Asteikolla min-max alettiin näkemään selkeitä eroja mallien tarkkuudessa. Sentence-BERT ja USE olivat ainoat mallit, jotka antoivat jokaiselle hakulausekkeelle oikean vastauksen. Kaikista tehdyistä testeistä tämä antoi selkeimmän kuvan mallien välisistä eroista kielen ymmärryksessä (Taulukko 8)

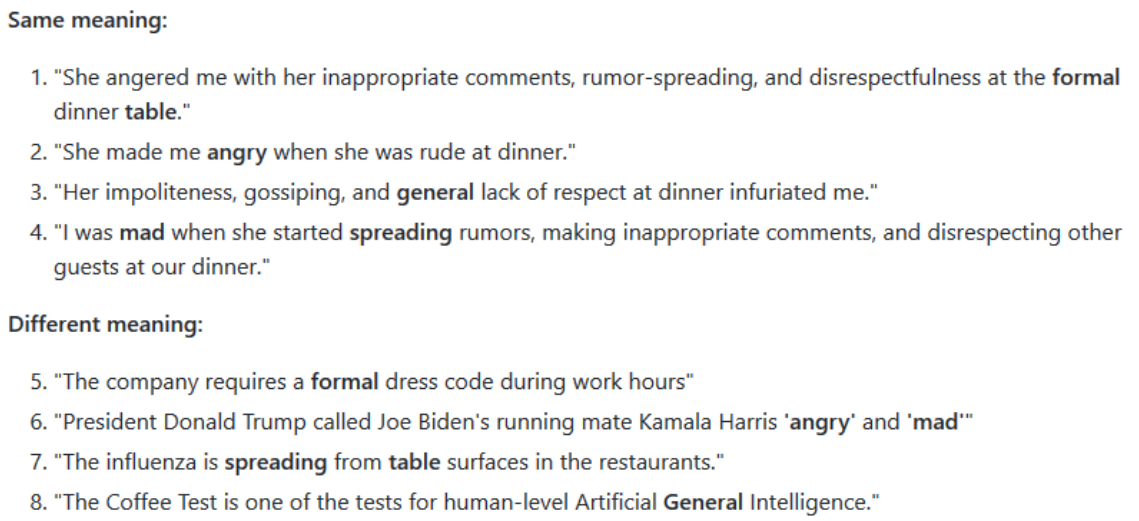
Taulukko 8. Homonyymien haun tulokset alueella min-max.



4.4.5 Samoja sanoja sisältävät lauseet, joilla on eri merkitys

Tähän testiin valittiin neljä samaa tarkoittavaa lausetta, jotka kuitenkin poikkesivat rakenteeltaan, sekä neljä eri tarkoituksen omaavaa lausetta, jotka sisältävät yhteisiä sanoja ensimmäisen neljän lauseen kanssa.

Lihavoinnilla merkityt sanat ovat sanojat, jotka toistuvat samaa tarkoittavissa ja eri tarkoituksen omaavissa lauseissa (Kuva 10)



Same meaning:

1. "She angered me with her inappropriate comments, rumor-spreading, and disrespectfulness at the **formal** dinner **table**."
2. "She made me **angry** when she was rude at dinner."
3. "Her impoliteness, gossiping, and **general** lack of respect at dinner infuriated me."
4. "I was **mad** when she started **spreading** rumors, making inappropriate comments, and disrespecting other guests at our dinner."

Different meaning:

5. "The company requires a **formal** dress code during work hours"
6. "President Donald Trump called Joe Biden's running mate Kamala Harris '**angry**' and '**mad**'"
7. "The influenza is **spreading** from **table** surfaces in the restaurants."
8. "The Coffee Test is one of the tests for human-level Artificial **General** Intelligence."

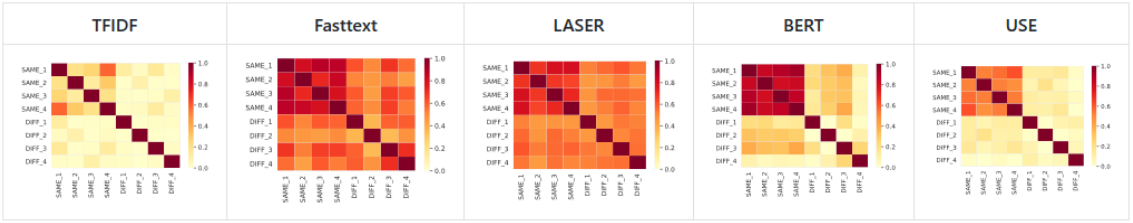
Kuva 10. Sama- ja erimerkityksiset lauseet.

Sama ja erimerkityksisten lauseiden tulokset:

Tämän testin tavoitteena oli jälleen luoda diagonaalinen viiva vasemmasta yläreunasta oikeaan alareunaan, mutta sen lisäksi 4 × 4 kokoinen tummempi laatikko vasempaan yläreunaan.

TF-IDF antoi odotetusti kaikista malleista pienimmän samankaltaisuuden neljälle samaa tarkoittavalle lauseelle. fastText ja LASER taas antoivat suhteessa pienemmän eron samaa ja eriä tarkoittavien lauseiden välillä, mutta onnistuivat kuitenkin erottelemaan ne. BERT ja USE jälleen onnistuivat parhaiten, mutta näistä kuitenkin Sentence-BERT antoi selkeästi suuremman samankaltaisuuden samaa tarkoittaville lauseille (Taulukko 9)

Taulukko 9. Samaa ja eriä tarkoittavien lauseiden tulokset alueella 0–1.



5 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli verrata TF-IDF-, fastText-, LASER-, Sentence-BERT- sekä USE-kielimallien eroja tarkkuudessa ja käytettävyydessä semanttisen samankaltaisuuden mittaamisessa. Opinnäytetyössä semanttinen samankaltaisuus määritettiin sana- ja lausevektoreiden kosinisamankaltaisuudella mitattavaksi arvoksi. Soveltavassa tutkimuksessa tutkittiin vertailtavien mallien rakennetta sekä niiden välisiä eroja semanttisen samankaltaisuuden mittaamisessa.

Soveltavan tutkimuksen toteutus sisälsi kaksi osaa. Ensimmäinen osa pohjautui STS Benchmarkin lausepareilla tehtävään testiin. Toisessa osassa pyrittiin tarkemmin selvittämään kielimallien välistä eroa omilla viiden eri tyypin testilausekkeilla. Kielimallien implementaatio tehtiin Python-ohjelmointikielellä Jupyter Notebook -tiedostoihin. Kaikki tutkitut lauseet olivat englanninkielisiä.

STS Benchmark-testi antoi kielimallien tarkkuudesta vääristävät tulokset, koska osa kielimalleista oli esikoulutettu STS Benchmarkin koulutuslausepareilla ja osa ei. Tarkemmin ottaen testi puolsi TF-IDF, Sentence-BERT ja USE kielimalleja. Testistä kuitenkin saatiin selkeä kuva mallien vaatimasta laskenta-ajasta. Transformer-arkkitehtuuriin perustuvat mallit olivat odotetusti tarkimpia, mutta myös hitaimpia. fastText oli selkeästi nopein, mutta myös epätarkin. Testin perusteella DAN-versio USEsta oli paras kompromissi tarkkuuden ja laskenta-ajan suhteen. Yksinkertaisin TF-IDF oli yllätyksenä vain hieman epätarkempi kuin kehittyneemmät Transformer- ja DAN-kielimallit.

Toisessa testissä kielimallien ymmärrystä haastettiin itse luoduilla testilausekkeilla ja kielimallien tulokset esitettiin lämpökartoilla, jotka kuvasivat lauseiden välistä kosinisamankaltaisuutta. Tämä testi antoi selkeämmän ja tasa-arvoisemman kuvan kielimallien tarkkuudesta oikeaa maailmaa vastaavissa tehtävissä. Ensinnäkin huomattiin, että fastText ja LASER antavat keskimäärin muita kielimalleja suuremmat samankaltaisuusarvot kaikille lausepareilla. Kaikissa testeissä Sentence-BERT ja USE (DAN) olivat keskenään lähellä toisiaan ja selkeästi muita malleja tarkempia. LASER oli tarkempi kuin TF-IDF sekä fastText. TF-IDF oli keskimäärin epätarkempi kuin fastText, mutta suoriutui kahdessa viidestä testissä paremmin.

Kokonaisuutena testeistä parhaimpana vaihtoehtona suoriutui USEn DAN-versio, joka oli tarkkuudeltaan samalla tasolla Sentence-BERT-kielimallin kanssa, mutta vaati vain noin 13 % Sentence-BERTin laskenta-ajasta.

Tämän tutkimuksen tulokset antavat hyvän lähtökohdan ohjelmistoprojektin kielimallia valittaessa. Lopullisen päätöksen tekeminen vaatii kuitenkin tarkempaa tutkimista ja mallien vertailua lopullisessa tuotteessa käytettävän datan kanssa. Tutkimus osoitti, että eri kielimallit toimivat paremmin tietyissä tilanteissa kuin toiset. Esikoulutuksessa käytetyllä datalla on usein suurempi vaikutus kuin kielimallin sisäisellä arkkitehtuurilla.

Opinnäytetyön parannuskohteena olisi ollut paneutua kielimallien rakenteeseen syvemmin, jotta aiheeseen perehtymättömän henkilön ei tarvitsisi tukeutua niin paljoa ulkoisiin lähteisiin. Tuloksia olisi myös voinut esittää useammalla tapaa, jotta kielimallien toiminnasta olisi saanut kokonaisvaltaisemman kuvan.

Jatkokehityksen kohteena voisi olla kielimallien tuloksien vertaaminen yhteisellä esikoulutuskorpuksella. Voisi myös tutkia pystyisikö yksinkertaisempia kielimalleja yhdistämällä saavuttaa Transformer-mallien kaltainen tarkkuus ilman, että laskenta-aika kasvaa merkittävästi.

LÄHTEET

Anaconda Invidual Edition 2020. Anaconda. Viitattu 10.11.2020. https://www.anaconda.com/

Cer, D.; Yang, Y.; Kong, S.; Hua, N.; Limtiaco, N.; John, R. St.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.-H.; Strope, B. & Kurzweil, R. 2018. Universal Sentence Encoder. http://arxiv.org/abs/1803.11175

Cheng, X. & Zhou, D. (2018). SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation. Mathematische Annalen, 371(1–2), 371–389. https://doi.org/10.1007/s00208-018-1662-3

Cosine distance 2020. Scipy. Viitattu 10.11.2020. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cosine.html

Cosine Similarity 2020. ML+. Viitattu 10.11.2020. https://www.machinelearningplus.com/nlp/cosine-similarity/

Devlin, J.; Chang, M. W.; Lee, K. & Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 1(Mlm), 4171–4186.

fastText 2020. Facebook Open Source. Viitattu 11.12.2020. https://fasttext.cc/

Jupyter Notebook 2020. Jupyter Team. Viitattu 10.11.2020. https://jupyter-notebook.readthedocs.io/en/stable/

Kielitiede: homonyymi 2020. Tieteen termipankki. Viitattu 2.12.2020. https://tieteentermipankki.fi/wiki/Kielitiede:homonyymi

Kielitiede: synonyymi 2020. Tieteen termipankki. Viitattu 2.12.2020. https://tieteentermipankki.fi/wiki/Kielitiede:synonyymi

LASER 2020. Facebook Research. Viitattu 11.11.2020. https://github.com/facebookresearch/LASER

Maind, A.; Deorankar, A.; & Chatur, P. 2012. Measurement of Semantic Similarity Between Words: A Survey. International Journal of Computer Science, Engineering and Information Technology, (2012), 2(6). https://doi.org/10.5121/ijcseit.2012.2605

Mikolov, T.; Chen, K.; Corrado, G. & Dean, J. 2013. Efficient estimation of word representations in vector space. 1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings, (2013), 1-12. http://arxiv.org/abs/1301.3781

Papers with Code 2020a. BiLSTM. Viitattu 11.11.2020. https://www.paperswithcode.com/method/bilstm#

Papers with Code 2020b. LSTM. Viitattu 11.11.2020. https://www.paperswithcode.com/method/lstm

Pearson Correlation 2020. Kent State University. Viitattu 16.11.2020. https://libguides.library.kent.edu/SPSS/PearsonCorr

Recurrent Neural Network 2020. IBM Cloud Education. Viitattu 11.12.2020. https://www.ibm.com/cloud/learn/recurrent-neural-networks

Reimers, N. & Gurevych, I. (2020). Sentence-BERT: Sentence embeddings using siamese BERT-networks. EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference, 3982–3992. Viitattu 10.11.2020. https://doi.org/10.18653/v1/d19-1410

scikit-learn 2020a. TF-IDF. Viitattu 10.11.2020. https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

scikit-learn 2020b. TfidfVectorizer. Viitattu 10.11.2020. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

STS Benchmark 2020. Eneko Agirre. Viitattu 12.11.2020. https://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł. & Polosukhin, I. 2017. Attention is all you need. Advances in Neural Information Processing Systems, 2017-Decem(Nips), 5999–6009.