KONEOPPIMINEN UUTISVIRRAN SUODATUKSESSA

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Tieto- ja viestintätekniikan koulutusohjelma, Riihimäen kampus

Syksy 2020

Henri Ailio

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka
Riihimäen kampus

Tiivistelmä

Tekijä Henri Ailio
Vuosi 2020

Työn nimi Koneoppiminen uutisvirran suodatuksessa

Ohjaaja Timo Karppinen

TIIVISTELMÄ

Opinnäytetyössä tarkastellaan miten koneoppimista voidaan hyödyntää uutisten luokittelussa erilaisiin kategorioihin ja kuinka suuria aineistoja tarvitaan hyvin toimivan luokittelijan opettamiseen. Opinnäytetyö jakautuu kahteen osaan, joista ensimmäisessä käsitellään tekstin luokitteluun liittyvää teoriaa. Toisessa osassa esitellään käytetyt teknologiat ja saadut tulokset.

Opinnäytetyössä käytettiin support vector classifier -algoritmia (SVC) ja bi-directional long short-term memory -tyyppistä neuroverkkoa (bi-LSTM) luokittelemaan uutisia. Molempia luokittelijoita testattiin kahdella eri aineistolla. Parhaimmillaan päästiin yli 97 % tarkkuuteen SVC-luokittelijalla. Johtopäätöksenä voidaan todeta, että neuroverkot vaativat enemmän koulutusmateriaalia toimiakseen hyvin kuin SVC-algoritmi. Tulokset eivät olleet vastaavat molemmissa aineistoissa, vaan käytetty aineisto vaikutti tuloksiin. Suomenkielisessä aineistossa bi-LSTM saavutti suuremman tarkkuuden, kuin SVC.

Avainsanat Koneoppiminen, tekoäly, neuroverkot, algoritmit
Sivut 26 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information technology
Riihimäki campus

Abstract

Author Henri Ailio
Subject Machine learning in news filtering
Supervisor Timo Karppinen

Year 2020

ABSTRACT

In this thesis machine learning as a tool for news filtering was researched. Machine learning was used to categorize news. The main research question was how much machine learning needs training data to work properly. The thesis consists of two parts. The first part discusses the theory and the second part explains which technologies were used and what kinds of results were obtained.

Support Vector Classifier (SVC) algorithm and bi-directional long short-term memory (bi-LSTM) neural network were used as classifiers. Two different datasets were used to train and test the classifiers. The SVC got the highest accuracy, over 97 %. It can be concluded that bi-LSTM needs more data to perform than SVC, but the used dataset affected the results. In the Finnish dataset bi-LSTM was more accurate than SVC.

Keywords Machine learning, artificial intelligence, neural networks, algorithms
Pages 26 pages

Sisällys

Taulukkoluettelo

Kuvaluettelo

1 Johdanto

Tämän opinnäytetyön aiheena on tutkia miten viime aikoina runsaasti esillä ollutta koneoppimista sovelletaan käytäntöön. Aihepiiriksi valikoitui uutisvirran suodattaminen eli miten koneoppimista voidaan hyödyntää uutisten automaattiseen luokitteluun. Aihe kuuluu luonnollisen kielen prosessoinnin alalle, joka on yksi osa-alue koko tekoälyn ja koneoppimisen kentässä.

Opinnäytetyö koostuu kahdesta osasta. Teoriaosuudessa käsittelen eri koneoppimismalleja ja miten niitä voidaan soveltaa tekstien käsittelyyn. Opinnäytetyön käytännön osuus muodostuu soveltuvien koneoppimismallien valinnasta, kouluttamisesta, testaamisesta ja niiden soveltamisesta. Lopputuloksena muodostuu sovellus, joka palauttaa käyttäjälle uutisen ja arvion sen sisällöstä.

Opinnäytetyön tutkimuskysymyksiksi määritin:

1. Mitkä koneoppimismallit soveltuvat luonnollisen kielen luokitteluun?
2. Kuinka paljon dataa koneoppimismalli tarvitsee, jotta se voi suodattaa uutisvirran 95 % tarkkuudella?

Opinnäytetyön painopisteenä on tekoälyn toteuttaminen ja siihen liittyvät osa-alueet. Työ kuvaa kaikkia osa-alueita, jotka tulee huomioida, kun tekoälyä lähdetään soveltamaan käytäntöön. Aihepiiri on saavuttanut siinä määrin kypsän vaiheen, että jokaista osa-aluetta ei tarvitse enää ohjelmoida alusta pitäen, mutta toisaalta tekoälyyn nojaavia sovelluksia käsittelevää kirjallisuutta on vielä melko vähän. Aiheeseen liittyvä kirjallisuus käsittelee enemmänkin yksittäisiä menetelmiä, ei niinkään niiden toteuttamista osana kokonaisuutta.

Opinnäytetyön lopussa käsittelen, miten valituilla menetelmillä päästiin asetettuihin tavoitteisiin. Samalla tuon esille tekoälyn hyödyntämiseen liittyviä rajoitteita, joita edelleen on.

2 Teoria

Teoriaosuus esittelee opinnäytetyön tutkimuskysymykset ja aiheeseen liittyvää teoriaa. Teoria koostuu koneoppimisen käsitteestä, luonnollisen kielen prosessoinnista, datan esikäsittelystä ja viimeisenä erilaisista koneoppimismalleista. Teoria rakentuu käytännön sovelluksen ympärille ja siten keskittyy vain pieneen osaan koko tekoälyn kentästä.

2.1 Koneoppiminen

Bellin (2015, s. 2) määritelmän mukaan koneoppiminen on tekoälyn yksi sovelluksista. Hänen mukaansa kyseessä on tietokonejärjestelmä, joka kykenee oppimaan tiedosta. Opettamisen myötä järjestelmä voi oppia paremmaksi ja se kykenee muodostamaan opitun perusteella vastauksia kysymyksiin.

Bell (2015, ss. 3-4) jakaa koneoppimisen kahteen osaan, valvottuun ja valvomattomaan. Valvotussa oppimisessa algoritmille annetaan näytteitä, jotka on valmiiksi luokiteltu. Luokiteltujen näytteiden avulla järjestelmä oppii luomaan mallin, jota käytetään luokitteluun. Valvomaton oppiminen tarkoittaa hänen mukaansa algoritmia, joka tunnistaa suurista tietojoukoista rakenteita. Bellin mukaan valvomaton oppiminen muistuttaa enemmän varsinaista oppimista, kun taas valvottu oppiminen on lähempänä tiedon louhintaa. (Bell, 2015, ss. 3-4)

Käyttäjältä tulevan syötteen perusteella tehtävä uutisvirran suodatus on valvottua koneoppimista. Käyttäjä toimii opettajana merkitsemällä ensin kaikista uutisista itseään kiinnostavat. Kun materiaalia on tarpeeksi, voidaan valitulla datalla opettaa koneoppimismalli, joka jatkossa luokittelee uutiset. Toisin sanoen koneoppimismalli antaa ennusteen mihin luokkaan uutinen kuuluu.

2.2 Luonnollisen kielen prosessointi

Luonnollisen kielen prosessointi (natural language processing, NLP) on osa tekoälyn kenttää. Ennen kuin koneoppimisalgoritmi voi alkaa rakentamaan teksteistä mallia, täytyy teksti muuttaa algoritmille soveltuvaan muotoon. Aggarwalin (2018, s. 26) mukaan prosessoinnin tarkoitus muuttaa jäsentymätön teksti jäsenneltyyn, moniulotteiseen muotoon. Aggarwalin mukaan yleisimmät prosessin vaiheet ovat:

1. Tekstin hankkiminen. Esimerkiksi internetistä tai RSS-syötteen XML:n muodossa olevat rakennetta osoittavat merkinnät ja merkit tulee huomioida, kun teksti otetaan käyttöön.
2. Stop-sanojen poistaminen. Stop-sanoiksi kutsutaan usein toistuvia sanoja, jotka eivät tuo lisäarvoa tekstin analysoinnin kannalta.
3. Stemmaus, kirjainkoon yhtenäistäminen ja välimerkkien poistaminen. Stemmauksella tarkoitetaan taivutetun sanan muuttamista perusmuotoonsa. Esimerkiksi sanan **kissan** perusmuoto on **kissa**. Esimerkiksi suuret alkukirjaimet muutetaan usein pieniksi ja kaikki välimerkit poistetaan.
4. Taajuuteen perustuva normalisointi. Usein harvoin toistuvat sanat tai rakenteet kuvaavat parhaiten tekstin erityispiirteitä. Tällöin hyvin harvoin esiintyvien sanojen painoarvoa voidaan nostaa luokittelussa. Tällaista normalisointia kutsutaan inverse-document frequency -normalisoinniksi. (Aggarwal, 2018, s. 27)

Edellä mainittu prosessi kuvaa yksinkertaisen bag-of-words -tyyppisen luokittelun vaiheita. Bag-of-words -muodossa sanojen järjestyksellä, merkityksellä tai lauseilla ei ole merkitystä. Aggarwalin mukaan bag-of-words -tyyppinen luokittelu on usein riittävä esimerkiksi tekstin luokitteluun tai suosittelujärjestelmiin. (Aggarwal, 2018, s. 2)

Edellä mainitusta prosessista haastavin vaihe on stemmaus. Jokainen kieli on rakenteeltaan yksilöllinen, joten algoritmien tulee olla sovitettu kyseiseen kieleen. Tässä työssä käytetään Pythonin NLTK-kirjaston Snowball-stemmeriä. Itse algoritmi on kehitetty vuonna 1980 ja suomen kielelle se on sovitettu vuonna 2002. (Finnish stemming algorithm, n.d.)

Kuten taulukosta 1 nähdään, ei Snowball-stemmeri ole täydellinen. Kettunen, Kunttu ja Järvelin (2005) vertailivat artikkelissaan eri menetelmiä ja totesivat, että Snowball häviää suomen kielen osalta noin 7,3 % tarkkuudessa FINTWOL-stemmaukselle. Heidän mukaansa ero on huomattavissa, mutta ei merkittävä. (Kettunen, Kunttu & Järvelin, 2005)

Taulukko 1. Esimerkki Snowball-stemmerin stemmauksesta. (Finnish stemming algorithm, n.d.)

Sana	**Stemmattu muoto**
edeltäjien	edeltäj
edeltäjiensä	edeltäjie
edeltäjiinsä	edeltäj
edeltäjistään	edeltäj
edeltäjiä	edeltäj
edeltäjiään	edeltäjiä
edeltäjä	edeltäj
edeltäjälleen	edeltäj

2.3 Tekstin esikäsittely

Tekstin luokittelu semanttisesti samankaltaisiin luokkiin on näennäisesti helppo tehtävä. Tekstit esikäsitellään, koodataan vektoreiksi ja lasketaan eri tekstien euklidinen etäisyys. Käytännössä kuitenkin prosessissa on useita vaikeasti hallittavia muuttujia.

Taulukossa 2 on esitetty kolme erilaista lyhyttä tekstiä ja ne on luokiteltu kuuluviksi joko politiikan, terveyden tai koulutuksen luokkaan. Jo näin yksinkertaisella esimerkillä voi huomata, että luokat eivät välttämättä ole yksiselitteisiä. Esimerkiksi politiikkaa käsittelevässä tekstissä voi olla huomattavasti koulutukseen liittyviä sanoja. Toisaalta myös koneoppimismallin opetukseen käytettävän datan jakauma voi olla erittäin epätasainen. Esimerkiksi uutisten luokittelu joko terrorismiin tai kotimaan politiikkaan kuuluviksi johtaa todennäköisesti erittäin epätasaiseen jakaumaan, koska terrorismista kirjoitetaan huomattavasti vähemmän.

Taulukko 2. Esimerkki erilaisten tekstien luokittelusta. (Yle 2020)

Teksti	**Luokka**
Kokoomuksen Petteri Orpon ja Kai Mykkäsen mukaan perussuomalaisten ehdotukset tarkoittaisivat työttömyyden lisääntymistä Suomessa.	Politiikka
Ryanair keskeyttää kaikki lennot Italiaan perjantaista alkaen. Ryanair lentää Suomesta Lappeenrannasta Bergamoon.	Terveys
Lähes 30 kuntaa osallistuu elokuussa jatkuvaan viisivuotiaiden maksuttoman varhaiskasvatuksen kokeiluun, kerrotaan opetus- ja kulttuuriministeriön tiedotteessa (siirryt toiseen palveluun).	Koulutus

Van Hulse, Khoshgoftaar ja Napolitano (2007) testasivat erilaisia menetelmiä, joilla pyrittiin parantamaan koneoppimismenetelmien tarkkuutta hyvin epätasaisesti jakautuneissa datamassoissa. Tutkimuksessa vertailtiin seitsemää eri menetelmää 35 datamassassa. Vähiten epätasapainossa olleessa aineistossa pienimmän luokan määrä oli 35 % muista, kun taas pahiten vinoutuneessa aineistossa pienimmän luokan osuus oli vain 1,33 %. Testatut menetelmät olivat satunnainen alinäytteistäminen (random undersampling), satunnainen ylinäytteistäminen (random oversampling), yksipuoleinen valinta (one-sided selection), parvipohjainen ylinäytteistys (cluster-based oversampling), Wilsonin editointi, SMOTE (Synthetic Minority Oversampling Technique) ja borderline-SMOTE. Testatuista menetelmistä parhaimmaksi nousi satunnainen alinäytteistäminen. Erityisen hyvin menetelmä sopi pahasti vinoutuneiden datamassojen tasapainottamiseen. Menetelmässä yliedustetusta luokasta otetaan satunnaisesti näytteitä niin, että epätasapaino poistuu. Toiseksi parhaaksi arvioitiin satunnainen ylinäytteistäminen. (Hulse, Khoshgoftaar & Napolitano, 2007)

Taulukossa 3 on esitetty kahden kuvitteellisen artikkelin koodaaminen vektoreiksi. Vektorisoinnissa koko opetusdata ladataan muistiin ja jokaiselle sanalle annetaan yksilöllinen indeksi. Tällöin esimerkiksi lause kissa kävelee kuumalla katolla muuttuu lukusarjaksi [234, 6784, 12, 33]. Toinen tapa esittää sanastoa on taulukon 3 tyyppisesti koodata sanojen esiintyvyys lauseessa tai artikkelissa binäärisesti (One Hot Encoding). Toisin sanoen, artikkelia esittävän vektorin pituus on koko sanaston pituinen ja vain esiintyneiden sanojen kohdalla on 1. (Aggarwal, 2018, s. 321)

Taulukko 3. Esimerkki tekstien koodaamisesta vektoreiksi.

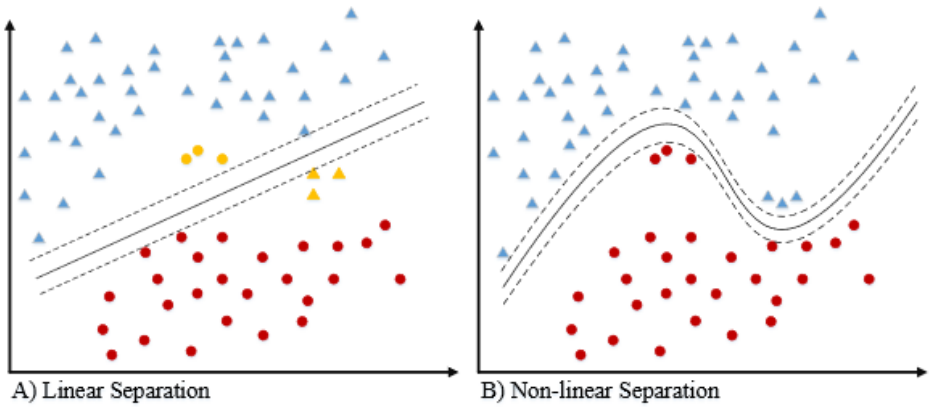
	kissa	koira	kävellä	syödä	lentokenttä	lentää	öljy
Artikkeli 1	1	1	0	1	0	0	0
Artikkeli 2	0	0	0	1	1	1	0

2.4 Algoritmit ja mallit

Kun teksti on esikäsitelty vektoreiksi, voidaan aloittaa koneoppimismallien käyttö. Erilaisia malleja on kehitetty lukuisia ja jokaisella on erilainen määrä asetuksia, jotka vaikuttavat tuloksiin. Zhou, Sun, Liu ja Lau (2015) vertasivat erilaisia luokittelijoita kysymysten luokitteluun. Opetusmateriaali koostui 5452 kysymyksestä, jotka oli jaettu kuuteen eri luokkaan. Testiaineisto koostui 500 kysymyksestä. Tarkastelussa SVM-luokittelijan (support vector machine) tarkkuus oli 95 % ja LSTM-luokittelijan 93,2 %.

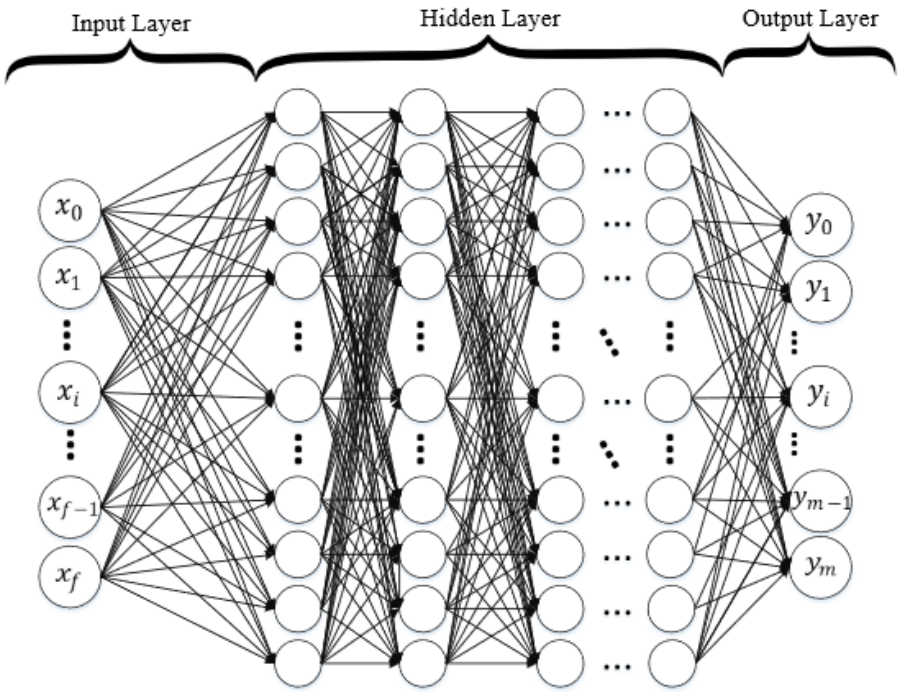
SVM-luokittelija kehitettiin alun perin vuonna 1963 ja se luotiin binääriseksi luokittelijaksi, mutta siitä on rakennettu sovelluksia moniluokitteluun. Kuvassa 1 on esitetty SVM:n tuottama luokittelu kahden kaksiulotteisen datan avulla. SVM pyrkii löytämään ”jakolinjan” eri luokkien välille, joka olisi mahdollisimman kaukana lähimmistä datapisteistä. (Kowsari, 2019)

Kuva 1. Lineaarinen ja ei-lineaarinen luokittelu SVM:n avulla. (Kowsari, 2019, s. 29)



Kuvassa 2 on esitelty tavallinen syvä feedforward- tyyppinen neuroverkko. Kowsarin (2019) mukaan syvä neuroverkko muodostuu sisääntulokerroksesta, yhdestä tai useammasta piilotetusta kerroksesta ja ulostulokerroksesta. Sisääntulon kautta syötetään verkolle ominaisuuksia ja piilotettujen kerroksien avulla neuroverkko pyrkii oppimaan tekstien ja niiden luokkien välisen riippuvuuden. Toisin sanoen, teksti koostuu yksittäisistä sanoista ja sanoista on muodostettu vektori. Vektori syötetään neuroverkolle, joka pyrkii löytämään vektorin ja sille asetetun luokan riippuvuuden. Tätä opittua riippuvuutta voidaan käyttää tekstien luokitteluun. (Kowsari, 2019, ss. 34-35)

Kuva 2. Täysin kytketty syvä neuroverkko. (Kowsari, 2019, s. 35)



Kowsarin (2019) mukaan LSTM-tyyppinen (long short term memory) neuroverkko on takaisinkytkeytyvän (recurrent neural network, RNN) neuroverkon sovellus. Takaisinkytkeytyvä neuroverkko käy datan läpi sekvenssinä ja pitää yllä tietoa edellisen askeleen informaatiosta. LSTM-verkon neuroni voi välittää informaatiota taaksepäin verkossa, poistaa tietoa tai syöttää sitä eteenpäin. (Kowsari, 2019, ss. 35-37)

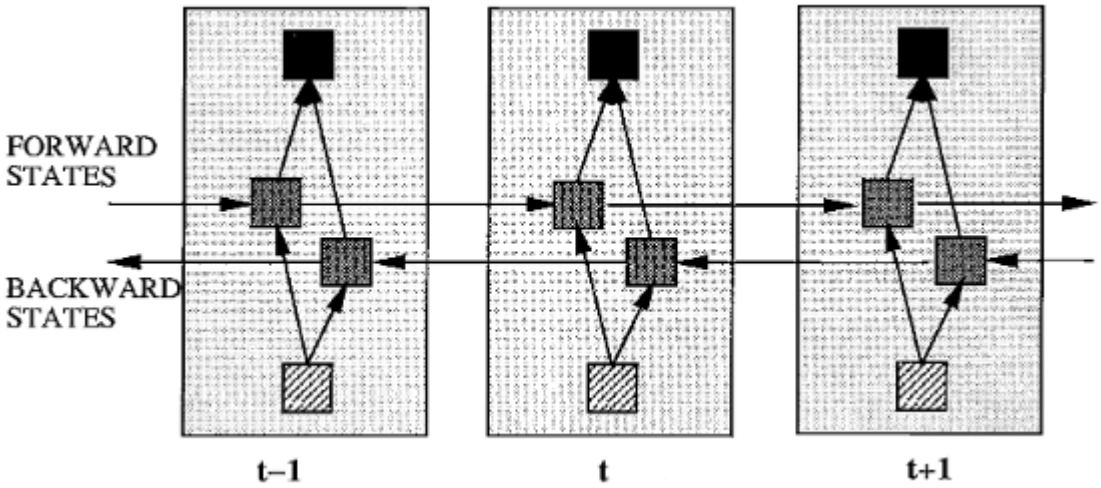
Nam, Kim, Mencia, Gurevych ja Frunkranz (2014) huomauttavat artikkelissaan, että vaikka SVM on tehokas luokittelija, siinä on puutteita aineiston kasvaessa. Heidän mukaansa SVM-luokittelijan suorituskyky laskee, kun tutkittavien nimikkeiden määrä kasvaa ja niiden jakauma ei ole tasapainossa. Ratkaisuksi he ehdottavat neuroverkkoja, joiden aktivointifunktiona käytetään ReLU-funktiota (rectified linear unit). Lisäksi heidän artikkelinsa mukaan käyttämällä dropout-kerrosta tai kerroksia ja pieniä oppimiskertoimia voidaan neuroverkon tarkkuutta parantaa.

ReLU-aktivointifunktio on yksi monista mahdollista aktivointifunktioista, joita voidaan käyttää neuroverkon neuronien aktivoinnissa. ReLU voidaan kirjoittaa muotoon $\Phi(v) = \max\{v, 0\}$ Toisin sanoen ReLU saa maksimiarvon nollan ja sisään tulevan arvon väliltä. Jos sisään tuleva arvo on alle 0, on ulos lähtevä arvo 0. Jos sisään tuleva arvo on suurempi, kuin 0, on ulos lähtevä arvo yhtä kuin sisään tuleva arvo. (Aggarwal, 2018, s. 324; DeepAi, n.d.)

Dropout-kerroksella tarkoitetaan syvän neuroverkon yhtä kerrosta, joka satunnaisesti sulkee neuronien välisiä yhteyksiä. Tällä pyritään estämään neuroverkon ylioppimista, jota tapahtuu helposti, kun neuroverkkoja opetetaan pienillä aineistoilla. Ylioppimisella tarkoitetaan tilannetta, jossa luokittelija on oppinut luokittelemaan opetusdatan niin tehokkaasti, että se ei kykene luokittelemaan testidataa hyvin. Toisin sanoen luokittelija toimii liian kapealla sektorilla, eikä kykene luokittelemaan materiaalia yleisemmin. (Hinton, Srivastava, Krizhevsky, Sutskever & Salakhutdinov, 2012)

Graves, Fernandes ja Schmidhuber (2005) tutkivat foneemien luokittelua bi-LSTM, LSTM ja RNN -tyyppisten neuroverkkojen avulla. Bi-LSTM-neuroverkko osoittautui parhaimmaksi luokittelijaksi. Schusterin ja Kuldipin mukaan bi-LSTM on RNN-tyyppisen neuroverkon sovellus, jossa neuroverkko opetetaan kaikella saatavissa olevalla tiedolla niin menneisyydestä kuin tulevaisuudestakin. Toisin sanoen bi-LSTM lukee sekvenssin ensin toiseen suuntaan ja sen jälkeen päinvastaiseen suuntaan muistiin. (Schuster & Kuldip, 1997) Bi-LSTM:n ideaa voi verrata ihmisen tapaan lukea tekstiä. Ihminen voi lukea kirjoitetun tekstin ensin normaalisti ja palata takaisinpäin tarvittaessa. Kuvassa 3 on esitetty Bi-LSTM-neuroverkon perusrakenne.

Kuva 3. Bi-LSTM-tyyppisen neuroverkon rakenne sekvenssin kolmessa vaiheessa. (Schuster & Kuldip, 1997)



Kaikissa edellä mainituissa malleissa opetusteksteistä tehdään kirjasto luokittelijan käyttöön. Jos luokiteltavassa tekstissä on sellaisia sanoja, joita mallin kirjastossa ei ole, se ei voi tulkita sen merkitystä. Toisaalta kaikilla sanoilla ei ole samanlaista informaatioarvoa. Tf-idf (term frequency–inverse document frequency) on yleisesti käytössä oleva menetelmä, jolla voidaan arvioida sanan merkitystä koko kirjastossa. Sana saa korkeamman painoarvon, jos se esiintyy usein hyvin harvoissa teksteissä. Tällaisessa tapauksessa sanan informaatioarvo on korkea. Toisaalta jos sana esiintyy harvoin, mutta lähes jokaisessa tekstissä, se sisällä juurikaan informaatioarvoa. (Cambridge University Press, 2009)

2.5 Koneoppimismallien arviointi

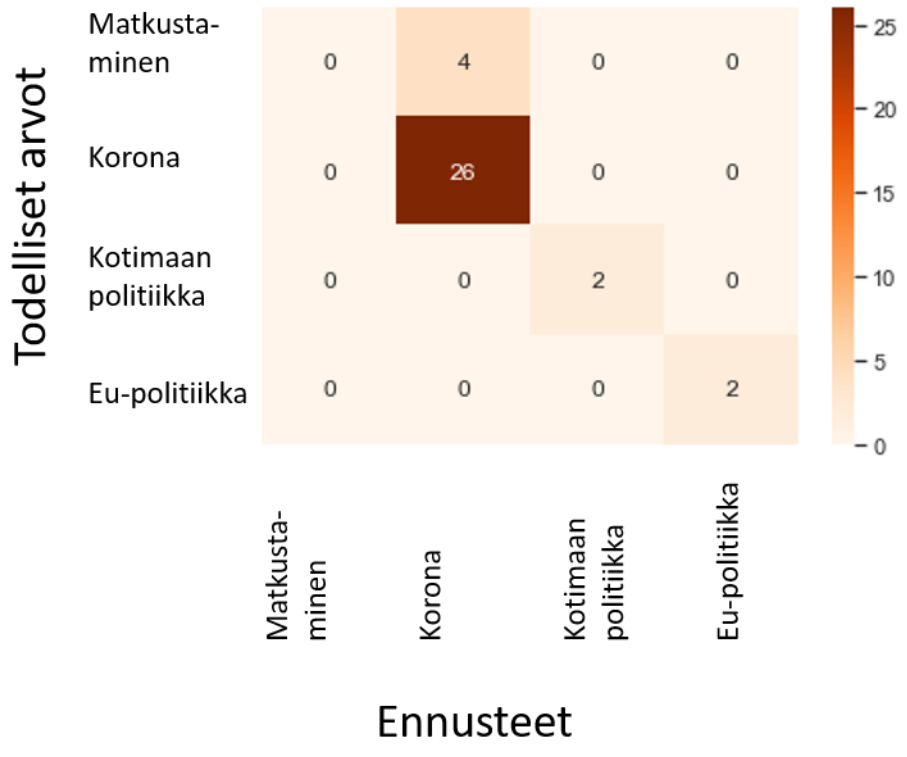
Luokittelevien koneoppimismallien kyvykkyyttä voidaan mitata useilla erilaisilla mittareilla. Mittareiden käytössä on kuitenkin huomioitava käytettävä luokittelu. Luokittelu voidaan tehdä binäärisesti, eli esimerkiksi onko artikkeli kiinnostava vai. Luokkia voi olla useita, jolloin on kyseessä moniluokittelusta (multi-class). Esimerkiksi eläin voidaan luokitella kissaksi tai koiraksi (Aly, 2005). Moninimikkeisessä (multi-label) luokittelussa taas kohde voi saada useita arvoja. Esimerkiksi eläin voi olla yhtä aikaa kissa ja kotieläin (Madjarov, 2012). Tässä opinnäytetyössä keskityn moniluokitteluun ja binääriseen luokitteluun.

Yksinkertaisin mittari on mallin tarkkuus. Tarkkuus on oikeaan osuneiden ennusteiden suhde kaikkiin ennusteisiin. Tarkkuus on tärkein tarkasteltavista mittareista, joskaan ei ainoa. Ongelmana tarkkuuden mittauksella on se, ettei se huomioi vääriä positiivisia tai vääriä negatiivisia tuloksia. Lisäksi erittäin epätasaisesti jakautuneissa aineistossa pelkkä tarkkuus ei kerro riittävästi mallin hyvyydestä. Jos esimerkiksi aineistosta 96 % kuuluu yhteen luokkaan 1 ja 4 % luokkaan 2, luokittelemalla koko aineisto luokkaan 1 on tarkkuus mallin 96 %. (Google, 2020)

Kahden luokan suhteen luokittelijan hyvyyttä voidaan tarkastella ROC-kaaviolla (receiver operating characteristic curve), jolla visualisoidaan kuinka hyvin koneoppimismalli ennustaa luokat oikein. Kun luokkia on enemmän kuin kaksi, paremmin väärien positiivisten ja negatiivisten määriä voidaan hahmottaa confusion-matriisin avulla (confusion matrix). (Google, 2020)

Kuva 4 esittää kuvitteellisen luokittelijan tuloksia confusion-matriisin avulla. Kuvitteellisessa aineistossa on ollut neljä luokkaa, matkustaminen, korona, kotimaan politiikka ja EU-politiikka. Vaakarivillä esitetään mallin ennusteet ja pystyrivillä todelliset arvot. Esimerkin perusteella malli on ennustanut kaikki korona-uutiset oikein, mutta lisäksi se on luokitellut neljä matkustamiseen liittyvää uutista myös korona-uutisiksi.

Kuva 4. Esimerkki confusion-matriisista.



3 Koneoppimismallien soveltaminen uutisten luokitteluun

Tässä tutkimuksessa teorian ohjaamana luotiin kokonaisuus, joka kykenee luokittelemaan uutisia ja palauttamaan uutiset luokitteluineen käyttäjälle. Käytännön sovelluksen luominen alkoi Jupyter-notebookien avulla tehdyistä prototyypeistä ja päättyi yksinkertaiseen Flask-pohjaiseen palveluun, joka lukee RSS-syötettä, analysoi sen ja palauttaa virran luokitteluineen käyttäjälle jälleen RSS-syötteenä.

3.1 Käytetyt teknologiat

Käytettyjen teknologioiden valintaa ohjasi ennen kaikkea saatavilla oleva kirjallisuus ja teknologioiden levinneisyys. Tarkoituksena oli löytää ennemminkin hyvin dokumentoituja ja toimivaksi todettuja menetelmiä, kuin kokeellisia ja mahdollisesti lyhytikäisiä välineitä.

Python-ohjelmointikieli on tekoälyn ja data-analytiikan saralla lähes lingua francan tasolla. Toisaalta Pythonilla on mahdollista toteuttaa palvelinratkaisuja, joita tarvitaan frontendin toteuttamisessa. Kaikki opinnäytetyön ohjelmoitavat osuudet toteutettiinkin Pythonilla.

Pythonille on toteutettu useita koneoppimis- ja tekoälykirjastoja. Näistä käyttöön valikoituivat Scikit-learn ja Tensorflow. Scikit-learnin kehittäminen alkoi vuonna 2007 ja sen avulla voidaan toteuttaa erilaisia valvottuja- ja valvomattomia koneoppimismenetelmiä. Lisäksi kirjasto tarjoaa erittäin hyvät välineet mallien arvioimiseen.

3.2 Datan keräys ja tekstin esikäsittely

Koneoppimisen opettaminen vaatii riittävän datamäärän keräämisen. Tarvittava aineisto kerättiin Ylen uutisista lukemalla säännöllisesti RSS-syötettä ja tallentamalla uutiset. Lukeminen ja tallennus tehtiin yksinkertaisella Python-skriptillä, joka käynnistettiin Cron-työkalulla Linux-työasemalla. Uutisten julkaisutahti on siinä määrin rauhallinen, että lukeminen voitiin ajoittaa tapahtuvaksi kerran vuorokaudessa. Lopputuote mahdollistaa RSS-syötteen tallentamisen samalla, kun se luetaan.

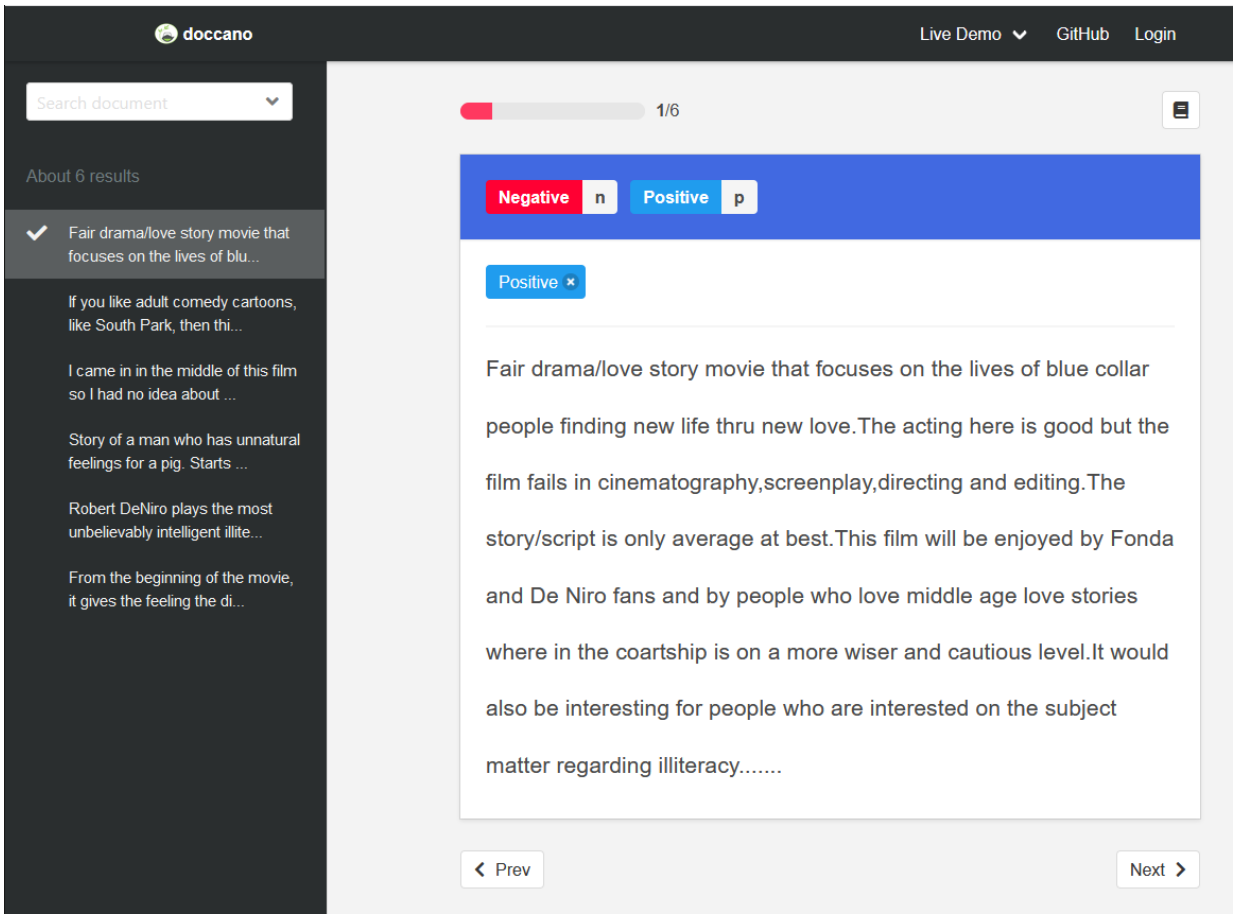
Kerätyt uutiset piti luokitella ennen varsinaista opetusvaihetta. Suoraviivaisin ja pienten aineistojen kanssa toimivin vaihtoehto on käyttää taulukkolaskentaohjelmaa. Uutisten osalta ongelmaksi muodostuu helposti niiden pituus. Muutamien satojen uutisten luokittelu manuaalisesti taulukkolaskentaohjelman avulla onnistuu, mutta se ei ole erityisen miellyttävää.

Pitkien artikkelien ja suurempien aineistojen käsittelyyn toimii paremmin esimerkiksi Doccano-annotointityökalu[1]. Avoimen lähdekoodin Doccano mahdollistaa annotointityön jakamisen lisäksi useammalle henkilölle. Suurin hyöty on kuitenkin luokitteluiden määrittelyiden käyttö. Binäärisen luokittelun ja pienten aineistojen käsittelyssä luokkien määrittely on helppo pitää mielessä. Tilanne muuttuu haasteelliseksi, kun aineiston ja käytettävien luokkien määrä kasvaa. Opinnäytetyössä käytetty suomenkielinen aineisto annotoitiin taulukkolaskentaohjelman avulla, mutta suurempien aineistojen osalta Doccanon tai vastaavan käyttö olisi järkevää.

Tekstien esikäsittelyä varten kirjoitettiin oma luokka, joka toteutti NLTK-kirjaston avulla tokenisoinnin, sanojen muuntamisen pieniksi kirjaimiksi, sanojen stemmauksen ja stop-sanojen poistamisen. NLTK-kirjasto mahdollistaa useiden kielien käyttämisen sekä stemmaukseen että stop-sanojen poistamiseen. (NLTK Project, 2020) Samaa esikäsittelyluokkaa voi käyttää esimerkiksi suomen ja englanninkielisten tekstien käsittelyyn. Stemmerinä käytettiin Snowball-stemmeriä. Snowball-stemmer toimii seuraavilla kielillä: arabia, tanska, hollanti, englanti, suomi, ranska, saksa, unkari, italia, norja, portugali, romania, venäjä, espanja ja ruotsi (NLTK Project, 2020).

[1] Annotoinnilla tarkoitetaan tässä työssä aineiston nimeämistä luokkiin

Kuva 5. Kuvankaappaus Doccano-työkalun demosta. Esimerkissä on kyseessä binäärinen luokittelu, jossa aineisto luokitellaan positiiviseksi tai negatiiviseksi. (Hironsan, n.d.)



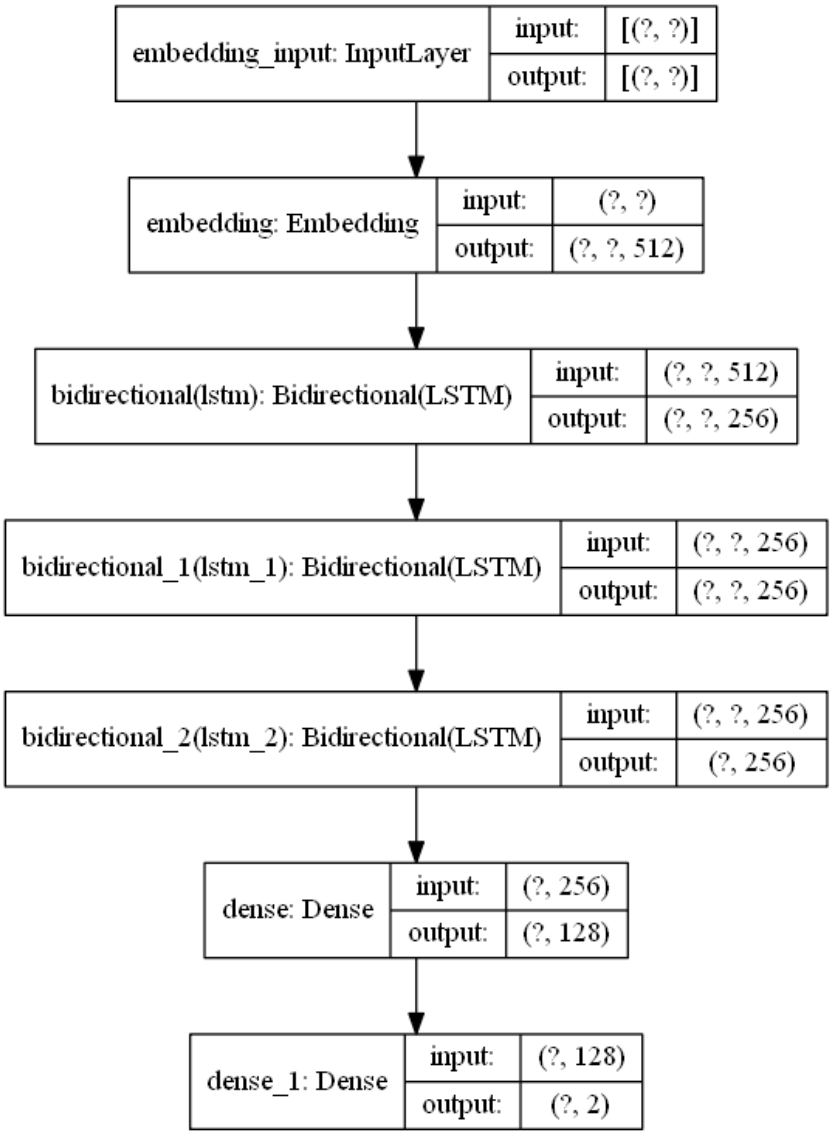
3.3 Koneoppimismallit ja niiden käyttö

Tässä tutkimuksessa käytetyiksi malleiksi valittiin lopulta SVC ja LSTM-neuroverkko. SVC toteutettiin Scikit-learn-kirjastolla ja LSTM Tensorflow-kirjastolla. Luokittelijat toteutettiin kirjoittamalla abstraktointiluokka, joka mahdollisti mallien opettamisen, tulosten tallentamisen tekstitiedostoon ja luokittelijan tallentamisen jatkokäyttöä varten. Edellä mainittujen lisäksi testattiin tavallista feedforward-neuroverkkoa, mutta se ei toiminut merkittävästi paremmin kuin SVC ollen huomattavasti monimutkaisempi ja raskaampi. SVC:n etuna on sen suhteellinen nopeus ja kyky oppia kohtuullisen hyvin pienistäkin aineistomääristä. Sen sijaan, kun koulutusaineiston määrä kasvaa, alkaa LSTM toimimaan paremmin.

LSTM:n ongelmana on sen raskaus ja taipumus niin sanotusti unohtaa aineistoa (vanishing gradient problem). Ongelma on merkittävämpi uutisten kuin esimerkiksi Twitterin twiittien osalta, koska uutiset ovat usein muutamien tuhansien sanojen mittaisia. LSTM osoittautui kuitenkin toimivaksi, kun luokitteluun käytettävä tekstiosuus rajattiin riittävän pieneksi ja se otettiin uutisen alusta. Eli esimerkiksi 1500 sanan mittaisesta uutisesta otettiin ensimmäiset 200 sanaa luokitteluun. LSTM:n käyttö poikkeaa tavallisen neuroverkon ja muiden algoritmien käytöstä siinä, että muille syötetään koko artikkeli bag-of-words-tyyppisesti. LSTM sen sijaan käsittelee artikkelin sanojen sekvenssinä, koska sanojen järjestyksellä on merkitystä.

Käytetty LSTM-malli on jokseenkin raskas, mutta se osoittautui varsin toimivaksi. Malli on esitetty kuvassa 6. Bi-LSTM osaa huomioida sanojen esiintymisjärjestyksen molempiin suuntiin ja neuroverkon syvyyden lisääminen parantaa sen tarkkuutta. Toisaalta monimutkaisempi malli tekee laskentaprosessista vaativamman ja raskaamman. Käytännössä näin raskaan mallin käyttö ei ole kovinkaan käytännöllistä ilman näytönohjainkiihdytystä.

Kuva 6. Käytetyn LSTM-mallin rakenne.



Sekä SVC:n että LSTM:n käyttö vaatii useiden parametrien optimoimista. SVC:n osalta kokeiltiin grid search-menetelmää, jossa luokittelija opetetaan useilla eri parametreilla ja valitaan näistä parhaat. Luotu SVC:n toteuttava luokka mahdollistaa sekä grid searching käytön, että yksittäisten parametrien muuttamisen. LSTM:n koulutus on raskaampi prosessi ja parametrien optimointia tehtiin pelkästään käsin.

Molempiin luokittelijoihin luotiin mahdollisuus tasapainottaa koulutusaineistoa Scikit-learn-kirjaston avulla. Lisäksi abstraktointiluokat mahdollistavat koulutusaineiston jakamisen testi- ja koulutusaineistoon. Tällöin luokittelijan ja sen parametrien toimintaa voi testata luotettavasti ja sopivien parametrien löydyttyä kouluttaa luokittelija käyttäen koko aineistoa. Molemmilla luokittelijoilla voidaan toteuttaa binäärinen tai moniluokallinen luokittelija.

LSTM:n loss-funktioiksi valittiin binary crossentropy ja categorical crossentropy. Näistä ensin mainittua käytetään binääriseen luokitteluun ja jälkimmäistä moniluokitteluun. Luokittelija palauttaa sekä binäärisessä että moniluokittelussa listan todennäköisyyksiä. Esimerkiksi binäärisessä luokittelussa luokittelija palauttaa listan [0.2, 0.8], jossa 0.2 on vaihtoehdon 0 todennäköisyys ja 0.8 vaihtoehdon 1 todennäköisyys.

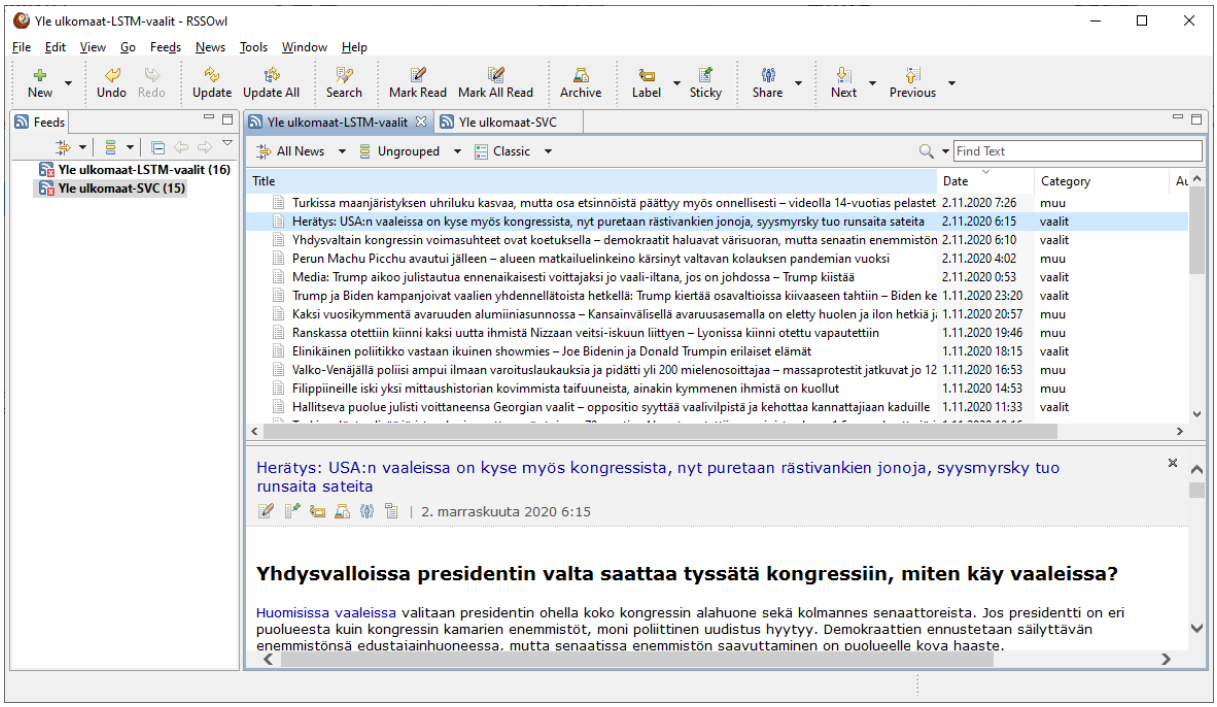
Mallien arvioimiseksi tarkasteltiin useita eri muuttujia. Käytännössä tärkeimpiä arvoja LSTM:n osalta olivat testi- ja koulutusaineistojen tarkkuus sekä loss-arvo. Kuvassa 6 on esitetty, miten yhdessä ajossa mallin tarkkuus muuttui koulutussyklien suhteessa.

3.4 Flask ja RSSOwl

Käyttäjäystävällisen lukuympäristön luominen on oma haasteensa. Tässä työssä ongelma ratkaistiin luomalla Flask-pohjainen palvelin, joka huolehti uutisten hakemisesta ja luokittelemisesta. Flaskille määritettiin muutamia url-osoitteita, joita kutsumalla Flask tekee RSS-kyselyn ennalta määritettyyn osoitteeseen, luokittelee vastauksena saadut uutiset ja palauttaa luokitellut uutiset ja luokittelijan arvioin uutisen sisällöstä. Tällöin käyttäjä voi käyttää mitä tahansa RSS-lukijaa uutisten lukemiseen.

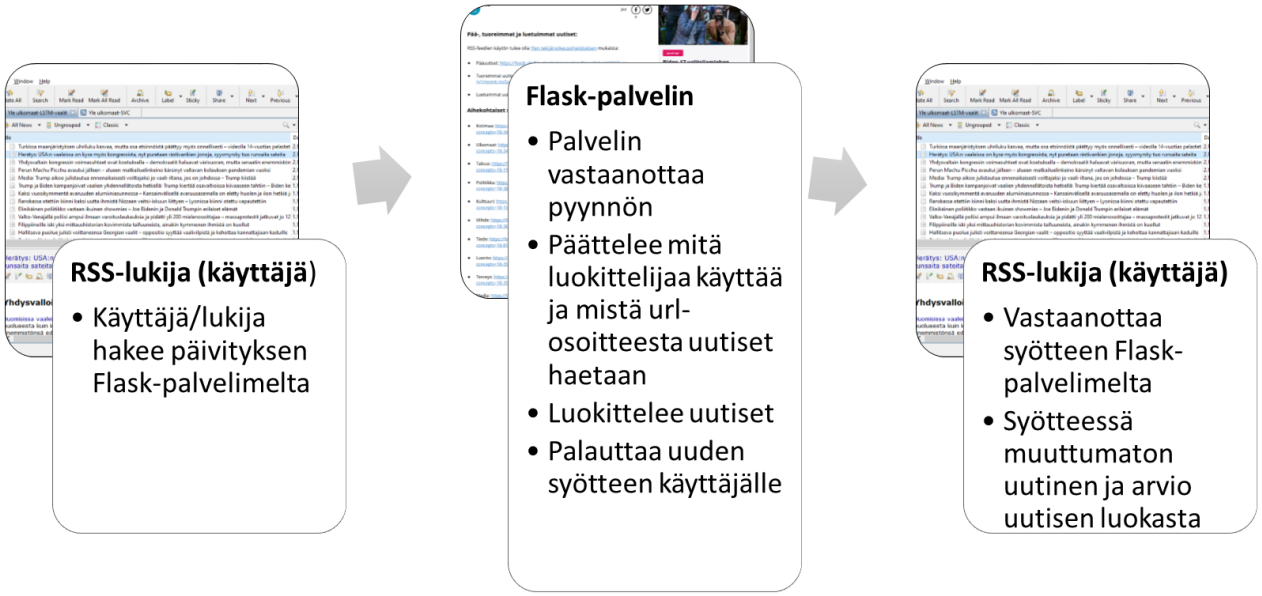
RSS-endpoint, eli url-osoite, josta käyttäjä pyytää luokittelu, muodostuu palvelimen ip-osoitteesta, staattisesta nimestä ja luokittelijan tyypistä. Esimerkiksi kysely osoitteeseen http://127.0.0.1:5000/yle-ulkomaat/LSTM_uusin palauttaa Ylen ulkomaanuutiset luokiteltuna LSTM_uusin-nimellä tallennetun luokittelijan arviolla. Esimerkki käyttäjän näkymästä on kuvassa 7.

Kuva 7. Esimerkki käyttäjälle näkyvästä tulosteesta RSSOwl-lukijassa. Category-sarakkeessa on esitetty käytetyn LSTM-luokittelijan arvio kyseisen uutisen kategoriasta.



Kuvassa 8 on esitetty valmiin järjestelmän toimintaperiaate. Valittu ratkaisu, jossa palvelin vastaanottaa get-metodin yhteydessä halutun luokittelijan nimen, mahdollistaa samojen uutislähteiden luokittelun useilla eri luokittelijoilla. Tällöin luokittelijoiden laatua voidaan verrata helposti.

Kuva 8. Uutisvirran luokittelun kokonaisprosessi, kun luokittelija on opetettu ja otettu käyttöön.



3.5 Koneoppimismallien tulokset

Eri malleja testattiin kahdella eri koulutusaineistolla. Ensimmäinen, laajempi, aineisto koostui 2225 BBC:n uutisesta, jotka oli luokiteltu viiteen eri kategoriaan (talous, viihde, politiikka, urheilu ja teknologia) (BBC, 2006). Aineisto oli englannin kielistä. Valmiin koulutusaineiston käyttö mahdollisti koneoppimismallien tulosten vertailun laajempaan joukkoon. Kaggle.com on koneoppimiseen ja tekoälyyn keskittynyt sivusto, jossa useat alan tutkijat ja harrastajat ovat julkaisseet omia menetelmiään muun muassa kyseisen koulutusaineiston luokittelusta. Aineistolla on saavutettu yli 90 % tarkkuuksia (Kaggle, 2019).

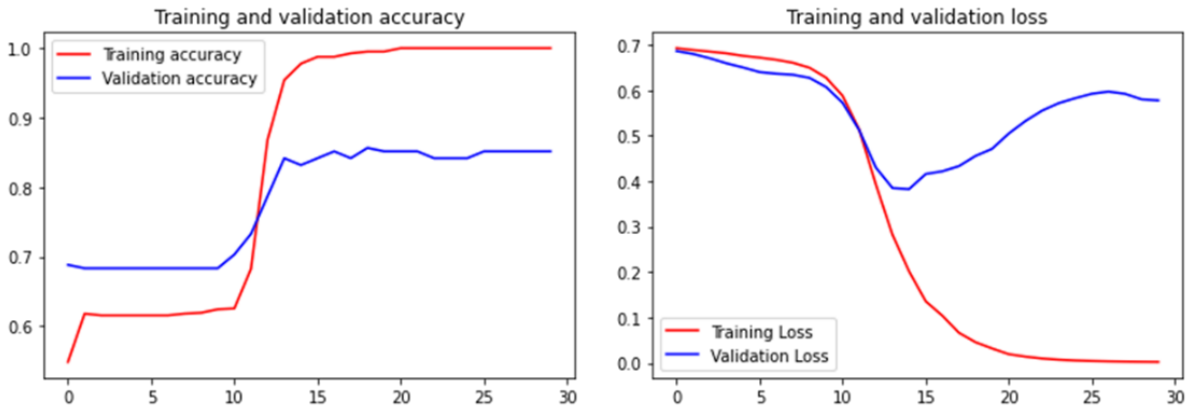
Toinen käytetty aineisto oli Ylen uutisista koostettu 504 suomenkielisen uutisen joukko, jossa oli uutisia eri aiheista. Aineisto jaettiin kahteen kategoriaan (vaalit n = 188 ja muut n = 316). Korona-pandemia vaikutti jonkin verran aineiston koostamiseen, koska uutisvirta painottui huomattavasti koronaa käsitteleviin aiheisiin.

Kaikkia menetelmiä testattiin molempia aineistoja vasten. Samalla testattiin, kuinka paljon uutisia täytyy syöttää luokittelijoille opetusvaiheessa, jotta tarkkuus pysyy yli 95 prosentissa.

Tuloksia tarkasteltiin määrällisesti ja laadullisesti. Koulutusaineisto jaettiin testi- ja koulutusaineistoihin. Kun malli oli koulutettu, tarkisteltiin tuloksia ensin tarkkuuden ja loss-funktion tulosten avulla. Jos näiden arvot vaikuttivat riittävän hyviltä, tallennettiin vielä testiaineisto erilliseen tiedostoon, jossa oli sekä luokittelijan arvio kyseisestä uutisesta, että sen todellinen luokka. Tällöin oli mahdollista tarkastella minkä tyyppisissä uutisissa luokittelija teki virheitä.

Kuvan 9 tilannetta lähemmin tarkastellen selvisi, että pääosin luokittelijan tekemät virheet tulivat sellaisista uutisista, joiden kohdalla ihminenkin tekisi virheitä. Tässä ajossa testiaineisto koostui 101 uutisesta, joista luokittelija luokitteli 15 väärin. Luokittelijalla oli vaikeuksia tunnistaa vaaleihin liittyviksi uutisiksi muun muassa pitkiä artikkeleita, joiden yhteys oli häilyvä, kuten Minskin mielenosoituksia käsitteleviä uutisia. Toisaalta se luokitteli vaaleihin liittyväksi presidentti Niinistön koiraa käsittelevän uutisen.

Kuva 9. Esimerkki Ylen aineistoon perustuvan LSTM-luokittelijan opetuksesta. Luokittelija on oppinut 100 % tarkkuudella opetusaineiston noin 20. epochin kohdalla, eivätkä tulokset enää parane tämän jälkeen.



Seuraavia huomioita syntyi LSTM-luokittelijasta testien aikana:

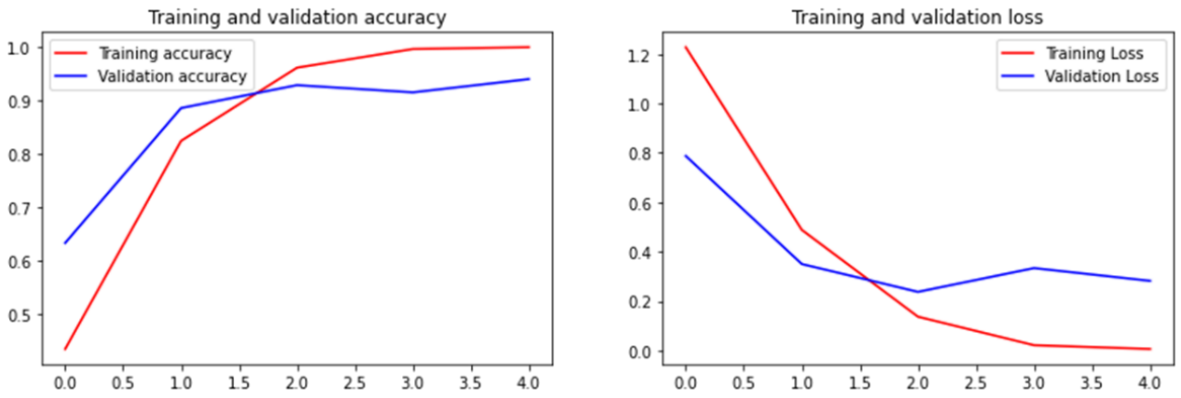
- Aineisto tulisi tasapainottaa ennen luokittelijan opetusta
- Testaaminen on hidasta, mutta vain yhtä muuttujaa tulisi muuttaa kerrallaan
- Neuroverkkojen testaamisessa epocheja tulisi olla aluksi riittävän paljon, jotta parametrien muutokset näkyvät paremmin

Paras saavutettu tarkkuus Ylen uutisten osalta saatiin Bi-LSTM-luokittelijalla. Kun koulutusaineistona käytettiin 403 uutista ja testaukseen 101, saatiin testausaineistossa tarkkuudeksi 0,9109. Jakson pituus oli 500, osajakson pituus 128 ja oppimisnopeus (learning rate) oli 0,0001. Kyseisen luokittelijan tarkkuus ja loss-funktion arvot löytyvät kuvasta 10.

SVC kykeni luokittelemaan Ylen uutiset testiaineistossa 0,901 tarkkuudella. SVC:n luokitteluvirheet olivat hyvin samanlaisia, kuin bi-LSTM:llä. Niin sanotusti tavallista, syvää, neuroverkkoa ei saatu toimimaan missään kohtaa hyvin tekstin luokittelussa ja tarkkuus jäi aina alle 70 %.

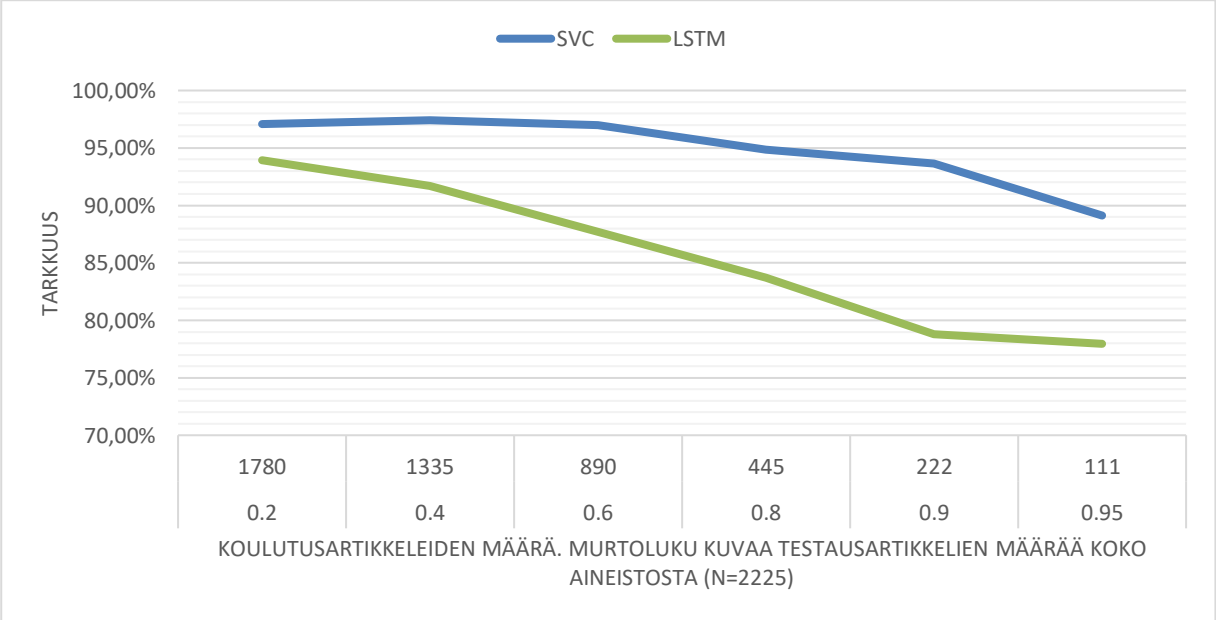
BBC:n testiaineisto on sanastoltaan hieman yksinkertaisempi kuin Ylen. BBC:n aineistoon jäi tekstin esikäsittelyn jälkeen 19726 yksilöllistä sanaa, kun taas Ylen aineistossa oli 24324 sanaa. SVC suoriutui BBC:n materiaalissa erinomaisesti. Optimoidun luokittelijan tarkkuus testimateriaalissa oli 0,9708, kun koulutukseen käytettiin 1780 uutista ja testaamiseen 445. Optimoidun SVC-luokittelijan parametrit olivat: {'C': 1.4, 'class_weight': 'balanced', 'kernel': 'sigmoid', 'tol': 0.01}. Bi-LSTM saavutti vastaan suuruisella testiaineistolla 0,9393 tarkkuuden. Jakson pituus oli 400, osajakson pituus 64 ja oppimisnopeus (learning rate) oli 0,001. Bi-LSTM:n tekemät virheet olivat saman tyyppisiä kuin Ylen uutistenkin suhteen. Esimerkiksi teknologiakategoriaan kuulunut uutinen käsitteli rugbyn analysointiin tarkoitettua ohjelmaa (BBC News 2005). Bi-LSTM luokitteli kyseisen uutisen urheiluksi. Luokittelijoiden tarkkuuden eri testi- ja koulutusaineistojen suhteilla löytyy kuvasta 11.

Kuva 10. Bi-LSTM-luokittelijan tarkkuus ja loss-funktio epochien mukaan BBC:n materiaalin luokittelussa.



Tarvittavan koulutusdatan arvioimiseksi BBC:n aineiston määrää vähennettiin ja luokittelijat opetettiin pienemmällä aineistolla. SVC-luokittelija optimoitiin jokaisella ajolla erikseen. Bi-LSTM:n osalta muutettiin vain epochien lukumäärä optimaaliseksi ja osajakson pituutta lyhennettiin pienimmällä koulutusdatan määrällä 32:een. Tulokset löytyvät kuvasta 11. Testi tehtiin niin, että kaikista 2225 artikkelista käytettiin osa luokittelijan koulutukseen ja loput sen testaamiseen. Eli esimerkiksi ensimmäisessä ajossa testaamiseen käytettiin 20 % artikkeleista ja 80 % kouluttamiseen. Näin olleen pienemmillä koulutusaineistoilla saatuja tuloksia voidaan pitää luotettavampina.

Kuva 11. Bi-LSTM- ja SVC-luokittelijan tarkkuus BBC:n aineistolla.



Suoritettujen testien perusteella vain SVC-luokittelija pääsi tavoiteltuun yli 95 % tarkkuuteen ja sekin vain BBC:n aineistossa. Bi-LSTM oli hieman tarkempi suomenkielisen aineiston luokittelussa, mutta ei merkittävästi. Hieman yllättäen SVC:n tarkkuus BBC:n aineiston osalta pysyi yli 95 % tasolla pitkään, vaikka koulutusaineiston määrää vähennettiin. 95 % tarkkuuteen riitti noin 660 artikkelin koulutusmäärä. Viiteen kategoriaan jaetun datan osalta tämä tarkoitta noin 132 artikkelia per kategoria. On kuitenkin huomattava, että Ylen uutisten osalta ei päästy samaan tarkkuuteen, vaikka uutisia oli käytössä enemmän opetukseen. Näyttää siltä, että tarkkuuden valossa monimutkaisen neuroverkon käyttö tekstinluokittelussa ei ole perusteltua, jos koulutusmateriaalia on alle 500 näytettä per luokka.

4 Johtopäätökset

Koneoppiminen on saavuttanut siinä määrin kypsän iän, että eri menetelmien käyttöön löytyy hyvin dokumentoitua kirjallisuutta ja toimivien järjestelmien rakentaminen voidaan tehdä jo olemassa olevia kirjastoja käyttäen. Tekstin luokittelussa päästään hyviin tarkkuuksiin, kun luokittelijoiden opetukseen on saatavilla tarpeeksi aineistoa.

Kuten useat aiemmat tutkimukset osoittavat, uutisten luokitteluun voidaan käyttää menestyksekkäästi useita erilaisia koneoppimismalleja (Zhou, ym., 2015). Valitun mallin lisäksi vähintään yhtä tärkeää on tekstin esikäsittely mallille soveltuvaksi. Mikäli tekstistä ei poisteta niin sanotusti kohinaa, on malleja vaikeampi kouluttaa. Toisaalta esikäsittelyssä on mahdollista poistaa liikaa informaatiota. Kuten Van Hulse, ym. (2007) osoittivat, satunnainen alinäytteistäminen näytti toimivan epätasapainossa olevien aineistojen tasapainottamiseen.

Käytännössä kaikilla yleisimmillä koneoppimismalleilla voidaan tehdä luonnollisen kielen luokittelua. Suhteellisen yksinkertaisetkin algoritmit toimivat kohtuullisesti, kun tekstin esikäsittely on tehty hyvin. Useilla kielillä toimivan Snowball-stemmerin käyttö mahdollistaa saman esikäsittelijän käytön helposti, mutta jokaiselle kielelle erikseen rakennettu stemmeri toimisi todennäköisesti paremmin.

LSTM:n tarkkuus laski BBC:n aineistossa nopeammin kuin SVC:n, kun koulutusaineiston määrää vähennettiin. Havainto on kuitenkin samansuuntainen kuin mitä Nam, Kim, Mencia, Gurevych ja Fürnkranz (2014) huomasivat omassa tutkimuksessaan. SVM:n (SVC on SVM:n sovellus) tarkkuus

laski heidän tutkimuksessaan, kun aineiston määrä nousi tietyn rajan yli ja ratkaisuksi he ehdottivat neuroverkkoja. On mahdollista, että tässä opinnäytetyössä käytetyt aineistot ovat niin pieniä, ettei bi-LSTM-luokittelijasta saatu vielä täyttä potentiaalia irti. Toisaalta SVC:n osalta ei saavutettu kriittistä pistettä, jossa tulokset alkavat laskea.

Uutisten luokittelu eri aiheisiin on valvottua koneoppimista, jossa ihminen luokittelee ensin opetusaineiston ja malli pyrkii oppimaan, miten aineisto on luokiteltu. Mikäli luokkia ei määritellä erittäin tarkasti, muodostuu koulutusaineisto helposti erittäin moniulotteiseksi. Esimerkiksi Suomen eduskuntavaaleista käsittelevien uutisten erottaminen Yhdysvaltojen presidentinvaaleja käsittelevistä vaatii enemmän aineistoa, kuin yleisellä tasolla vaaleja käsittelevien uutisten erottaminen vaikkapa ilmastonmuutosta käsittelevistä. Lisäksi laajojen artikkelien osalta ongelmaksi voi muodostua se, että itse artikkeli käsittelee lopulta useita eri osa-alueita.

Linjanveto milloin uutisia kannattaa suodattaa tekoälyn ja milloin hakusanojen kanssa, ei ole täysin yksiselitteinen. Esimerkiksi vaaleja käsittelevien uutisten löytäminen uutisvirrasta muutamalla hakusanalla on erittäin nopeaa. Toisaalta tilanne muuttuu nopeasti, jos aihealuetta halutaan rajata tarkemmin. Esimerkiksi jos haetaan uutisia henkilöautojen päästöjen vaikutuksesta ilmastonmuutokseen Euroopassa, muuttuu hakusanalause helposti hyvin pitkäksi ja toisaalta se saattaa sulkea tahattomasti joitain uutisia pois. Tällöin tekoäly alkaa olemaan jo varteenotettava vaihtoehto.

Neuroverkot kykenevät käsittelemään paremmin suuria koulutusaineistoja ja löytämään niistä hienojakoisempia yhteyksiä. Toisaalta niiden kouluttaminen ja rakentaminen on huomattavasti haastavampaa kuin algoritmien käyttäminen. Vaikka erilaisia hyvin dokumentoituja kirjastoja on runsaasti, täytyy valittu neuroverkko optimoida kyseiseen ongelmaan. Neuroverkkojen ongelmina ovat niiden vaatima laskentateho ja tulkinnan haasteellisuus. Epäsuorasti molemmat ongelmat liittyvät toisiinsa. Koska parametreja on runsaasti ja neuroverkkojen kouluttaminen vaatii runsaasti laskentatehoa, on kaikkien parametrien optimointi vaikeaa tai käytännöllisesti katsoen mahdotonta.

Opinnäytetyön toteuttamiseen varatusta ajasta ehdottomasti suurin osa meni bi-LSTM-neuroverkon muodostamiseen ja erilaisten parametrien kokeilemiseen. Teoria ohjasi nopeasti kokeilemaan syvää bi-LSTM-rakennetta, mutta parametrien osalta tutkimustietoa ei ole juurikaan

saatavilla. Kirjallisuudesta ei löytynyt juurikaan suuntaviivoja kuinka pitkiä sekvenssejä LSTM kykenee käsittelemään luotettavasti. Eri lähteiden arvioit pyörivät muutamasta sadasta aina viiteen sataan asti. Havaintojen mukaan kuitenkin jopa yli 600 sanan sekvenssit toimivat. Sekvenssien pituuden valinnassa täytyy huomioida käytössä oleva aineisto. Tekstin esikäsittely lyhentää artikkeleita karkeasti noin 10 % eivätkä yli 1000 sanan artikkelit ole poikkeuksellisia.

Koneoppimisen implementoiminen osaksi uutisvirran suodatusta on periaatteessa suhteellisen helppo tehtävä. Valittu menetelmä, RSS-syötteen kierrättäminen Flask-alustalla toimivan luokittelijan kautta, oli suhteellisen helppo toteuttaa. Nykyaikaisempi ratkaisu olisi toteuttaa luokittelija REST-tyyppisenä. Tällöin käyttäjän palautteen lukeminenkin olisi helppoa ja malleja voitaisiin hienosäätää ja uudelleen opettaa. Toisaalta on mahdollista, että käyttäjän mieltymyksien seuraaminen ei olisi erityisen helppoa. Kuten huomattiin, yksiselitteisen ja selkeän koulutusaineiston luominen ei ole erityisen helppoa.

Koneoppimismallien testaamiseen käytetyt aineistot olivat siinä määrin pieniä, että sellaista pistettä ei löytynyt, jossa Bi-LSTM olisi ollut parempi, kuin SVC, vaikka näin pitäisikin tapahtua teorian mukaan. Jatkotutkimusaiheena voisikin olla tutkia, miten paljon tekstiaineistoja tarvitaan, ennen kuin neuroverkot ohittavat tarkkuudessa algoritmit.

Lähteet

Aggarwal, C. C. (2018). Machine Learning for Text. *Springer International Publishing AG*.

Aly, M. (2005). Survey on Multiclass Classification Methods. *Caltech*.

BBC News (3.2.2005). *Piero gives rugby new perspective.* BBC News. http://news.bbc.co.uk/2/hi/technology/4229695.stm

BBC (2006). BBC Datasets. Haettu 3.9.2020 osoitteesta http://mlg.ucd.ie/datasets/bbc.html

Bell, J. (2015). Machine learning: hands-on for developers and technical professionals. *Indianapolis, Indiana: Wiley.*

Cambridge University Press. (2009). *Tf-idf weighting*. Haettu 2.10.2020 osoitteesta https://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html

DeepAi. (n.d.). *What is ReLu?* Haettu 4.10.2020 osoitteesta https://deepai.org/machine-learning-glossary-and-terms/relu

Finnish stemming algorithm. (n.d.). Haettu 15.3.2020 osoitteesta https://snowballstem.org/algorithms/finnish/stemmer.html

Google. (2020). *Machine Learning Crash Course.* Haettu 15.3.2020 osoitteesta https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

Graves, A., Fernandez, S., & Schmidhuber, J. (2005). Bidirectional LSTM Networks for Improved PhonemeClassification and Recognition. Berliini: Springer. Haettu 2.10.2020 osoitteesta https://mediatum.ub.tum.de/doc/1290195/file.pdf

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012). Improving neural networks by preventingco-adaptation of feature detectors. Toronto: University of Toronto.

Hironsan. (n.d.). *Doccano demo.* Haettu 4.10.2020 osoitteesta http://doccano.herokuapp.com/demo/text-classification/

Van Hulse, J., Khoshgoftaar, T. M. & Napolitano, M. (2007). Experimental Perspectives on Learning from Imbalanced Data. Florida: Florida Atlantic University.

Kaggle. (2019). *Aryan BBC News Classification.* Haettu 2.10.2020 osoitteesta https://www.kaggle.com/aryankaul31/aryan-bbc-news-classification

Kettunen, K. & Kunttu, T. & Järvelin, K. (2005). To stem or lemmatize a highly inflectional language in a probabilistic IR environment? Journal of Documentation. http://dx.doi.org/10.1108/00220410510607480

Kowsari, K. M. (2019). Text Classification Algorithms: A Survey. Arxciv. https://arxiv.org/pdf/1904.08067.pdf

Madjarov, G. K. (2012). An extensive experimental comparison of methods for multi-label learning. https://www.sciencedirect.com/science/article/abs/pii/S0031320312001203?via%3Dihub

Nam, J., Kim, J., Mencia, E. L., Gurevych, I. & Fürnkranz, J. (2014). Large-Scale Multi-label Text Classification — Revisiting Neural Networks. Berliini: Springer. https://link.springer.com/chapter/10.1007/978-3-662-44851-9_28

NLTK Project. (2020). *Natural Language Toolkit*. Haettu 4.10.2020 osoitteesta http://www.nltk.org/

Schuster, M. & Kuldip, P. K. (1997). Bidirectional recurrent neural networks. IEEE. Haettu 26.9.2020 osoitteesta https://www.researchgate.net/publication/3316656_Bidirectional_recurrent_neural_networks/link/56861d4008ae19758395f85c/download

Yle. (2020). *Yle pääuutiset*. Haettu 4.10.2020 https://feeds.yle.fi/uutiset/v1/majorHeadlines/YLE_UUTISET.rss

Zhou, C., Sun, C., Liu, Z. & Lau, F. C. M. (2015). A C-LSTM Neural Network for Text Classification. arXiv.org. https://arxiv.org/abs/1511.08630