jamk.fi

Informaatiovaikuttamiseen AI-botteja RGCE:hen

Joonas Lähteinen

Opinnäytetyö
Toukokuu 2019
Tekniikan ja liikenteen ala
Insinööri (AMK), ohjelmistotekniikka tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Lähteinen, Joonas	Opinnäytetyö, AMK	Toukokuu 2019
	Sivumäärä 64	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Informaatiovaikuttamiseen AI-botteja RGCE:hen

Tutkinto-ohjelma
Ohjelmistotekniikka

Työn ohjaaja(t)
Mika Rantonen, Antti Häkkinen

Toimeksiantaja(t)
JYVSECTEC

Tiivistelmä

Tutkimuksessa perehdyttiin koneoppimiseen ja sen hyödyntämiseen luonnollista kieltä ymmärtävien bottien toteuttamisessa. Tavoitteena oli saada luotua botti, jonka täytyi pystyä tuottamaan informaatiovaikuttamista haluttaessa. Botin täytyi myös ymmärtää suomen että englannin kieltä. Samalla tutkimuksessa tutustuttiin erilaisiin koneoppimismenetelmiin sekä näiden hyödyntämiseen reaalimaailmassa.

Lopputuloksena saatiin toteutettua RGCE-ympäristöön soveltuva chattibotti, joka pystyi toimimaan itsenäisesti ilman erillisiä palveluita. Chattibotti pystyi vastaamaan GNU social-palveluun tuleviin viesteihin sekä ymmärtämään sekä englannin että suomenkielisiä viestejä. Kielenkääntämisessä botti käänsi suomenkielisen tekstin ensin englannin kielelle, jonka avulla se loi vastauksen haettuun viestiin. Tämän jälkeen botti käänsi englanninkielisen vastauksen suomenkieliseksi, jonka se lähetti vastaukseksi haettuun viestiin. Botti käytti omaa API-kirjastoaan kommunikoimiseen GNU social-palvelussa.

Tutkimuksesta saatiin hyvää tietoa, mitä omien chattibottien tekemiseen tarvitaan sekä millaista datajoukkoa botin kouluttamiseen vaaditaan. Lisäksi huomattiin, että varsinkin datajoukon laadulla on suuri merkitys botin älylliseen toimintaan ja sen luonnollisen kielen ymmärtämiseen.

Avainsanat (asiasanat)
Koneoppiminen, Neuroverkot, Botti

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Lähteinen, Joonas	Bachelor's thesis	May 2019
		Language of publication: Finnish
	Number of pages: 64	Permission for web publication: x

Title of publication
information influencing bots to RGCE

Degree programme
Software Engineering

Supervisor(s)
Rantonen, Mika; Häkkinen, Antti

Assigned by
JYVSECTEC

Abstract

The objective of the thesis was to explore various machine-learning techniques and their use in chatbots that can understand natural languages. The bot was able to produce propaganda information if required and understand both Finnish and English languages. Additionally, the research was to gain a better understanding how different machine-learning techniques are used in real world applications.

The research resulted in chatbot that was able to operate independently without separate services in RGCE. The chatbot was able to respond to messages in GNU social server and understand English and Finnish languages. The bot translated a Finnish message first into English and after that, it used the English translation to create its own response to the original message. Next the bot translated the response message into Finnish before it sent the response to a server. The bot used its own API library when it communicated with Gnu social server.

The result of the research provided useful information on what it takes to create chatbots and what kind of datasets are needed to train chatbots. It was discovered that especially the quality of the dataset is very critical when creating intelligent chatbots that can understand natural language.

Keywords/tags (subjects)
Machine learning, Neural network, bot

Miscellaneous (Confidential information)

Sisältö

Kuviot

Sanasto

Botti	Tietokoneen hallinnoima ohjelma, joka pyrkii simuloimaan ihmistä
GloVe	Global Vectors for Word Representation on kokoelma sanavektoreja, joita voidaan hyödyntää neuroverkoissa
Koneoppiminen	Tietokoneella tehtävä tekoälyn luominen, jossa kone opetetaan tekemään omia ratkaisuja.
LSTM	Long short-term memory on yksi neuroverkon solutyypeistä, jossa koneopittu tieto säilytetään
Neuroverkot	Neuroverkko on ketju matemaattisia funktioita, joilla kone pyrkii tunnistamaan datasta yhtäläisyyksiä
RGCE	Realistic Global Cyber Environment on internettiä sekä sen palveluja simuloiva harjoitusympäristö
RNN	Recurrent Neural Network on rekursiivinen neuroverkko, jossa tieto kulkee rekursiivisesti solusta toiseen
Tekoäly	Keinotekoinen äly, jolla pyritään simuloimaan ihmismäistä ratkaisu/toiminallisuuskykyä.
Virtualisointi	Tietokoneen fyysisien laitteiden jakaminen useammaksi loogisiksi resursseiksi

1 Johdanto ja työn lähtökohdat

1.1 Johdanto informaatiovaikuttamiseen tekoälyllä

Opinnäytetyössä perehdyttiin tekoälyn hyödyntämiseen bottien tekemisessä. Tarkoituksena oli toteuttaa botti, joka kykenee tuottamaan informaatiovaikuttamisen tapaista tekstiä omatoimisesti ilman ulkopuolisia palveluita. Työssä perehdyttiin bottien toimintaan ja näiden toteuttamiseen käyttäen tekoälykirjastoja sekä selvitettiin yleisimpiä koneoppimisen toteutusmalleja, joilla voidaan toteuttaa luonnollista kieltä ymmärtäviä botteja. Tutkimuksessa informaatiovaikuttamisella tarkoitetaan tapaa, jolla botti saadaan tuottamaan hallitusti valheellista tietoa tai tuottamaan negatiivista kommentteja jostain asiasta.

1.2 Toimeksiantaja

Toimeksiantajana toimi Jyväskylä Security Technology, joka on itsenäinen kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus. Jyväskylä Security Technology toimii osana Jyväskylän ammattikorkean IT-instituuttia. Jyväskylä Security Technologyn osaamisalueita ovat kyberturvallisuus, tietoturvapoikkeamien hallinta, kehittyvät teknologiat ja informaatioteknologiat. (About Us n.d.)

1.3 Tavoite

Tutkimuksen tavoitteena oli saada tehtyä itsenäisesti toimiva chattibotti, joka osaisi tuottaa informaatiovaikuttavaa tekstiä. Viestialustana toimi avoimenlähdekoodiin GNU social-ohjelmisto, jossa botin täytyi pystyä reagoimaan tuleviin viesteihin sekä vastaamaan näihin omatoimisesti. Tämän lisäksi botin vastausintoa tuleviin viesteihin haluttiin pystyä säätämään halutun mukaiseksi, kuten botti reagoisi vain tiettyihin avainsanoihin tai botti vastailisi vain tietyn viestimäärän jälkeen. Botin haluttiin myös osavan reagoida sekä suomenkielisiin että englanninkielisiin viesteihin. Mikäli viesti annettiin suomeksi, botin täytyi tällöin myös osata antaa oma viestinsä suomenkielisessä muodossa. Näin ollen botille jouduttiin toteuttamaan oma kielikääntäjä käyttäen neuroverkkoa, joka ei tukeutuisi ulkopuolisiin palveluihin.

Toimeksianto määritteli, että botin täytyisi pystyä toimimaan RGCE-ympäristössä, josta ei ole internetyhteyttä ulkopuoliseen verkkoon. Näin ollen botin ohjelmisto täytyi suunnitella niin, että botti olisi täysin omatoiminen eikä se olisi riippuvainen ulkopuolisista palveluista tulkittaessa ja vastattaessa viesteihin. Muutoin käytetty ohjelmointikieli sekä botin toiminnallisuuden toteutustapa oli vapaasti valittavissa, kunhan vaaditut ehdot täyttyivät. Toimeksiannossa ei määritelty käytettäviä datajoukkoja, joita olisi tarkoitus käyttää botin koulutukseen. Näin ollen datajoukkoja ladattiin Kaggle-nimisestä palvelusta, josta löytyy eri tyyppisiä ja valmiiksi kasattuja datapaketteja erilaisista tietolähteistä. Datajoukkoja koottiin ja muokattiin niin, että lopputuloksena koettiin saavan tarpeeksi sisältöä botin koulutukseen. Datajoukko täytyi saada kasattua monipuoliseksi, jotta botti saataisiin vastaamaan selkeästi ja osaisi ymmärtää sekä suomenkielistä että englanninkielistä tekstiä.

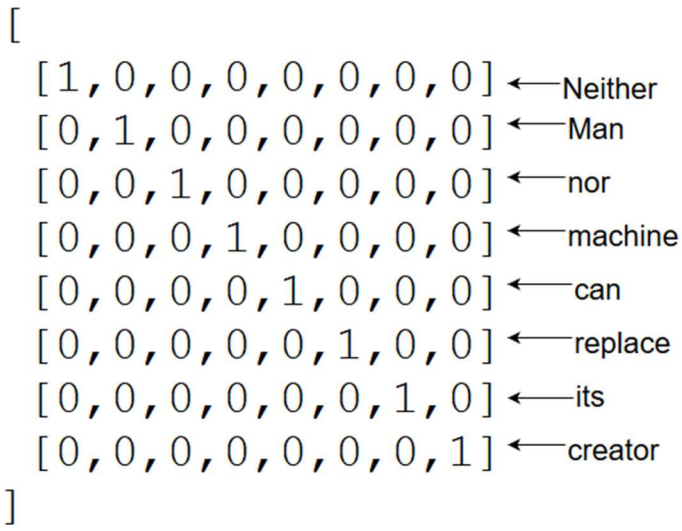
2 Sanavektorit

2.1 One-hot

One-hot-enkoodaus on tapa, jolla datan ominaisuudet saadaan esitettyä numeraalisessa muodossa. Datan ominaisuudella tarkoitetaan datasta löytyviä datapisteitä, joiden avulla voidaan ennustaa tuloksia. Esimerkiksi chattiboteista puhuttaessa koulutusdatasta löytyviä ominaisuuksia ovat sanat tai kirjaimet, joita botti käyttää lauseen ymmärtämiseen ja tulosteen luomiseen. Tämä riippuu, siitä käyttääkö botti sanapohjaista menetelmää vaiko kirjainpohjaista menetelmää.

Oletetaan, että botti käyttää sanapohjaista menetelmää, jossa siis jokainen sana on oma erillinen ominaisuus. One-hot-enkoodattu vektorin pituus on ominaisuuksien yhteenlaskettu määrä. Mikäli botilla on sanavarasto, joka koostuu sadasta eri sanasta, tällöin one-hot-enkoodatun vektorin pituus olisi siis sata erillistä alkiota. One-hot-vektorin alkioiden arvojen esitysmuoto on binaarimuotoista eli nolla tai ykkönen.

Esimerkkilause ”Neither Man nor machine can replace its creator” muodostaa kahdeksanulotteisen taulukon. Mikäli sanavarastoon ei kuuluisi muuta kuin nämä kahdeksan sanaa, olisi tämä lause one-hot-enkoodattuna Kuvio 1 mukainen.

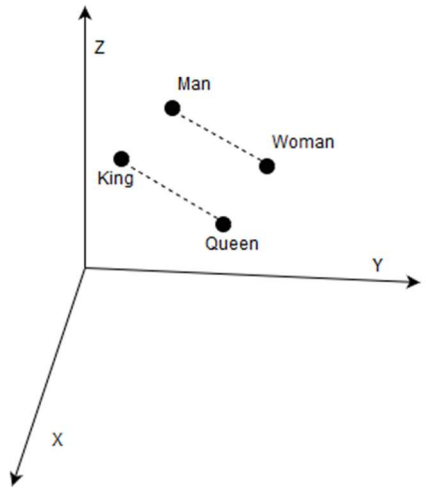


Kuvio 1. Lause one-hot-enkoodattuna, jossa sanavarasto on kahdeksan sanaa

Mikäli one-hot-menetelmää käytettäisiin chattiboteissa, jotka käyttävät sanapohjaista menetelmää, koulutusdatan muuntaminen one-hot-enkoodatuiksi vektoreiksi muodoistasi ongelman, jossa vektorit kasvaisivat valtavan suuriksi. Mikäli sanottaisiin, että chattibotti osaa tuhat eri sanaa, olisi koulutusdatan vektorin pituus tällöin tuhat alkiota ja taulukon ulottuvuus määräytyisi lauseen pituudesta. Näin ollen one-hot-enkoodattu esitysmuoto pitäisi sisällään paljon turhaa dataa, sillä suurin osa arvoista olisi nollia. (One-Hot Encoding and Binning 2018.)

2.2 Embedding

Koneoppimisessa embedding mahdollistaa datan esittämisen pienemmässä muodossa. Siinä missä one-hot-enkoodattu sanavektorin pituus muodostuu sanaston koosta, embedding koostaa vektorin määritellyn pituuden mukaan. Embedding-sanavektorien arvot päivittyvät neuroverkon koulutuksen yhteydessä. Embedding auttaa myös samalla tunnistamaan samankaltaisia sanoja. Koska embedding-kerros tuottaa sanoista sanavektoreita, voidaan näin ollen sanoista luoda kolmiulotteinen sanakartta (ks. Kuvio 2), josta sanojen etäisyys toisistaan voidaan laskea.



Kuvio 2. Sanat aseteltuna kolmiulotteiseen koordinaatistoon

Myös samankaltaiset sanat ovat tässä sanakartassa lähempänä toisiaan. Tämä auttaa paremmin ymmärtämään sanojen merkitystä lauseessa. Esimerkkinä sanat kuningas ja mies esiintyisivät lähempänä toisiaan verrattuna sanoihin kuningatar ja nainen. Embedding myös auttaa ymmärtämään sanojen taivutuksia, jolloin perusmuodossa oleva sana löytyy kolmiulotteisessa sanakartasta hyvin läheltä taivutusmuodossa olevaa sanaa. (Koehrsen 2018.)

3 Koneoppiminen

3.1 Tekoäly koneoppimisessa

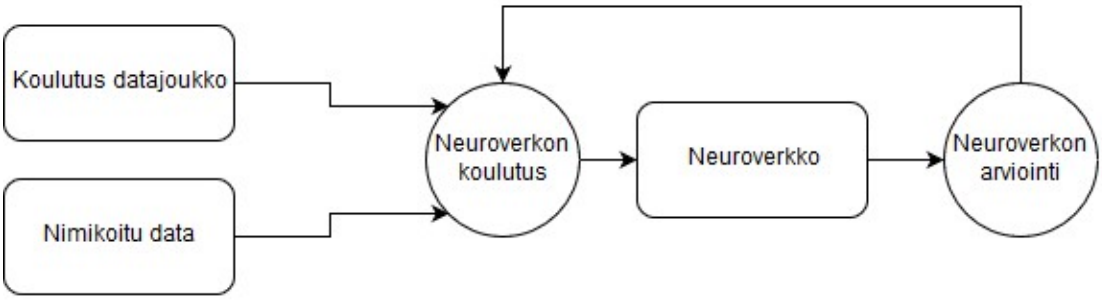
Tekoälyllä koneoppimisessa tarkoitetaan menetelmää, jolloin kone oppii itsenäisesti algoritmien avulla havainnoimaan ja poimimaan tietynlaisia kuvioita tai asioita annetusta tietolähdemassasta ilman, että koneelle on ohjelmallisesti kerrottu tarkasti mitä etsiä. Nykypäivinä tekoälyä on mahdollista hyödyntää monipuolisesti useissa eri aihealueissa tai osana palveluja. Tekoälyllä on mahdollista luoda esimerkiksi kuvantunnistuksia, jossa kone osaa koulutetun tiedon perusteella kertoa, mitä kuva sisältää. Kuvan luominen ja muokkaaminen on mahdollista toteuttaa käyttäen tekoälyä, jossa kone itsenäisesti osaa muokata kuvaa esimerkiksi halutun teeman mukaisesti

tai luoda kuvia itsenäisesti. Myös tekstin tuottaminen ja analysointi on yksi tekoälyä hyödyntävistä aiheista, jossa kone pystyy tuottamaan oikeaoppista tekstiä kuten arvosteluja tai keskusteluviestejä. Myös tekstin analysointi kuten propagandan tai vihapuheen tunnistaminen on mahdollista toteuttaa käyttäen nykyisiä tekoälytekniikoita. (Barack Obama is the Benchmark for Fake Lip-Sync 2018.)

3.2 Koneoppimistyypit

3.2.1 Ohjattu oppiminen

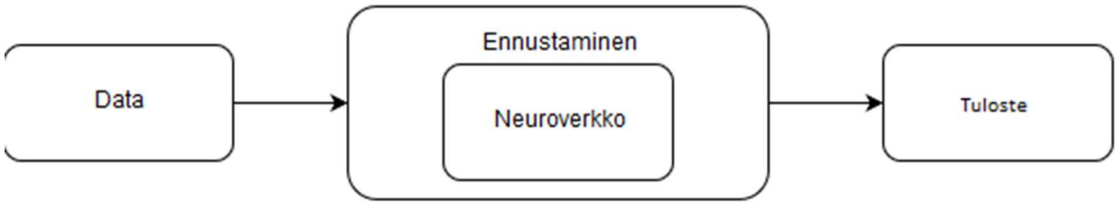
Ohjattu oppiminen on yksi koneoppimisen menetelmä, jossa annettu data on ennalta nimikoitua. Nimikoidulla datalla ohjatussa oppimisessa tarkoitetaan datarakennetta, jossa neuroverkkoa koulutettaessa annettavalle syötteelle on tiedossa oikea tai haluttu lopputulos. Näin ollen ohjatun oppimisen neuroverkkoa opetettaessa kone luo opetusdatan perusteella painotetun verkon, jossa eri syötteille annetaan eri painoarvoja perustuen näiden saman- tai erikaltaisuuteen ennalta määriteltyjen lopputulosten välillä (ks. Kuvio 3). Ohjattu oppiminen soveltuu hyvin esimerkiksi kuvantunnistukseen, jossa neuroverkko osaa tunnistaa asioita tai eliöitä kuvista.



Kuvio 3. Ohjatun oppimisen koulutus logiikka

Koska ohjatussa oppimisessa neuroverkon koulutukseen tarvitaan valmiiksi nimikoitua datajoukkoa, täytyy datan yleisimmin olla valmiiksi käsiteltyä, tai data on luotu käyttäen muita koneoppimismenetelmiä hyödyntäen kuten esimerkiksi vahvistettua

oppimista (ks. luku 3.2.3). Ohjattua oppimista käyttävän neuroverkon koulutus aloitetaan rakentamalla koulutusdata, jossa siis nimikoidaan koulutusdata. Chattibo-teissa tämä tarkoittaisi sitä, että koulutusdata on syötettävä teksti ja nimikoitu data tälle olisi vastaus kyseiseen viestiin. Sen jälkeen kun koulutusdata on kasattu ja nimi-koitu, se syötetään neuroverkolle. Neuroverkko suorittaa koulutusjakson, jonka jäl-keen neuroverkkoa testataan, ja mikäli se saavuttaa halutun tarkkuuden tai koulutus-kerta määrä saavutetaan, voidaan koulutus lopettaa. Neuroverkkoa käytettäessä uusi tuntematon data syötetään neuroverkolle, joka tulkitsee syötteen aiemmin opitun tiedon pohjalta (Kuvio 4). Tämän jälkeen neuroverkko tuottaa vastauksen annetulle syötteelle. (Ivan 2015.)



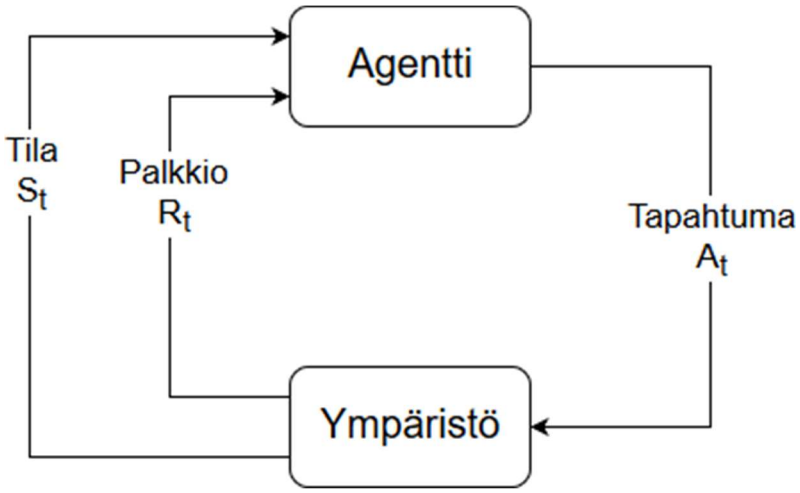
Kuvio 4. Ohjatun oppimisen neuroverkon käyttäminen

3.2.2 Puoliohjattu oppiminen

Puoliohjatussa oppimisessa koulutusdatana käytetään sekä nimikoitua että myös ei-nimikoitua datajoukkoa. Puoliohjattu neuroverkko pyrkii nimikoidun datan perus-teella kouluttamaan verkon niin, että se pystyy nimikoimaan dataa, joissa lopputu-losta ei tiedetä. Tällaista menetelmää voidaan esimerkiksi käyttää objektin tunnistuk-sessa, jossa neuroverkko opetetaan pienellä määrällä datajoukkoa tunnistamaan eri objekteja. Tämän jälkeen koulutettua verkkoa käytetään nimikoimaan loppu tunte-maton datajoukko ja näin koko datasta saadaan luotua täysin nimikoitua. (Shubham 2017.)

3.2.3 Vahvistettu oppiminen

Vahvistetussa oppimisessa ajatus on löytää dataan perustuva paras mahdollinen ratkaisu, joka ratkaisisi käsiteltyä ongelmaa. Vahvistetussa oppimisessa käytetty data ei ole ohjatun oppimisen tapaan nimikoitua, sillä data ei sisällä tietoa mikä on haluttu lopputulos. Näin ollen vahvistettua oppimista käyttävä neuroverkko joutuu siis kokeilemaan ratkaisuja ja painottamaan sisäisesti toimenpiteet, joilla verkko pääsi lopputulokseen. Vahvistettua oppimista käyttäviä neuroverkkoja käytetään yleisimmin pelien bottien tekemiseen. Vahvistettu oppiminen toimii pelibottien tapauksissa hyvin, sillä yleisimmin peleissä ei ole yhtä ainutta oikeaa vaihtoehtoa, joilla pääsee haluttuun lopputulokseen kuten esimerkiksi pelin kentän loppuun. Näin ollen neuroverkolle kelpaa koulutusdatajoukoksi ohjaimen napin painallukset, joilla botti pääsee etenemään pelissä (ks. Kuvio 5). Neuroverkko painottaa verkon esimerkiksi kentän läpäisynopeuden perusteella, jossa näppäinkombinaatiot ovat koulutusdatajoukko. (A Beginner's Guide to Deep Reinforcement Learning n.d)



Kuvio 5. Vahvistetun oppimisen yksinkertaistettu logiikka

Vahvistetussa oppimisessa agentti on asia, joka oppii ympäristöstään suorittamalla erilaisia toimintoja. Agentti halutaan saada optimoitua niin, että toiminnot olisivat ympäristön kannalta kaikista parhainta ratkaisuja. Toiminto on tapahtumia, joita agentti voi suorittaa ympäristössä. Ympäristö on tila, jossa agentti suorittaa

toimintoja ja ympäristö määrittelee agentin toiminnon paremmuuden muihin mahdollisiin toimintoihin. Palkkio on ympäristön antama arvo, jota agentti käyttää hyväksi määritellessään toiminnon paremmuuksia. Mitä parempi toiminto oli ympäristön kannalta, sitä suurempi palkkio palautetaan agentille. Tila on tapahtumien jälkeinen agentin tila. (Kurama 2018.)

Esimerkkinä peleissä agentti olisi pelihahmo, jota ohjataan pelissä. Toiminto on lista mahdollisista toiminnoista, joita pelihahmo voi suorittaa ympäristössään. Ympäristö on pelissä tietenkin pelin oma pelattava ympäristö, jossa pelihahmo pystyy liikkumaan ja toimimaan. Palkinto on pelissä toimintojen järkevyys pelin kannalta kuten esimerkiksi, onnistuiko pelihahmo saamaan halutun tavaran pelikentästä. Tila on taas paikka pelin ympäristössä, johon pelihahmo päätyi toimintojen jälkeen.

3.2.4 Ohjaamaton oppiminen

Ohjaamatonta oppimista käytetään tapauksissa, jossa datasta halutaan löytää yhtäläisyyksiä tai tietynlaisia rakenteita, jotka ovat vielä alussa tuntemattomia, tai ei tiedetä vielä, mikä tasan tarkkaan on haluttu lopputulos. Ohjaamattomassa oppimisessa myös syötetty data on yleisimmin tuntematonta eikä ole kovin käsiteltyä tai organisoitua. Näin ollen koneen on itse opittava tunnistamaan hyödylliset rakenteet ja yhtäläisyydet. Ohjaamatonta oppimista voidaan hyödyntää esimerkiksi kuvantunnistuksessa, jossa pyritään kategorisoimaan kuvat ja näin ollen helpottamaan esimerkiksi kuvien nimikoimista, joita voidaan käyttää sittemmin ohjatussa oppimisessa. (Van Loon 2018.)

3.3 Chattibotin periaatteet koneoppimisessa

Kun halutaan tehdä itsenäisesti toimiva chattibotti, joka ymmärtää luonnollista kieltä, täytyy ensimmäiseksi miettiä, mistä botin koulutusdata saadaan hankittua. Datan rakenne täytyisi myös olla tarpeeksi monipuolinen, jotta botti ymmärtää myös monimutkaisia lauserakenteita. Toiseksi tulisi miettiä, kuinka botti saadaan ymmärtämään tekstimuotoista dataa, ja botti myös osaisi itse tuottaa tätä, sillä tietokone

eivätkä koneoppimiskirjastot ymmärrä suoraan tekstimuotoista dataa, vaan tämä täytyy ensin muuttaa koneen ymmärtämään muotoon.

Chattibotteja tehdessä tekstimuotoinen datajoukko on tarkoitus ensin muuntaa numeraaliseen muotoon, jotta kone pystyy käsittelemään näin dataa matemaattisin menetelmin. Yleisimpiä tapoja on luoda lista jokaisesta halutusta merkistä, mitä halutaan käyttää luonnollisen kielen ymmärtämisessä, ja antaa jokaiselle merkille uniikki numeraalinen arvo. Toinen tapa on vastaava, miten merkkilistausmenetelmä toimii, mutta merkkien sijasta luodaan lista kokonaisista sanoista. Sanojen listausmenetelmä on huomattavasti raskaampi ja vaatii enemmän laskentatehoa kuin merkkipohjainen menetelmä, sillä esimerkiksi suomenkielessä on vain 29 eri kirjainta, jotka tarvitsee tallentaa muistiin. Näistä 29 merkistä pystytään koostamaan mitä tahansa suomenkielisiä sanoja, kun taas sanapohjaisessa ratkaisussa teoreettisesti jouduttaisiin ensin tallentamaan muistiin kaikki sanat taivutusmuotoineen (ks. Kuvio 6). Tämä taas vaatisi valtavan määrän muistia sekä laskentatehoa.

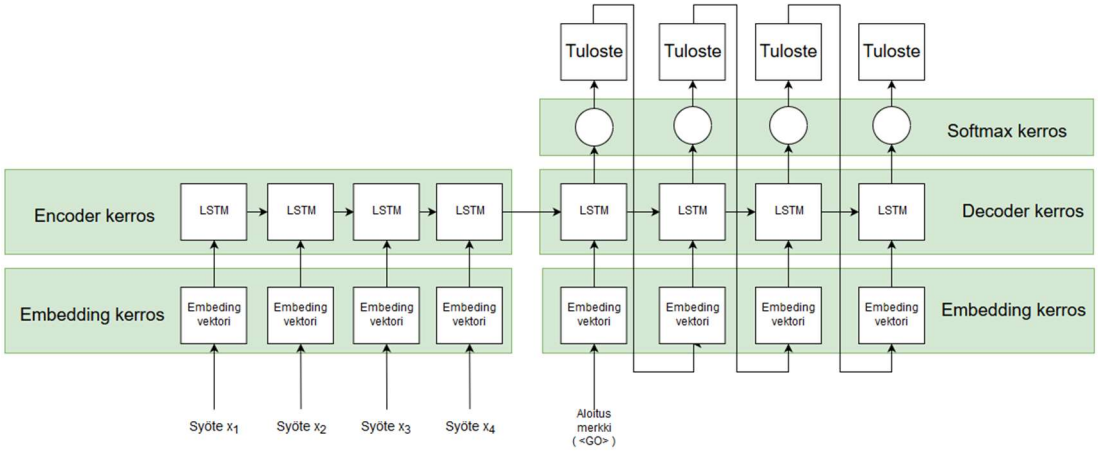
```
1. Vocabulary = (the: 1, quick: 2, brown: 3, fox: 4, jumps: 5, over: 6, lazy: 7, dog: 8 … )
2. // Lause: the quick brown fox jumps over the lazy dog
3. sequences = [1, 2, 3, 4, 5, 6, 1, 7, 8]
```

Kuvio 6. Lauseen muuntaminen numeraaliseen muotoon

Sanapohjaisen ratkaisun etuna on kuitenkin parempi lauseen ymmärtäminen sekä itse lauseiden rakenteen ymmärtäminen luomalla vektoreita sanoista. Näin ollen sanoista saadaan luotua kolmiulotteinen sanakartta, jossa sanojen etäisyys voidaan laskea toisistaan ja jossa samantyyppiset sanat löytyvät lähempänä toisiaan.

Kun teksti on saatu numeraaliseen muotoon, voidaan tämän jälkeen data syöttää halutulle neuroverkolle. Yksi yleisin tapa on käyttää niin sanottua sequence to sequence-mallia, jossa neuroverkko koostuu kahdesta recurrent neural network (RNN) -neuroverkosta. Näitä kahta RNN-neuroverkkoa kutsutaan encoderiksi ja decoderiksi. Encoder-neuroverkolla on tarkoitus käsitellä saadun lauseen tarkoitus, ja decoder-verkko tuottaa vastauksen tälle. Sequence to sequence-malli sopii nimensä

mukaisesti tekoälyongelmaan, jossa dataa käsitellään vaiheittain kuten esimerkiksi kielenkääntämisessä (ks. Kuvio 7).



Kuvio 7. Sequence to sequence-neuroverkko

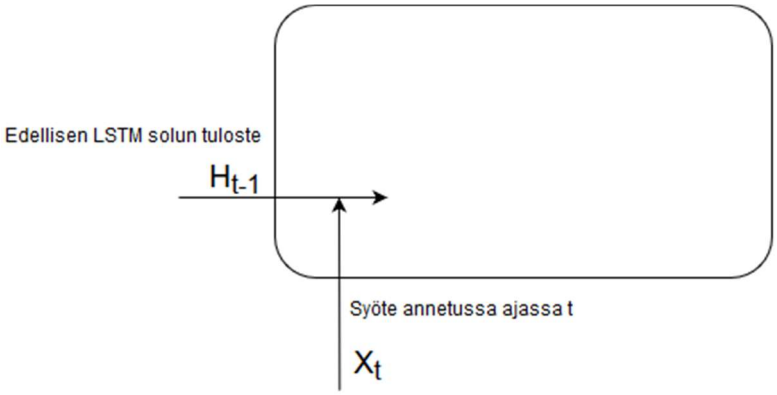
Sequence to sequence on kuitenkin vain malli, joka koostuu RNN-neuroverkoista. RNN-neuroverkkojen etuna muodostuu niiden rekursiivinen oppiminen, jossa seuraava arvioitu lopputulos riippuu edellisistä lasketuista arvoista. Näin RNN-neuroverkot tuovat suuren edun niiden muistiominaisuuden takia luonnollisen kielen ymmärtämisessä ja tuottamisessa, sillä sanojen järjestyksellä on suuri merkitys lauseessa.

3.4 Long short-term memory

Long short-term memory (LSTM) -solun ominaisuutena on sen kyky muistaa pitkiä ketjuja tietoa. LSTM-soluja käytetään neuroverkoissa, joissa aiemmalla tiedolla on merkitystä tulevaan tietoon. LSTM-solun tila koostuu useammasta vaiheesta, joista käytetään nimitystä portit. LSTM-solusta löytyy useampi variaatioita, kuinka ne toimivat sisäisesti, mutta toiminnallisuus näissä on sama. Yleisimmin käytetty LSTM-versio on se, jossa solu koostuu kolmesta portista: forget gate, input gate ja output gate. Näiden kolmen portin avulla saadaan laskettua nykyiselle solulle sisäinen tila sekä tuloste. LSTM-solu koostuu kahdesta tilasta: tuloste sekä sisäinen tila.

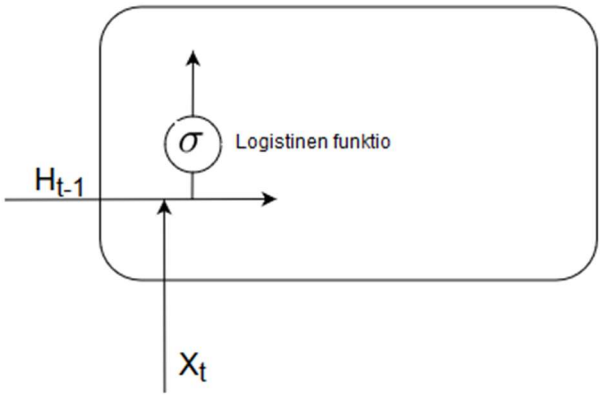
Englanninkielisissä teksteissä käytetään nimityksiä hidden state sekä internal cell memory state. (Understanding LSTM Networks 2015)

LSTM-solussa ensimmäinen vaihe on forget gate, jossa määritellään, kuinka paljon edellisen LSTM-solun tulosteesta on relevanttia tietoa nykyiselle solulle. LSTM-solu ottaa edellisen solun tulosteen sekä seuraavan syötteen ja yhdistää nämä vektoriksi (Kuvio 8). Vektori siis pitää sisällään tiedon edellisestä tulosteesta sekä nykyisestä uudesta syötteestä.



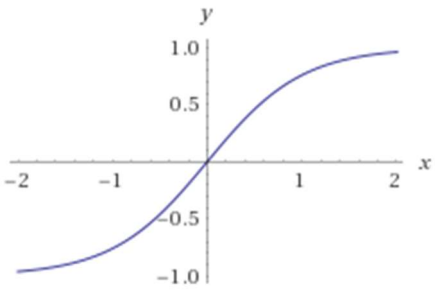
Kuvio 8. Edellisen LSTM solun tulosteen ja nykyisen syötteen yhdistäminen vektoriksi

Tämän jälkeen vektori syötetään logistiseen funktioon (ks. Kuvio 9), joka antaa tulokseksi arvon nollan ja yhden välillä. Mitä lähempänä tulos on nollaa, sitä enemmän aiemmasta tiedosta on tarkoitus unohtaa, kun taas mitä lähempänä arvo on ykköstä, sitä enemmän on tarkoitus säilyttää aiempaa tietoa.



Kuvio 9. Vektorin syöttäminen logistiseen funktioon

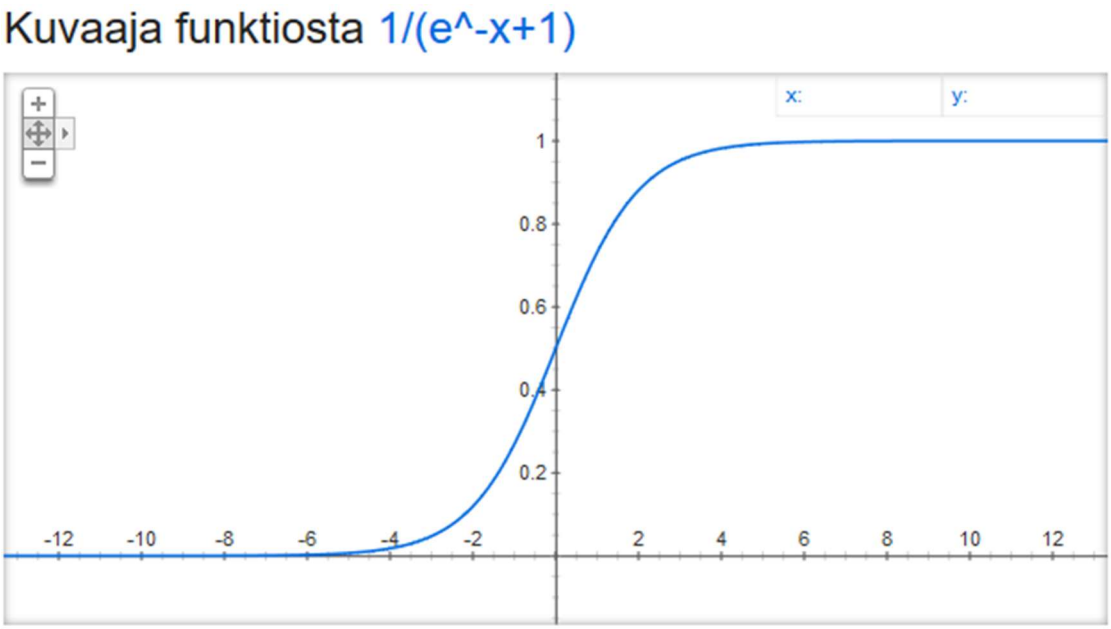
LSTM-solun sisäisen tilan päivittämiseen tarvitaan vielä input gaten antama arvo. Input gate koostuu kahdesta vaiheesta, jossa aiemman LSTM-solun tuloste ja nykyisen syötteen yhdistevektori syötetään erikseen hyperbolisen tangentti funktioon (ks. Kuvio 10) sekä logistiseen funktioon (ks. Kuvio 11).



$$f(x) = \tanh(x)$$

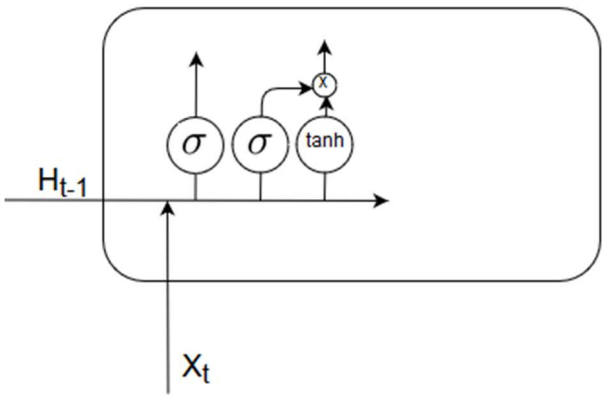
Kuvio 10. Hyperbolisen tangenttifunktion kuvaaja

Tässä logistisen funktion arvolla saadaan selville, mitä tietoa päivitetään solun sisäiseen tilaan, jossa lähempänä nollaa oleva arvo on vähemmän tärkeää tietoa ja lähempänä ykköstä olevat arvot ovat tärkeää tietoa.



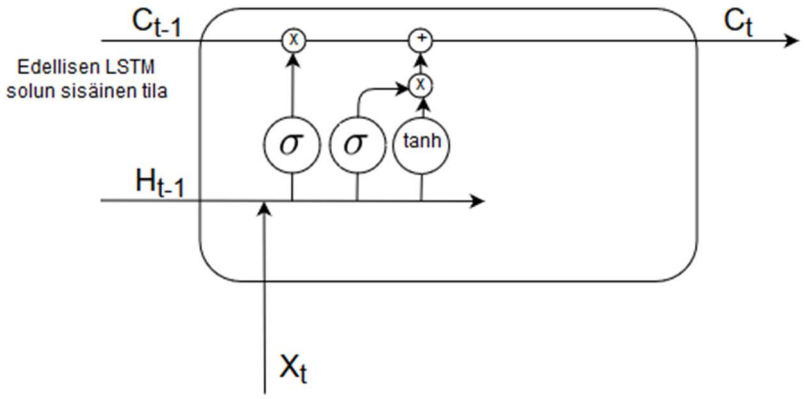
Kuvio 11. Logistisen funktion kuvaaja

Hyperbolisesta tangenttifunktiosta saadaan vektori, joka sisältää tiedon, mitä arvoja voitaisiin lisätä LSTM-solun sisäisen tilan arvoksi. Lopuksi nämä kaksi vektoria kerrotaan keskenään. Logistisen funktion vektori määrittää, mitä tietoja lisätään solun sisäiseen tilaan (Kuvio 12).



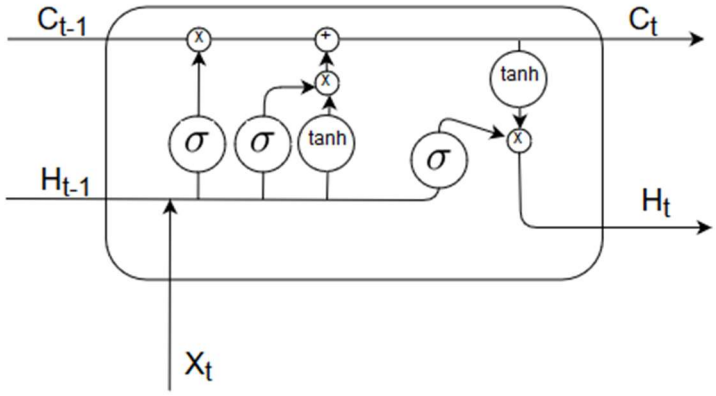
Kuvio 12. Uudesta tiedosta relevantin tiedon suodattaminen

Kun forget gate ja input gate-arvot on laskettu, voidaan LSTM-solun sisäinen tila päivittää. Ensimmäiseksi edellisen LSTM-solun sisäinen tila kerrotaan forget gatesta saadulla arvolla. Tämän jälkeen tästä saatava tulos summataan input gate-tuloksen kanssa (Kuvio 13). Tämä tulos on nykyisen LSTM-solun sisäinen tila, joka syötetään myöhemmin seuraavalle LSTM-solulle.



Kuvio 13. LSTM-solun sisäisen tilan muodostaminen

Viimeinen vaihe on output gate, josta saadaan kyseisen LSTM-solun ulos annettava tuloste. Tulosteen luomiseen tarvitaan uutta laskettua sisäistä tilaa sekä yhdistevektoria, joka koostui aiemman LSTM-solun tulosteesta sekä nykyisestä syötteestä. Yhdistevektori syötetään logistisen funktion lävitse ja nykyinen uusi LSTM solun sisäinen tila syötetään hyperbolisen tangenttifunktion lävitse. Nämä kaksi tulosta seuraavaksi kerrotaan keskenään (Kuvio 14) ja tästä saadaan lopullinen tuloste. (Nguyen 2018.)



Kuvio 14. LSTM -solun toiminto visualisoituna

4 Neuroverkot

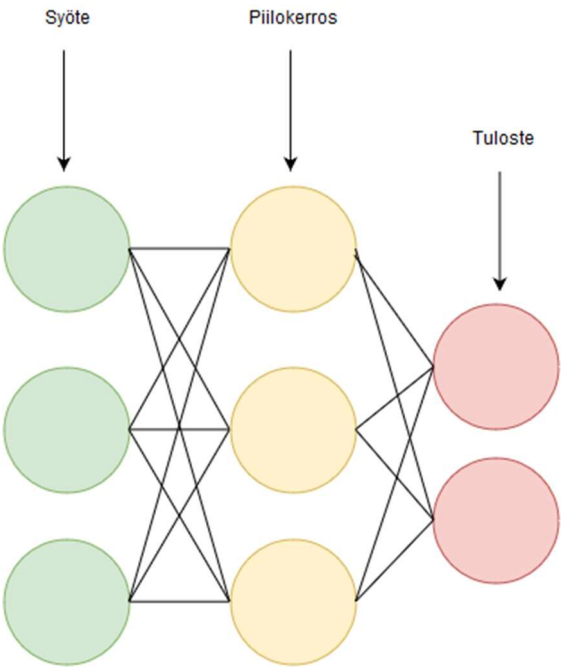
4.1 Neuroverkko

Neuroverkot ovat ketju matemaattisia algoritmeja, jotka pyrkivät luomaan tietokoneelle havaintoja datajoukoista käyttäen jotain koneoppimismenetelmää. Neuroverkot pyrkivät toimimaan samanlaisella periaatteella kuin ihmisen aivojen neuroverkkojärjestelmä. Neuroverkkoja täytyy ensin opettaa, ennen kuin niitä voidaan hyödyntää halutussa ongelmassa. Opettaminen tapahtuu käyttäen hyväksi tietokoneen laskentatehoa. Yksinkertaisuudessaan neuroverkko suorittaa matemaattisia laskuja ja pyrkii lopputuloksien avulla vertaamaan, onko saadut tulokset esimerkiksi toistuvia kuvioita tietyissä asioissa, ja mikä näiden kuvioiden suhde on toisiinsa. Näin neuroverkko oppii tarpeeksi monen koulutuskerran jälkeen tunnistamaan esimerkiksi kuvista asioita. Käytettävä neuroverkkotyyppi määräytyy ongelman mukaan. Erilaiset neuroverkot on suunniteltu ratkaisemaan tietyntyyppisiä ongelmia, ja näin ollen pelkästään yhdentyyppinen neuroverkko ei sovellu välttämättä kaikkiin ongelmiin. (Rouse 2018.)

4.2 Täysin kytketty neuroverkko

Täysin kytketty neuroverkko on neuroverkkoratkaisu, jossa jokainen neuroni kerroksien välissä on kytkettynä toisiinsa. Täysin kytkettyä neuroverkkoa käytettäessä tieto kulkee aina eteenpäin eli kerroksesta seuraavaan. Tieto siirtyy kerroksien välillä, kun

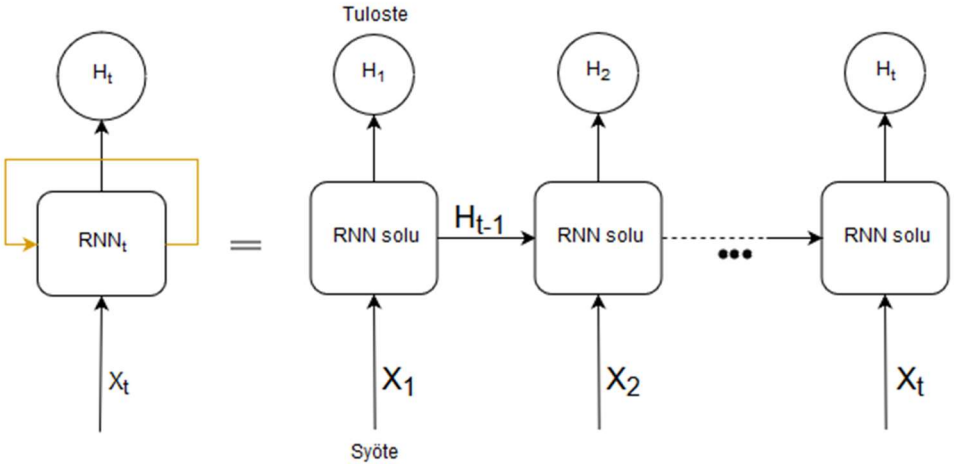
aktivointifunktio aktivoi joitain neuroneita seuraavasta kerroksesta (ks. Kuvio 15). Kaikki neuronit eivät välttämättä aktivoidu aina seuraavasta kerroksesta. Täysin kytketty neuroverkko on toiminnaltaan hyvin yksinkertainen verrattuna muihin neuroverkko malleihin. (Gupta 2017.)



Kuvio 15. Täysin kytketty neuroverkon visualisointi

4.3 Recurrent neural network

RNN on neuroverkko, joka käsittelee datan vaiheittain. RNN-neuroverkkoja käytetään tapauksissa, joissa halutaan ennustaa jaksomaista kaavaa tai tapahtumaa. Jaksomaisuudella tarkoitetaan tapahtumaa, jossa jonkin asian aloitus johtaa jonkin ajan päästä tiettyyn lopputulokseen. RNN-neuroverkon data, jota halutaan käsitellä yleisimmin sisältää tapahtumia, jotka ovat toistuvia tai riippuvaisia edellisestä asiasta. Yleisimpiä käyttösovelluksia ovat esimerkiksi puheentunnistus, chattibotit tai osakemarkkinoiden ennustaminen. RNN-neuroverkko koostuu RNN-soluista, jotka ovat kopioita edellisestä solusta (Kuvio 16).

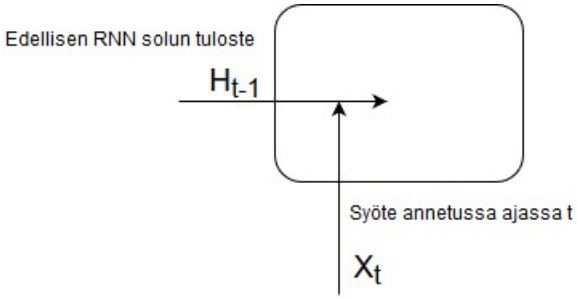


Kuvio 16. RNN-neuroverkko

RNN-neuroverkoissa tulosteen luomisessa otetaan aina huomioon aiemmin huomioidut asiat. Eli toisin sanoen seuraava solu tietää edellisen solun tulosteen, jota se käyttää hyödyksi luodessaan sen hetkistä tulostetta (Banerjee 2018).

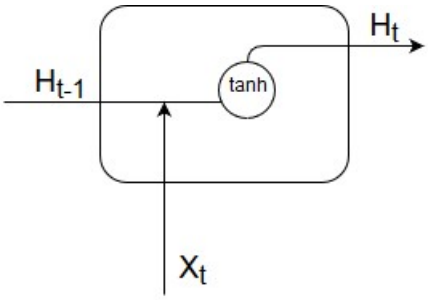
RNN-neuroverkon soluissa on kuitenkin ongelma, mikäli käytetään gradientti tyyppisiä optimointialgoritmeja. RNN-neuroverkkoa koulutettaessa neuroverkko laskee koulutusjakson lopuksi loss-funktiolla arvon, joka määrittelee, kuinka huonosti neuroverkko suoriutui. Tätä arvoa sittemmin neuroverkko käyttää säätäessään sisäisesti arvoja, jotta neuroverkko pääsisi haluttuun lopputulokseen. Tämä arvo kuitenkin pienenee eksponentiaalisesti, mitä kauemmas neuroverkon täytyy solujen tiloja arvioida. Näin ollen RNN-neuroverkon oppimiskyky heikkenee vaihe vaiheelta, jossa ensimmäinen solu oppii aina hitaimmin tunnistamaan datasta kuvioita. Tämä ongelma johtuu RNN-solun yksinkertaisuudesta. Tähän ongelmaan on kehitetty ratkaisu, jossa RNN-solu korvataan LSTM-solulla. LSTM-solun toiminta on hieman hienostuneempi ja monimutkaisempi kuin perinteisellä RNN-solulla.

RNN-solu ottaa edellisen solun tulosteen sekä seuraavan syötteen ja yhdistää nämä vektoriksi (Kuvio 17). Vektori siis pitää sisällään tiedon edellisestä tulosteesta sekä nykyisestä uudesta syötteestä.



Kuvio 17. Edellisen RNN-solun tulosteen ja nykyisen syötteen yhdistäminen

Tämän jälkeen vektori syötetään hyperboliseen tangenttifunktioon, josta saatava arvo on solun tuloste (ks. Kuvio 18). Hyperbolinen tangenttifunktio antaa arvoiksi yhden ja miinus yhden väliltä, ja näin ollen se auttaa säätelemään vektorin arvoja niin, että ne eivät kasvaisi liian suuriksi. (Tran 2016.)

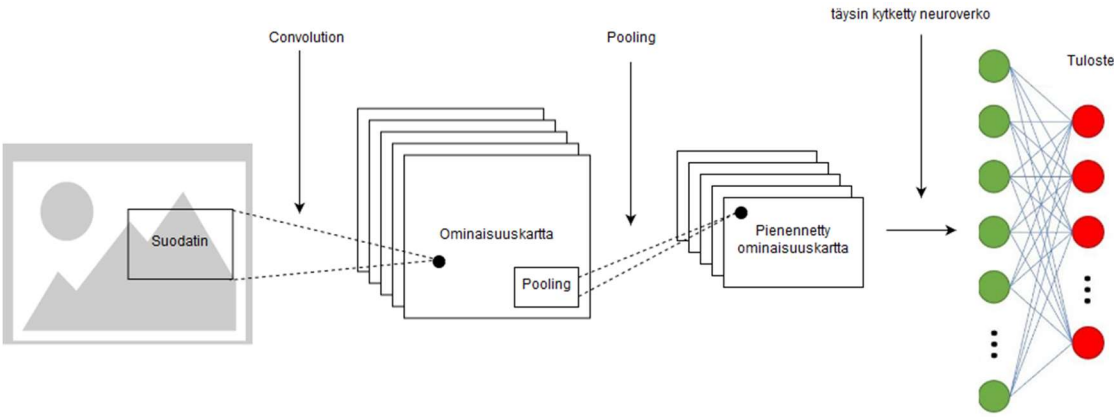


Kuvio 18. RNN-solun toiminto visualisoituna

4.4 Convolutional neural network

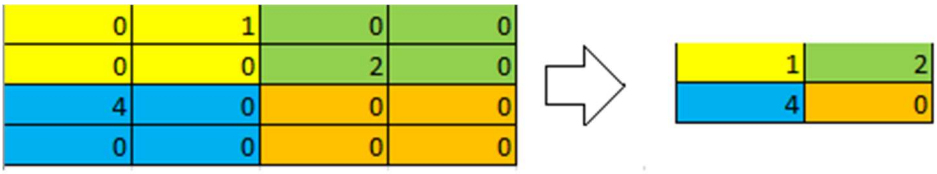
Convolutional neural network (CNN) on neuroverkko, joka käyttää suodattimia asioiden tunnistamiseen. Jokaisella suodattimella on CNN-verkossa oma tarkoituksensa, kuten esimerkiksi toimia kuvantunnistuksessa reunojen tunnistimena tai kuvan terävyyden tunnistimena. CNN-neuroverkkoa käytetään ongelmissa, jossa syötettävän datan muoto olisi tärkeä analysoida sellaisenaan. Jokaisessa convolutional-kerroksen vaiheessa pyritään aina tunnistamaan hienostuneempia yksityiskohtia edelliseen kerrokseen verrattuna. CNN-neuroverkkoa on käytetty yleisimmin objektin- tai tekstin tunnistamiseen kuvasta, mutta sitä on myös mahdollista käyttää esimerkiksi tekstin analysoinnissa tai puheentunnistuksessa. Kuvienkäsittelyssä CNN-neuroverkko

analysoi kuvan osissa, jossa kuvasta otetaan ennalta määritellyn kokoinen palanen. Tämä palanen on CNN-neuroverkoissa suodatin, jolla luodaan ominaisuuskarttoja. Kuva käydään läpi siirtäen suodatinta ennalta asetetun arvon verran sivummalle, kunnes koko kuva on menty lävitse. Samalla suodattimen matriisi sekä kuvasta otettava matriisi kerrotaan yhteen, ja näistä tuloksista saadaan loppujen lopuksi luotua kuvalle ominaisuuskartta. Saadut ominaisuuskartat syötetään täysin kytketty neuroverkon lävitse, josta saadaan lopullinen vastaus (ks. Kuvio 19).



Kuvio 19. CNN-neuroverkon toiminta

Mikäli käsiteltävän datan kokoa halutaan pienentää alkuperäiseen nähden tätä toimintoa, kutsutaan CNN-neuroverkoissa pooling kerrokseksi. Poolingissä voidaan esimerkiksi käyttää maksimimenetelmää, jossa ominaisuuskartoista otetaan määritellyn ikkunan sisällä olevista arvoista suurin arvo (ks. Kuvio 20). Näin alkuperäinen data pienenee määritetyn pooling ikkunan verran pienemmäksi, mutta samalla saatetaan menettää tärkeitä tietoja datasta (Saha 2018.)



Kuvio 20. Maksimi pooling, jossa ikkunan koko on kaksi

5 Informaatiovaikuttaminen

5.1 Propaganda

Propaganda on voimakas sekä tehokas informaatiovaikuttamisen tapa levittää haluttua aatetta. Internetin myötä kuka tahansa voi jakaa tai tuottaa informaatiota maailmanlaajuisesti riippumatta heidän geolokaatiostaan. Näin ollen internetissä muodostuukin ongelmaksi informaatiota luettaessa tai sitä jakaessa sen oikeellisuus sekä paikkansapitävyys. Mikäli ei ole tiedossa kyseisen tiedon alkuperää, on tämän paikkansapitävyys mahdoton arvioida. Toisin sanoen internetti myös mahdollistaa propagandan salamannopean levittämisen laajoille ihmismassoille, josta hyvänä esimerkkinä voidaan käyttää sosiaalisen media palveluita (ks. luku 5.2). Sosiaalisessa mediassa propagandaa on mahdollista levittä pienessä ajassa tuhansien ihmisien luettavaksi tai katseltavaksi, sillä sosiaalisen median palvelusta riippuen saattaa kyseisellä palvelulla olla jopa miljoonia eri käyttäjiä. Sosiaalisesta mediasta on tullut nykypäivänä se paikka, jossa ihmiset useimmin jakavat omia ajatuksiaan ja tietoa muille. Näin ollen, kun otetaan huomioon mahdollinen informaation leviämisnopeus sekä julkaistun propagandan tietolähteen mahdollinen vaikea selvittäminen, saattaa tällöin sosiaalinen media jopa tarjota ihanteellisen ympäristön propagandan levittämiseen.

Tekoälyteknologioiden ja bottien yleistyessä myös näiden tuottama informaatio eri kanaviin kasvaa, ja mikäli esimerkiksi chattibotti on koulutettu käyttäen laajaa ja monipuolista datajoukkoa saattaa tällöin botin tuottama teksti olla kieliopillisesti virheetöntä. Tällöin chattibotin tuottama data saattaa olla melkein mahdoton erottaa ihmisen tuottamasta datasta. Tekoälyn kehitys viime vuosina on kehittynyt niin paljon, että jopa neuroverkoilla on mahdollista tuottaa aidon näköistä sekä kuuloista videomateriaalia jostain henkilöstä käyttäen pelkästään neuroverkkoja sekä muita tekoälyteknologioita. Näin ollen videomateriaaleistakin tulee kyseenalaistettavaa informaatiota, jonka alkuperä tulisi tunnistaa sen oikeellisuuden takia, mikä ennen on saatettu pitää itsestään selvyytenä. Nykyinen tekoälyteknologia siis mahdollistaa voimakkaan tavan jakaa propagandaa näin halutessaan tekstin tai jopa videon muodossa. Tämä

kaikki onnistuu ilman, että ihmisen tarvitsee välttämättä hallinnoida bottia, muutoin kuin botin opetusvaiheessa. Tämän jälkeen botti voidaan määritellä käyttämään palveluiden ohjelmistorajapintoja toimiakseen itsenäisesti.

5.2 Sosiaalinen media

Nykyään voidaan sanoa, että kaikki käyttävät jollaintavoin jonkinlaista sosiaalisen median palvelua. Kolme yleisintä sosiaalisen median palvelua vuonna 2018 olivat Facebook, YouTube ja Instagram (top 15 most popular social networking sites 2018). Näin ollen tekoäly näkyy myös suurien sosiaalisen median palveluiden alustoissa, joihin on mahdollista luoda omia botteja. Facebook mahdollistaa yrityskäyttäjille chattibotti alustan, jota voidaan käyttää asiakaskommunikaatioon Facebookin kautta. Twitteri myös mahdollista omien bottien luomisen palveluun. Kyseisissä alustoissa botti käyttää kommunikoinnissa kyseisen alustan ohjelmistorajapintoja.

Suuret sosiaalisen median alustat myös hyödyntävät tekoälyä datan keruussa ja sen analysoinnissa. Toisin sanoen sosiaalisen median palvelut, jotka keräävät käyttäjistään dataa voivat luoda asiakasprofiileja käyttäjistä. Näitä profiileja voidaan hyödyntää yrityksen omiin palveluihin. Esimerkiksi Facebook kerää käyttäjistään erinäköistä dataa, jotta heidän oma tekoälynsä osaa kohdistaa mahdollisesti kiinnostavia mainoksia käyttäjille. YouTube pyrkii profiilin avulla ehdottamaan videoita, jotka saattavat kiinnostaa käyttäjää katsoa seuraavaksi. Datan keruusta saattaa kuitenkin tulla ongelma, jos palvelu ei hallinnoi sitä oikein.

5.3 Tekoäly informaatiovaikuttamisessa

Tekoälyn käyttäminen informaatiovaikuttamisessa on lisääntynyt vahvasti osittain koneiden tehonkasvamisen takia. Jopa kotikäyttöisillä tietokoneilla, joista löytyy erillisiä näytönohjaimia, on nykypäivänä mahdollista opettaa botteja tuottamaan kielellisesti selkeätä ja virheetöntä tekstiä. Tekoälyn avulla voidaan myös ohjata ihmisiä tietyntyyppisiin kanaviin, kuten esimerkiksi mainoksien avulla. Tekoäly on myös mahdollista opettaa tunnistamaan propagandan tapaisia uutisia tai mainoksia, jossa

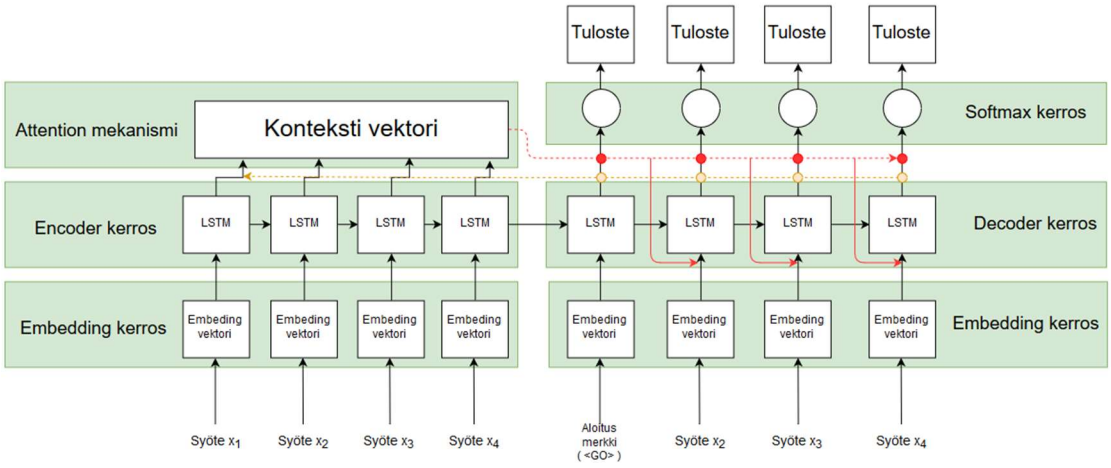
tekoäly pyrkii tekstin perusteella analysoimaan, onko uutinen tai viesti propagandaa vai ei.

Etelä-Kalifornian yliopiston vuonna 2017 tekemässä tutkimuksessa pelkästään Twitterin käyttäjistä arvioitiin suurimmillaan olevan jopa 15 prosenttia botteja. Tuolloin Twitterissä olevista käyttäjistä 15 prosenttia oli arviolta noin 48 miljoonaa botille tarkoitettua tiliä. Yksi suurimmista ongelmista on poliittisten aatteiden levittäminen bottien avulla eri sosiaalisissa medioissa. Yhdysvaltain presidentin vaaleissa vuonna 2016 nähtiin poliittisten aatteiden levittämistä sosiaalisissa medioissa käyttäen hyväksi botteja. Twitterissä nimellä Jenna Abrams esiintynyt Venäläiseen Internet Research Agency nimiseen yritykseen yhdistetty tekoäly botti toimi Donald Trumpin kannattajana, ja kyseisellä tilillä oli jopa 70 tuhatta seuraajaa. Useat isot mediat jopa viittasivat kyseisen Twitter tilin viesteihin uutisissaan tietämättä, että kyseessä oli botti eikä oikea ihminen (Jenna Abrams: the Trump-loving Twitter star who never really existed 2017.).

6 Toteutus

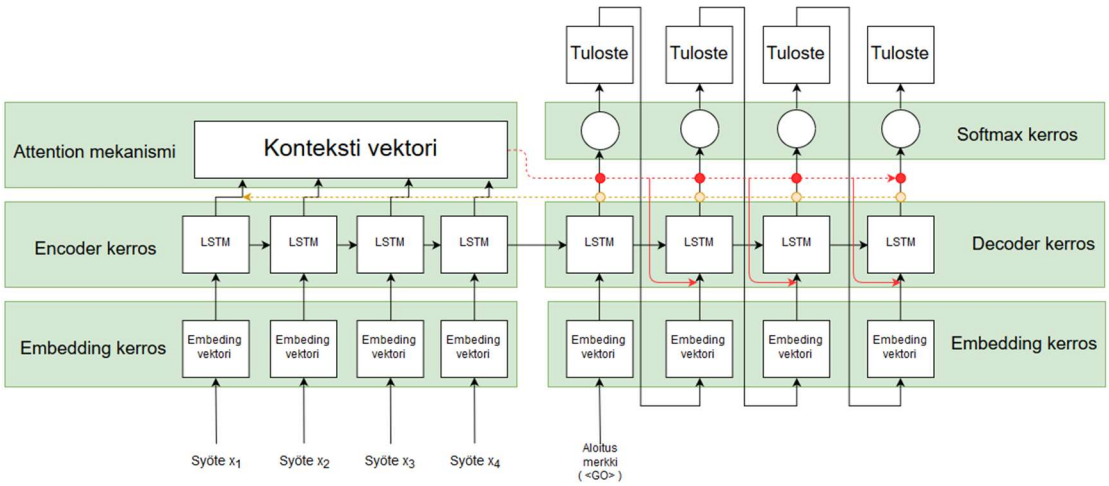
6.1 Toiminnallisuus

Tehtävänannon mukaisesti botille haluttiin pääominaisuutena olevan Gnu Social-mikroblogi alustalle tulevien viestien tulkitseminen ja näihin vastaaminen. Botin täytyi myös pystyä luomaan propagandamaista informaatiovaikuttamista näin haluttaessa. Botille siis haluttiin luoda niin sanotusti useampia persoonallisuuksia, joiden avulla botti vastaa annettuun viestiin tietyn tavan mukaisesti. Itse botti päädyttiin toteuttamaan käyttäen LSTM RNN-neuroverkkoa, joka opetettiin käyttäen ohjattua oppimisen menetelmää. Neuroverkko toimii koulutettaessa ja käytettäessä samalla tavalla, mutta kun neuroverkkoa koulutettiin tällöin decoderille syötettiin haluttu vastaus oman edellisen LSTM-solun tulosteen sijasta (Kuvio 21).



Kuvio 21. Neuroverkon käyttäytyminen koulutettaessa

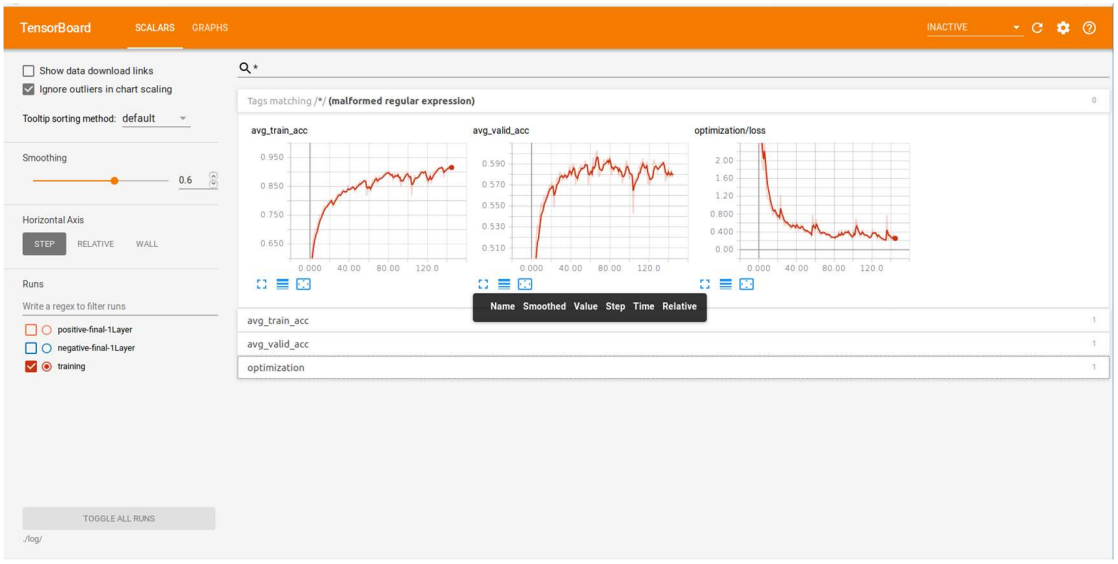
Neuroverkkoa käytettäessä decoderin LSTM-solun tuloste syötetään seuraavan LSTM-solun syötteeksi (Kuvio 22). Encoderi ja decoderi eivät jaa embedding-kerrosta vaan kummallakin RNN-verkolla on omat embedding-kerrokset. Tästä on hyötyä kielen kääntämisessä, jolloin encoderin embedding-kerros sisältää käännettävän kielen sanoja ja decoderin embedding-kerros sisältää käännetyn kielen sanoja.



Kuvio 22. Neuroverkon käyttäytyminen käytettäessä

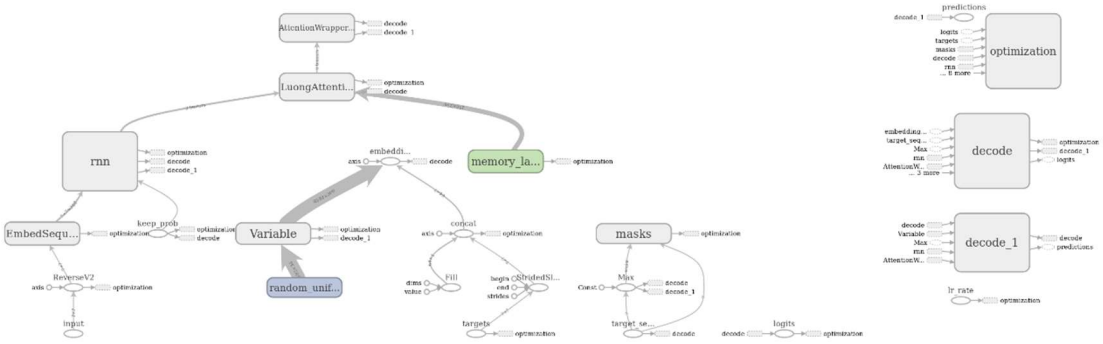
Neuroverkko myös hyödynsi attention mekanismia, jolloin decoderi ottaa paremmin huomioon encoderin tulosteet (ks. Luku 6.4). Kielenkääntäminen käyttää tismalleen samanlaista verkkoratkaisua. Ainoana erona toki oli vain datajoukko, joka koostui englannin- ja suomenkielisistä lauseista. Kielenkääntämistä opetettaessa encoder-verkon syötteet olivat käännettävän kielen lauseita ja decoderin-verkon syötteet taas halutun kielen lauseita.

Koulutuksen edistymistä voitiin seurata käyttäen Tensorflowista löytyvää tensorboardia, josta nähtiin vaihevaiheelta botin oppimiskyky sekä paljonko boti voi vielä oppia koulutusdatajoukosta. Botin oppimiskyky määräytyi loss-arvolla. Mitä lähempänä loss-arvo oli nolla, sitä vähemmän botti enää pystyi kouluttamaan itseään. Loss-arvoa seurattaessa myös voitiin havaita milloin botti lopettaa opettelemisen ja alkaa vain tallentamaan muistiin mitä syötettyyn lauseeseen pitää vastata. Tämänkaltainen tapahtuma voitiin todentaa sillä, mikäli loss-arvo alkaisi kasvamaan laskemisen sijasta. Tätä kutsutaan englanninkielisissä teksteissä nimellä overfitting eli ylioppiminen. Tensorboardista samalla myös voitiin havaita millä prosentilla botti osasi tuottaa vastauksia nähtyihin lauseisiin, kuin myös tuntemattomiin lauseisiin. Tensorboardissa löytyvästä accurancy-arvosta nähtiin botin ennalta nähtyjen lauseiden oikeanlainen vastaaminen, ja validation accurancy kertoi ennalta tuntemattomiin lauseisiin oikeanlainen vastaaminen (Kuvio 23).



Kuvio 23. Tensorboard

Tensorboardista myös pystyttiin näkemään verkon flow-malli, joka esitti kuinka tieto kulkee neuroverkossa (Kuvio 24).



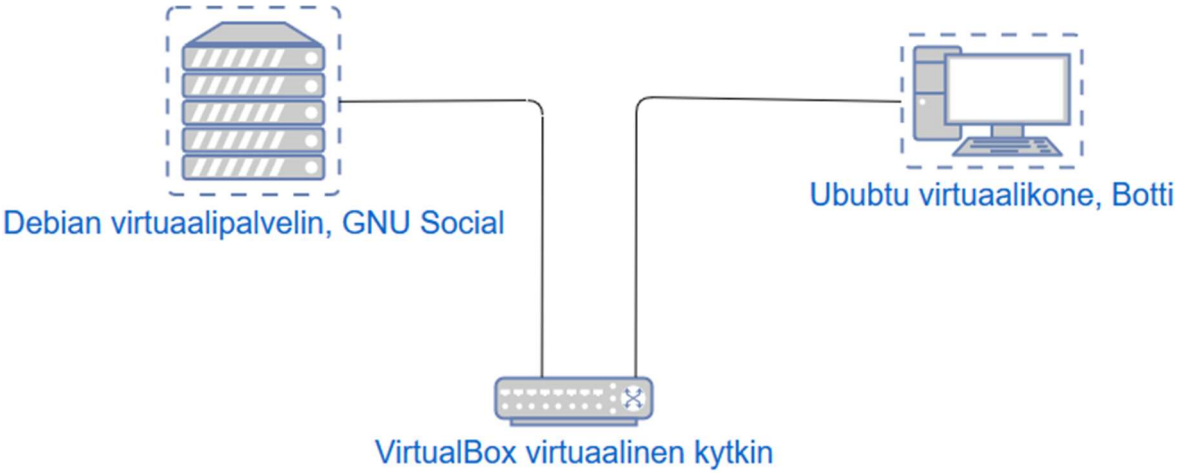
Kuvio 24. Tensorboard flow-kaavio

6.2 Ympäristö

Chattibotin toimintaympäristö koostui erillisistä palvelin implementaatioista. Itse botti pystyi toimimaan alustariippumattomasti millä vain laitteella, johon on asennettu python ohjelmointikieli sekä Tensorflow koneoppimiskirjasto. Botin sovellus on suunniteltu niin, että se käyttää kommunikoimiseen eri alustojen ohjelmointirajapintoja. Näin ollen botti voitaisiin liittää mihin tahansa sosiaalisen median palveluun, mikäli kyseisestä alustasta vain löytyy ohjelmointirajapinta tuki. Tutkimuksessa käytettiin viestien ja keskustelujen luomiseen avoimen lähdekoodin GNU social-mikrobloggaus alustaa, joka on ominaisuuksilta Twitterin tapainen. GNU socialista löytyy ohjelmointirajapinta, joka on hyvin samankaltainen kuin Twitterillä. Virtuaaliympäristönä käytettiin Oracle VM VirtualBox virtualisointi alustaa, jota ajettiin normaalilla pöytäkoneella.

Pelkästään botin oma ympäristö koostui linux virtuaalikoneesta, johon oli käyttöjärjestelmäksi asennettu ubuntu. Botin neuroverkko oli siirretty verkko ja näin ollen valmiiksi koulutettu erillisellä tehokkaammalla näytönohjaimisella työasemalla. Virtuaalikoneella toimiakseen botti ei vaatinut suuria laskentatehoja vaan pystyi pyörimään vain pelkän prosessorin voimin. GNU socialin ympäristö koostui linux palvelimesta, johon oli käyttöjärjestelmäksi asennettu debian (Kuvio 25). Debian palvelinkoneelle

asennettiin MariaDB-tietokanta, jota GNU social käytti viestien ja käyttäjien tallentamiseen.



Kuvio 25. Virtuaaliympäristö

6.3 Datan prosessointi

6.3.1 Keskusteluominaisuuden datan prosessointi

Botin koulutusdata koostui eri hotellien arvosteluista, jotka saatiin hankittua Kaggle-datapankki palvelusta. Datajoukosta suodatettiin vain arvostelut, jotka olivat positiivisia tai negatiivisia käyttäen pythonin pandas-kirjastoa. Positiiviset sekä negatiiviset viestit eriteltiin omiksi koulutusdatajoukoiksi. Kumpaankin koulutusdatajoukkoon loppujen lopuksi muodostui 8000 eri arvostelua, josta 1600 viestiä käytettiin neuroverkon koulutuksen testaamiseen ja arvioimiseen.

Neuroverkkoa koulutettaessa sen koulutusdata ladataan tekstitiedostosta muistiin, josta jokaiselle erilaiselle sanalle annetaan uniikki kokonaisluku arvo. Nämä uniikit numeroarvot tallennetaan sittemmin listaan, josta niitä voidaan hakea tarvittaessa. Näin teksti saadaan muunnettua numeraaliseen muotoon, jotta neuroverkko ymmärtäisi syötteen sekä myöhemmässä vaiheessa neuroverkon tuloste saadaan muunnettua takaisin tekstimuotoon. Kun jokaiselle sanalle on saatu annettua numeraalinen arvo, voidaan lauseet muuntaa numerolistaksi, jossa sana korvataan sille määritellyllä numerolla. Tämän jälkeen kaikki numerolistat tasataan samanpituisiksi selvittämällä pisin lause datasta. Tasaus tapahtuu lisäämällä täytemerkin numeraalinen arvo

muunnettujen numerolistojen alkuun tai loppuun. Täytemerkki on varattu numerolla nolla. Tämän lisäksi muita erikoismerkkejä on koulutusdatan aloitus- ja lopetusmerkki, jotka lisätään decoderin koulutusdatan lauseiden alkuun ja loppuun. Nämä merkit voivat olla mitä vain ja niitä käytetään määrittämään, milloin neuroverkko lopettaa tai aloittaa lauseen.

6.3.2 Kielenkääntämisen datan prosessointi

Botin kielenkääntämisen koulutusdata kerättiin monista eri lähteistä. Osa keskusteluominaisuuden koulutusdatasta käännettiin suomeksi käyttäen Google kääntäjää. Tämä tehtiin siitä syystä, että kääntäjän sanavarastoon saatiin kerättyä sanoja, jotka esiintyisivät keskusteluissa. Näin varmistuttiin siitä, että botin tuottamista sanoista suurin osa pystyttäisiin tunnistamaan ja kyseinen lause näin ollen kääntämään järkevästi.

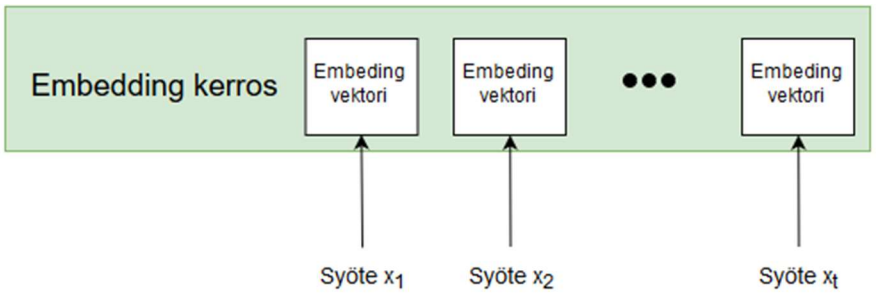
Kielenkääntämisen koulutusdata käsiteltiin tismalleen samalla tavalla kuin keskusteluominaisuuden koulutusdata. Ensimmäiseksi data siis ladattiin tekstitiedostosta muistiin. Tämän jälkeen kummastakin kielestä poimittiin uniikit sanat, jotka tallennettiin sanalistaan ja samalla sanoille annettiin oma uniikki kokonaisluku arvo. Seuraavaksi tekstimuotoiset lauseet muunnettiin numeraaliseen muotoon, jonka jälkeen lauseet tasattiin samanpituisiksi. Tässä vaiheessa siis data oli siinä muodossa, että sitä voitaisiin käyttää koulutuksessa. Encoderin data koostui käännettävän kielen sanoista ja decoderin data koostui käännetyn kielen sanoista.

6.4 LSTM RNN-neuroverkko

Botin neuroverkko suunniteltiin käyttämään sequence to sequence-mallin mukaista RNN-neuroverkkoa, jolloin lopullinen neuroverkko koostui encoder ja decoder-verkoista. Kummatkin neuroverkot koostuivat omista LSTM-soluista, joita oli mahdollista myös pinota päällekkäin. Pinoamisella saavutettiin syvempi datan analysointi. Sequence to sequence-mallin mukainen RNN-verkko valittiin toteutustavaksi, koska sillä oli hyvä toteuttaa sekä keskusteluominaisuus kuin myös kielenkääntäminen.

Sequence to sequence-malli onkin yksi käytetyimmistä tavoista luoda chattibotteja tai tehdä omia kielikääntäjiä.

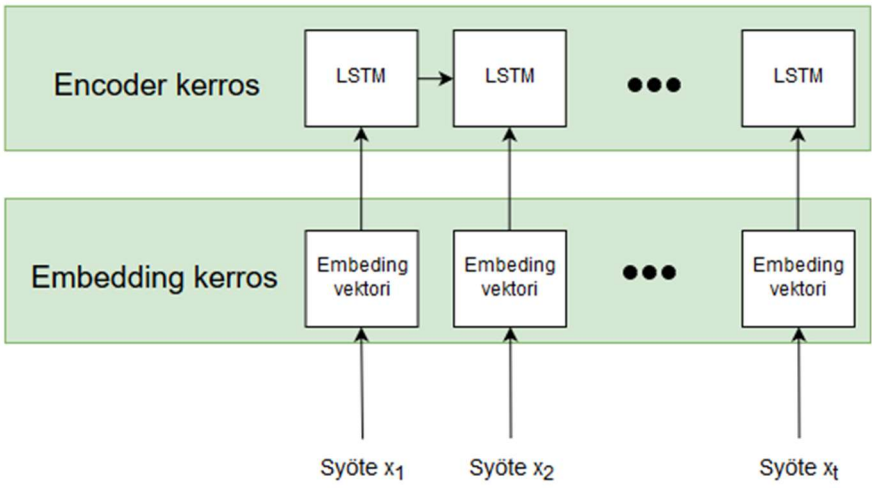
Ennen kuin RNN-neuroverkolle annettiin syötteitä, ne syötettiin ensin embedding-kerrokselle, jonka toiminnallisuutena on muodostaa sanoista sanavektoreja (Kuvio 26). Näin neuroverkko pystyisi helpommin hahmottamaan lauseiden rakennetta sekä lauseen kontekstia. Samalla embedding-kerroksella saavutettiin syötteen saaminen pienempään muotoon. Mikäli embedding-kerrosta ei olisi käytetty olisi tällöin annettava syöte jouduttu muotoilemaan neuroverkolle one-hot-enkoodattuna vektorina. One-hot-enkoodaus menetelmää emme halunneet käyttää sillä vektorit voivat näissä kasvaa suuriksi, mikäli sanavarasto on suuri. One-hot-enkoodattu vektori myöskään ei ole laskennallisesti järkevää käyttää verrattuna embedding matriisiin. Embedding-kerroksen kouluttaminen aloitettiin tyhjästä, jolloin kerros oppi sanoja ja näiden merkityksiä toisistaan koulutuksen yhteydessä. Tässä olisi myös mahdollisuutena ollut käyttää valmiiksi koulutettua embedding-kerrosta, kuten käyttäen esimerkiksi global vectors for word representationia (GloVe), mutta tutkimuksessa päädyttiin loppujen lopuksi tutkimaan neuroverkon omatoimista oppimista.



Kuvio 26. Syötteen antaminen embedding-kerrokselle

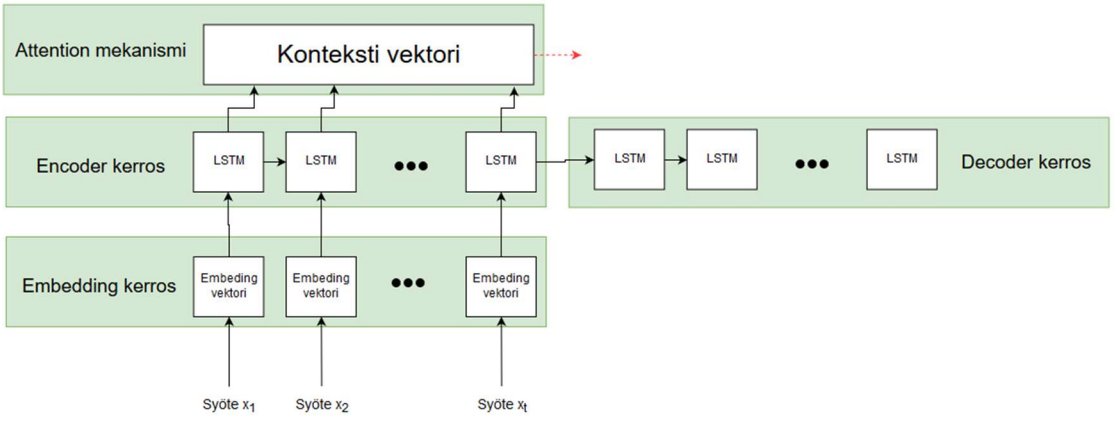
Kun lause on embedding-kerroksessa saatu muunnettua sanavektoreiksi, annetaan kyseinen syöte RNN-neuroverkon LSTM-soluille sanavektori kerrallaan (Kuvio 27). Kun LSTM-solu on saanut suoritettua datan analysoinnin se siirtää aina tilansa seuraavalle LSTM-solulle. Tätä tilaa sittemmin seuraava LSTM-solu käyttää hyväksi muodostaessaan tulostetta. Näin LSTM-solut saavat hyödynnettyä edellisien sanojen

järjestystä sekä merkitystä. Chattibotissa emme kuitenkaan olleet kiinnostuneita encoderin tulosteista vaan halusimme pelkästään siirtää encoderin LSTM-solujen keräämän tiedon decoderi-neuroverkolle.



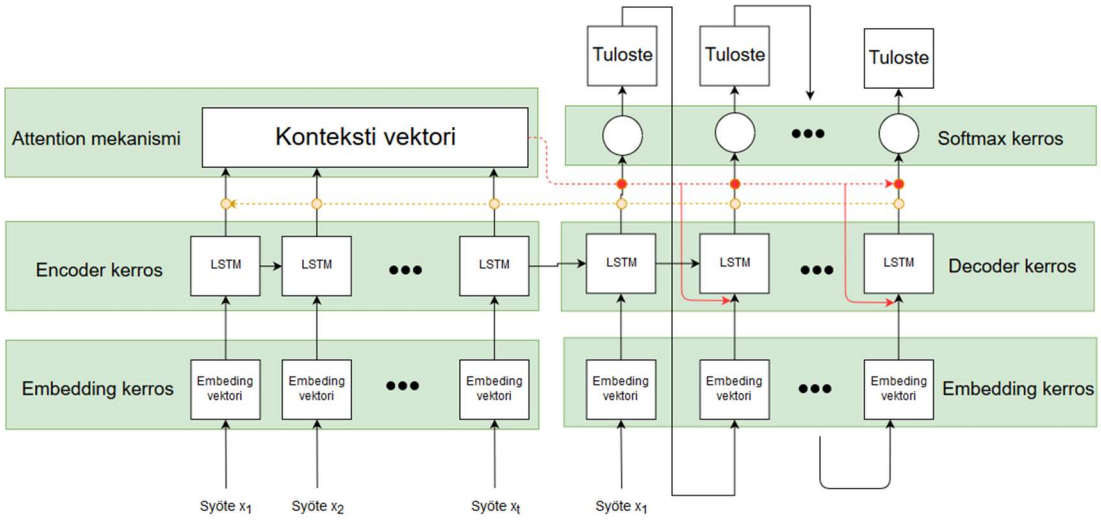
Kuvio 27. Sanavektorien syöttäminen encoder-kerrokselle

Kun Encoder kerros on käynyt lauseen sanat lävitse ja antanut aina LSTM-solun tilan seuraavalle solulle, tämän jälkeen encoder-kerros syöttää LSTM-solujen tilan decoder-kerrokselle (Kuvio 28). Ennen kuin decoder tuotti tulosteita lisäsimme myös luong attention mekanismin neuroverkkojen väliin, joka auttaa tulosteen luomisessa.



Kuvio 28. Encoderin LSTM-solujen tilan kopioiminen decoderin LSTM-soluille

Attention mekanismia hyödynnettäessä encoderin LSTM-solujen tulosteisiin yhdistetään decoderin sen hetkisen LSTM-solun tuloste. Tämän jälkeen nämä yhdistevektorit pisteytetään ja luodaan niin sanottu kontekstivektori, josta saadaan selville mihin encoderin LSTM-solun tulosteen sanaan keskitetään eniten huomiota. Tätä kontekstivektoria decoderi käyttää hyväksi, kun se syöttää lopullisen tulosteen softmax-kerrokselle. Tämän lisäksi kontekstivektori yhdistetään seuraavan decoderin LSTM-solun syötteen kanssa, jolloin seuraava decoderin LSTM-solu tietää mihin encoderin sanaan viimeksi keskitettiin eniten huomiota. Tämä tapahtuu jokaisessa decoderin LSTM-solun vaiheessa, kunnes decoderi on saanut tulostettua koko lauseen (Kuvio 29).



Kuvio 29. Neuroverkon toiminta kokonaisuudessaan havainnoituna

Decoderin softmax-kerroksesta otetaan aina todennäköisin sana, joka on lopullinen lauseen sana. Suurimman arvon omaava sana ei välttämättä aina ole paras mahdollinen sana lauseen kannalta, ja näin ollen tässä olisi voitu käyttää beam search (ks. 8.4) menetelmää, jolloin sanoja olisi voitu poimia lauseen kannalta järkevimmin.

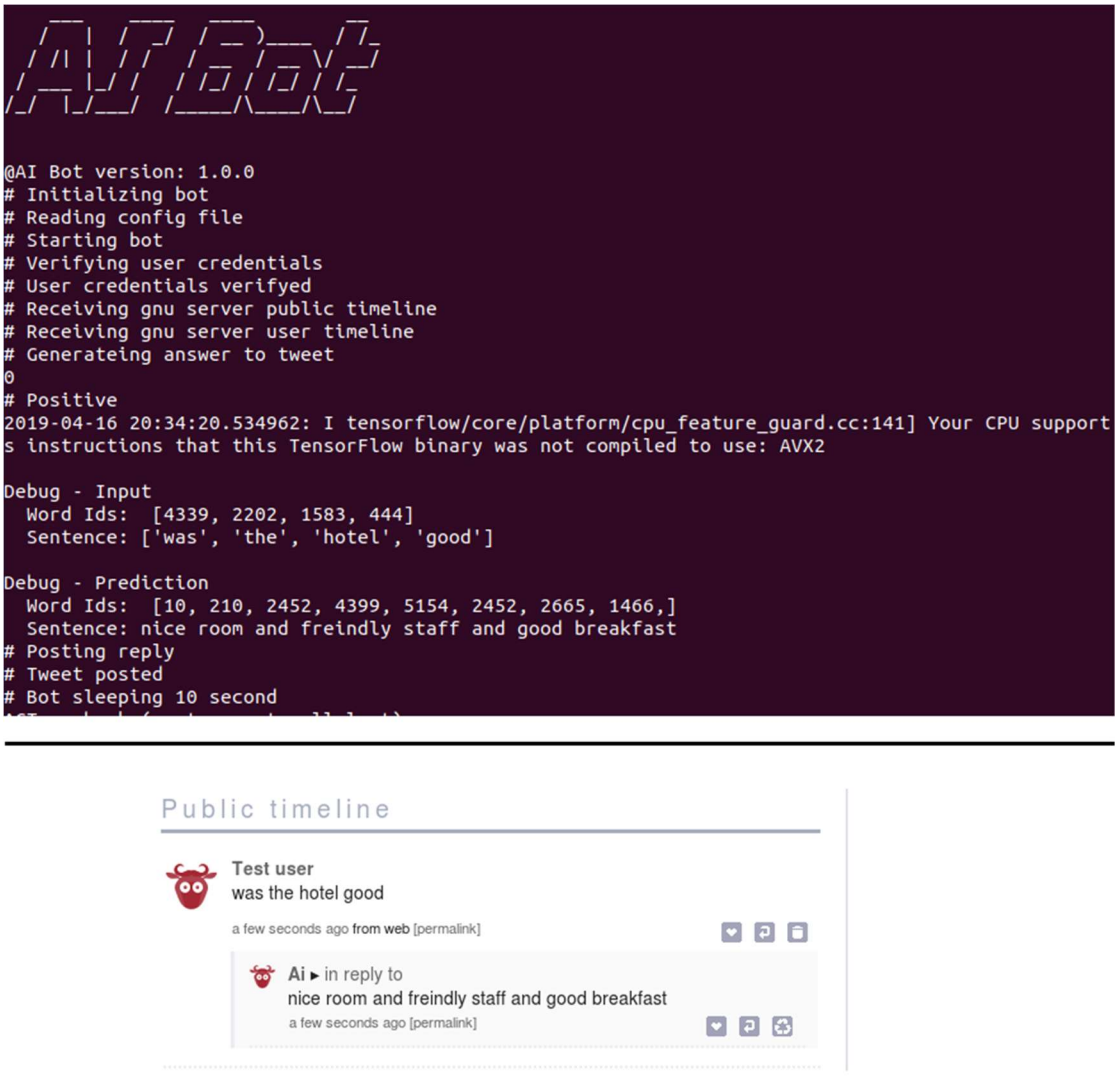
6.5 Botin käyttäminen

Koulutuksen jälkeen neuroverkko voitiin siirtää virtuaalikoneelle, jossa bottia ajettiin. Botin käynnistämiseen tarvittiin komentoriviltä ajaa käynnistystiedosto. Botille voitiin myös asetustiedostoon määritellä erilaisia asetuksia, kuten haluttu GNU social-palvelimen osoite sekä käyttäjä kyseiseen palveluun. Samalla voitiin myös määritellä mahdollinen haluttu propaganda sanalista, jolloin mikäli botti tunnistaa keskusteluista kyseisiä sanoja käyttäisi botti tällöin propagandalla koulutettua neuroverkkoa kyseiseen viestiin. Botti nouti määritellyn väliajoin viestejä GNU social-palvelimelta käyttäen omaa API-kirjastoaan. Botti myös samalla piti kirjaa mitä viestejä oli haettu, jolloin sen ei tarvinnut aina hakea kaikkia viestejä palvelimelta. Viestin kielentunnistamisessa käytettiin pythonista löytyvää kielentunnistus kirjastoa ja mikäli kieli tunnistettiin suomeksi, botti käytti kielenkääntämisen neuroverkkoa tällöin omassa vastauksessa. Sillä vastaus ominaisuus oli koulutettu pelkästään englanninkielisillä lauseilla, täytyi GNU social-palvelimelta haetut suomenkieliset viesti kääntää englanniksi, jonka jälkeen englanninkielinen teksti syötettiin keskustelu neuroverkolle. Tästä saatava vastaus sittemmin syötettiin kielenkääntämisen neuroverkolle, joka käänsi vastauksen suomenkielelle. Lopuksi suomenkielinen vastaus lähetettiin palvelimelle käyttäen API-kirjastoa.

API-kirjasto sisälsi kutsujen lähetysfunktiot, joita botti siis tarvitsi kommunikoimiseen. Kolme tärkeintä funktiota olivat käyttäjän tunnuksien tarkistaminen, viestien noutaminen sekä näiden lähettäminen. Botin käynnistyessä ensimmäiseksi tarkistettiin tunnuksien toiminta käyttäen tunnuksien tarkistus polkua, joka otti parametreina käyttäjän salasanan sekä käyttäjätunnuksen. Gnu social-palvelimen viestit haettiin käyttäen public timeline polkua. Kyseisen polun kutsuun voitiin myös määrittää, kuinka monta viestiä haettiin palvelimelta. Viesteihin vastaaminen tapahtui käyttäen status update polkua, johon määriteltiin lähetettävä viesti sekä vastattavan viestin uniikki tunniste.

Vastausta luodessa botti pilkkoi haetun viestin sanat listaan. Tämän jälkeen lista muunnettiin numeraaliseen muotoon käyttäen koulutusvaiheessa luotua sanastolistaa. Mikäli noudettu viesti sisälsi tuntemattomia sanoja, joita botin koulutusdatassa

ei esiintynyt nämä merkittiin <UNK> merkillä. Kyseinen merkki on vain täytesana, jotta neuroverkko voisi käsitellä sanoja, joita se ei entuudestaan tiedä. Tämän jälkeen muunnettu sanalista syötetään neuroverkolle, ja sanoja poimittiin lauseeseen niin kauan kunnes neuroverkko tuotti <EOS> lopetusmerkin (Kuvio 30).



Kuvio 30. Botin käyttäminen terminaalista

7 Tulokset

Lopputuloksena saatiin melko lupaavia sekä hyödynnettäviä tuloksia. Botti saatiin toimimaan täysin itsenäisesti omassa toimintaympäristössään eikä se ollut riippuvainen ulkopuolisista palveluista keskustellakseen henkilöiden kanssa. Botti ymmärsi sekä englannin- että suomenkielisiä viestejä ja osasi myös antaa vastauksensa kyseisillä

kielellä. Kielenkääntämisen ominaisuus saatiin toimimaan jopa odotettua paremmin huomioon ottaen käytetyn koulutusdatan kokoon nähden.

Botin keskusteluominaisuus on raskaasti riippuvainen opetusdatan laadusta ja määrästä. Keskustelujen tuloksiin myös vaikuttaa datan monipuolisuus, mikäli halutaan luoda ihmismäinen vastaaminen. Koulutusdata täytyisi myös koostua normaaleista keskusteluista, jolloin botti vastaisi enemmän ihmismäisesti. Botin tehovaatimus yllätti positiivisesti, sillä sitä oli mahdollista ajaa pelkän prosessorin voimin, mikäli verkko oli valmiiksi koulutettu.

Attention mekanismin hyödyntäminen neuroverkossa testien mukaan paransi huomattavasti botin vastauksen tuottamista verrattuna versioon missä sitä ei käytetty. Botti osasi monipuolisemmin vastata erilaisiin syötteisiin sekä muotoilla lauseita järkevämmiksi. Tutkimuksessa käytettyä datajoukkoa käytettäessä huomasimme, että mikäli attention mekanismia ei käytetty botti pääsääntöisesti vastaili samankaltaisiin viesteihin melko samalla tavalla. Tässä toki joudutaan huomioimaan se, että datajoukko sisälsi paljon samankaltaisia vastauksia ja olivat rakenteiltaan samantyyppisiä. Tästä huolimatta attention mekanismia käytettäessä botti osasi selkeästi erottaa viesteistä tärkeitä kohtia ja näin ollen itse tuottamaan selkeästi eroavia ja järkevämpiä vastauksia aiempaan verrattuna.

8 Pohdinta

8.1 Keskusteluominaisuuden datajoukko

Keskusteluominaisuudessa käytetty datajoukko oli heikonpuoleinen, mikäli keskustelu poikkesi hotelliarvosteluista. Sillä datajoukko ei pitänyt sisällään normaalia keskustelumaista rakennetta, botille ei voitu näin ollen kirjoittaa keskustelumaisia viestejä vaan viestien täytyi olla arvosteluun liittyviä, jotta saatiin järkevä vastaus aikaiseksi. Datajoukolla oli kuitenkin helppo havainnollistaa propagandamainen viestintä negatiivisien arvostelujen ansiosta. Mikäli botti haluttaisiin pystyä tuottamaan normaalia keskustelumaista kanssakäymistä, tällöin voitaisiin ennemmin käyttää esimerkiksi foorumeilla esiintyviä keskusteluja koulutusdatajoukkoina.

8.2 Kielenkääntämisen datajoukko

Botin kielenkääntämisessä käytetty datajoukon oikeaoppinen lauserakenne eri lauseissa hieman vaihteli, sillä osa datasta käännettiin käyttäen Google kääntäjää ja kaikkia lauseita ei ehditty tarkistamaan vaan luotettiin Google kääntäjän käännösten oikeellisuuteen. Keskusteluominaisuuden datajoukkoon verrattuna kielenkääntäjän datajoukko oli kuitenkin parempi laatuinen, sillä se sisälsi paljon erilaisia lauseita eri aihealueista. Kielenkääntämisen datajoukko oli niin sanotusti helpompi kasata, sillä tarvittiin vain saada kasattua paljon erilaisia lauseita, joiden ei tarvinnut liittyä tiettyyn aihealueeseen. Kielenkääntäjässä myös beam search ominaisuus olisi tuonut paremman lopputuloksen kääntämiselle. Näin oltaisiin myös saatu luotua mahdollisia käännösvariaatioita, jotka tarkoittavat samaa mutta ovat esitettynä eri tavoin.

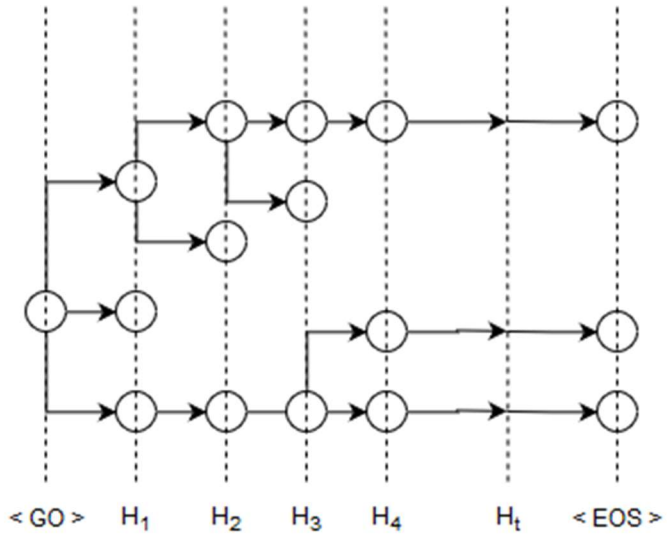
8.3 Keskusteluominaisuuden RNN-neuroverkko

Botin keskusteluominaisuuden RNN-neuroverkossa oltaisiin voitu myös testata tapaa, jossa encoderi sekä decoderi jakavat saman embedding-kerroksen. Tämä toimisi keskusteluominaisuuden RNN-neuroverkossa siitä syystä, että sekä encoderi ja decoderi teoriassa käsittelevät samaa asiaa eli englanninkielen lauseita. Näin embedding-kerrosta oltaisiin saatu koulutettua paremmin verrattuna erillisiin embedding-kerroksiin sekä embedding-kerroksen sanastotieto olisi kasvanut suuremmaksi. Botin jatkokehityksen kannalta olisi hyvä toteuttaa jatkokoulutus ominaisuus, jossa koulutetun verkon painotukset voitaisiin palauttaa ja näin ollen jatkaa koulutusta uudella datajoukolla. Kyseinen ominaisuus tapahtuisi käyttäen Tensowflowista löytyvää kaavion palautus ominaisuutta. Keskusteluominaisuudessa sanalistat olisi myös voitu yhdistää, jolloin ei olisi ollut encoderille sekä decoderille omia sanalistoja. Tämä vaatisi sen, että keskusteluominaisuudelle jouduttaisiin tällöin tekemään omaa sequence to sequence-neuroverkko, jota ei voitaisi käyttää kielenkääntämisessä.

8.4 Beam search

Botti käytti lauseen muodostamiseen niin sanottua greedy search menetelmää, joka valitsee todennäköisimmän sanan softmax-kerroksen antamasta vektorista. Tämän

tavan etuna on sen nopeus, mutta samalla lauseen muodostaminen saattaa kärsiä sillä välttämättä todennäköisin sana ei ole paras vaihtoehto. Beam search olisi tuonut etuna paremman lauseen muodostamisen, sillä se ottaa huomioon sanoille sopivia sanapareja. Beam search luo puumaisen rakenteen sanoille, jossa haarat ovat parhaimpia ehdotuksia lauseen seuraavaksi sanaksi. Yksinkertaisuudessaan beam search toimii niin, että decoderin ensimmäinen LSTM-solun tuloste syötetään normaalisti softmax-kerrokseen, josta saadaan sanojen todennäköisyysvektori. Tästä vektorista tallennetaan muistiin ennalta määritelty määrä sanoja. Tämän jälkeen decoderin seuraavalle LSTM-solulle syötetään muistiin tallennetut sanat yksi kerrallaan ja vertaillaan softmax-kerroksen todennäköisyysvektoreja. Näistä vektoreista poimitaan määritelty määrä todennäköisimpiä sanoja, jotka tallennetaan muistiin kyseisen aloitussanan jatkeeksi. Tätä jatketaan niin kauan, kunnes neuroverkko luo lopetusmerkin lauseille. Beam search voi muodostaa yhdelle sanalle useamman haaran ehdotettuja sanoja, mutta jokaisessa vaiheessa kerätään kuitenkin vain määritelty määrä sanoja (Kuvio 31). (Mishra 2018.)



Kuvio 31. Beam search, jossa muistiin tallennettava sanamäärä on kolme

Näin ollen, mikäli beam search tapaa olisi käytetty botin lauseen muodostaminen olisi saatu hienosäädettyä. Samalla oltaisiin mahdollisesti saatu saavutettua monipuolisempi vastaus kokonaisuus botille kuin mitä greedy search vaihtoehto tarjosi.

8.5 Havaintoja

Mielestäni tutkimuksessa saavutettiin haluttuja lopputuloksia sekä saatiin arvokasta tutkimusdataa chattibottien toteuttamisesta. Haastavaksi tutkimuksessa osoittautui oikeanlaisen datajoukon löytäminen ja rakentaminen, jotta propagandamainen informaatiovaikuttaminen saatiin selkeästi näkyviin viesteissä. Myös mikäli beam search ominaisuutta olisi käytetty botin tuottamat vastaukset olisivat voineet olla kieliopillisesti hieman parempia sekä viestien kontekstin muodostaminen olisi voinut parantua nykyiseen nähden.

Tutkimuksessa hyvänä lisänä olisi voinut myös olla selvittää valmiiksi koulutettujen embedding-kerroksien vaikutus botin tekstintuottamiseen nykyiseen toteutusmalliin nähden. Samalla olisi voinut vertailla näiden kahden mallin tuloksia keskenään ja havainnoida mahdollinen vaikutuksen suuruus. Tähän toki oltaisiin vaadittu suurempi datamäärä, jotta saataisiin luotettavia tuloksia. Valmiiksi koulutetuissa embedding-kerroksissa etuna tulisi botille heti aloituksessa laaja tietämys sanoista ja näiden suhteita toisiin sanoihin. Esimerkkinä GloVe, jossa suurimmassa tällä hetkellä saatavilla olevista koulutetuista malleista on käytetty 1,2 miljoonaa eri sanaa ja koulutus on tapahtunut 2 miljardilla Twitterin twiitillä.

Lähteet

A Beginner's Guide to Deep Reinforcement Learning. N.d. verkkoartikkeli. Viitattu 5.1.2019. https://skymind.ai/wiki/deep-reinforcement-learning

About Us. N.d. JYVSECTEC, verkkosivu. Viitattu 3.4.2019 https://jyvsectec.fi/about/overview/

Banerjee, S. 2018. An Introduction to Recurrent Neural Networks. Medium sivuston artikkeli. Viitattu 17.2.2019. https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912

Barack Obama is the Benchmark for Fake Lip-Sync. 2018. Medium sivuston artikkeli. Viitattu 17.2.2019. https://medium.com/syncedreview/barack-obama-is-the-benchmark-for-fake-lip-sync-videos-d85057cb90ac

Gupta, V. 2017. Understanding Feedforward Neural Networks. Learn OpenCv sivuston artikkeli. Viitattu 15.4.2019. https://www.learnopencv.com/understanding-feedforward-neural-networks/

Ivan, M. 2015. Types of machine learning algorithms. Proft.me sivuston artikkeli. Viitattu 1.4.2019. https://en.proft.me/2015/12/24/types-machine-learning-algorithms/

Jenna Abrams: the Trump-loving Twitter star who never really existed. 2017. Theguardian sivuston artikkeli- Viitattu 27.3.2019. https://www.theguardian.com/technology/shortcuts/2017/nov/03/jenna-abrams-the-trump-loving-twitter-star-who-never-really-existed

Koehrsen, W. 2018. Neural Network Embeddings Explained. Medium sivuston artikkeli. Viitattu 5.3.2019. https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526

Kurama, V. 2018. Reinforcement Learning with Python. Towards data science sivuston artikkeli. Viitattu 31.03.2019. https://towardsdatascience.com/reinforcement-learning-with-python-8ef0242a2fa2

Mishra, P. 2018. Beam Search - A Search Strategy. Hackernoon sivuston artikkeli. Viitattu 20.2.2019. https://hackernoon.com/beam-search-a-search-strategy-5d92fb7817f

Nguyen, M. 2018. Illustrated Guide to LSTM’s and GRU’s . Verkkoartikkeli. Viitattu 6.2.2019. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

One-Hot Encoding and Binning. 2018. Verkkoartikkeli. Viitattu 7.1.2019. https://evanlariviereblog.wordpress.com/2018/04/25/one-hot-encoding-and-binning/

Rouse, M. 2018. artificial neural network. search enterprise ai sivuston artikkeli. https://searchenterpriseai.techtarget.com/definition/neural-network

Saha, S. 2018. A Comprehensive Guide to Convolutional Neural Networks. towards data science sivuston artikkeli. Viitattu 14.4.2019. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Shubham, J. 2017. Introduction to Pseudo-Labelling : A Semi-Supervised learning technique. analytics vidhya sivuston artikkeli. Viitattu 14.4.2019. https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/

Tran, T. 2016. Creating A Text Generator Using Recurrent Neural Network. Verkkoartikkeli. Viitattu 18.2.2019. https://chunml.github.io/ChunML.github.io/project/Creating-Text-Generator-Using-Recurrent-Neural-Network/

top 15 most popular social networking sites. 2018. Dreamgrow sivuston artikkeli. Viitattu 14.11.2018. https://www.dreamgrow.com/top-15-most-popular-social-networking-sites/

Understanding LSTM Networks. 2015. Github blogin artikkeli. Viitattu 3.2.2019. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Van Loon, R. 2018. Machine learning explained. Verkkoartikkeli. Viitattu 3.2.2019. https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/

Liitteet

Liite 1. Chattibotti

JYVSECTEC
Jyväskylä Security Technology

Chattibotti

6.5.2019

jamk.fi
Institute of Information Technology

SISÄLTÖ

Kuviot

1 Lähtökohdat

Toimeksiantajana toimi Jyväskylä Security Technology, joka on itsenäinen kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus. Jyväskylä Security Technology toimii osana Jyväskylän ammattikorkean IT-instituuttia. Jyväskylä Security Technologyn osaamisalueita ovat kyberturvallisuus, tietoturvapoikkeamien hallinta, kehittyvät teknologiat ja informaatioteknologiat. (About Us n.d.)

Toimeksiantona oli luoda informaatiovaikuttamista chattibottien avulla RGCE:hen. Viestialustana toimi GNU social-palvelu, jossa bottien täytyi pystyä reagoimaan tuleviin viesteihin, sekä näiden vastaus toiminnallisuutta täytyi pystyä säätämään halutulla tavalla. Bottien haluttiin myös osata reagoida sekä suomenkielisiin että myös englanninkielisiin viesteihin. Mikäli viesti annettiin suomeksi, botin täytyi tällöin myös osata antaa oma viestinsä suomenkielisessä muodossa. Näin ollen botille jouduttiin toteuttamaan oma kielikääntäjä käyttäen neuroverkkoja, joka ei tukeutuisi ulkopuolisiin palveluihin.

Käytetty ohjelmointikieli sekä botin toiminnallisuuden toteutustapa oli vapaasti valittavissa. Botin koulutusdata kerättiin Kaggle nimisestä palvelusta, josta löytyy eri tyyppisiä ja valmiiksi kasattuja datapaketteja erilaisista tietolähteistä. Datajoukkoja koottiin ja muokattiin niin, että lopputuloksena koettiin saavan tarpeeksi sisältöä botin koulutukseen. Datajoukko täytyi saada kasattua monipuoliseksi, jotta botti saataisiin vastaamaan selkeästi sekä osaisi ymmärtää suomenkielistä ja englanninkielistä tekstiä.

2 Toteutus

2.1 Chattibotti

Chattibotti toteutettiin käyttäen python ohjelmointikieltä sekä Tensorflow koneoppimiskirjastoa. Tensorflow valittiin siksi, että siitä löytyy kattava kokoelma erilaisia valmiita koneoppimismoduuleja. Moduuleja oli helppo yhdistellä toisiinsa ja näin ollen mahdollista luoda monimutkaisiakin neuroverkkoja. Botti suunniteltiin toimimaan alustariippumattomasti käyttäen palveluiden ohjelmointirajapintoja, jolloin itse bottia ei oltu sidottu toimimaan vain yhdessä palvelussa. Näin bottia olisi myös mahdollista käyttää missä vain palvelussa, joka tarjoaa kelvollisen ohjelmointirajapinnan.

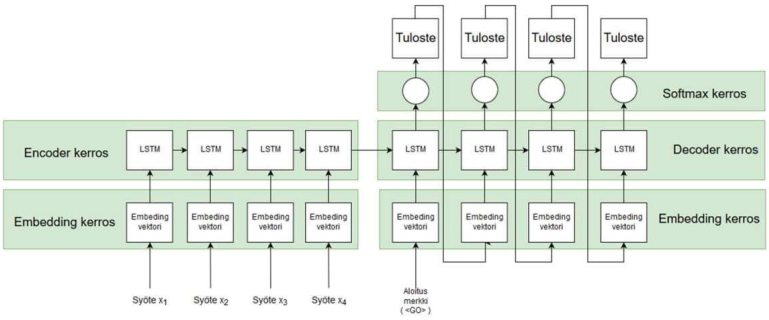
Botin koulutusdatana käytettiin Kagglesta ladattua hotelliarvostelu datajoukkoa. Datajoukosta suodatettiin vain arvostelut, jotka olivat positiivisia tai negatiivisia käyttäen python pandas kirjastoa. Pandas kirjastolla oli helppo käsitellä csv tiedostoja ja poimia haluttuja asioita. Positiiviset sekä negatiiviset viestit eriteltiin omiksi koulutusdatajoukoiksi. Botti käytti positiivisilla viesteillä koulutettua neuroverkkoa normaaliin kanssakäymiseen ja negatiivisilla viesteillä koulutettua neuroverkkoa propagandamaiseen viestintään. Kumpaankin koulutusdatajoukkoon loppujen lopuksi muodostui 8000 eri arvostelua per joukko, josta 1600 viestiä käytettiin neuroverkon koulutuksen testaamiseen ja arvioimiseen. Neuroverkkoa koulutettaessa sen koulutusdata ladataan tekstitiedostosta muistiin, josta jokaiselle erilaiselle sanalle annetaan uniikki kokonaisluku arvo. Sanojen muuntaminen uniikkeihin kokonaislukuihin tarvittiin siitä syystä, että neuroverkko ei ymmärrä tekstimuotoista dataa, sekä embedding-kerros pystyisi muuntamaan sanan vektoriesitys muotoon. Nämä uniikit numeroarvot tallennetaan sittemmin listaan, josta niitä voidaan hakea tarvittaessa. Kun jokaiselle sanalle on saatu annettua numeraalinen arvo, voidaan lauseet muuntaa numerolistaksi, jossa sana korvataan sille määritellyllä numerolla (ks. Kuvio 1).

```
1. Vocabulary = {the: 1, quick: 2, brown: 3, fox: 4, jumps: 5, over: 6, lazy: 7, dog: 8 … }
2. // Lause: the quick brown fox jumps over the lazy dog
3. sequences = [1, 2, 3, 4, 5, 6, 1, 7, 8]
```

Kuvio 1. Sanalista

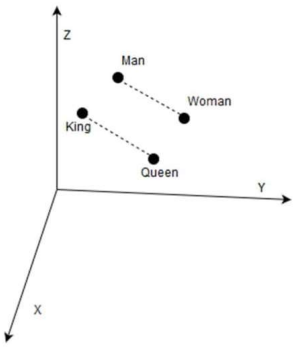
Tämän jälkeen kaikki numerolistat tasataan samanpituisiksi selvittämällä pisimmän lauseen datajoukosta. Numerolistojen tasaus tapahtuu lisäämällä täytemerkin numeraalinen arvo muunnettujen numerolistojen alkuun tai loppuun. Täytemerkki on varattu numerolle nolla. Tämän lisäksi muita erikoismerkkejä on aloitus- ja lopetusmerkki. Nämä merkit voivat olla mitä vain ja näitä käytetään määritellessä, milloin neuroverkko aloittaa tai lopettaa lauseen. Aloitusmerkki lisättiin jokaiseen nimikkodatan lauseen alkuun ja lopetusmerkki lisättiin lauseiden loppuun.

Botin neuroverkko arkkitehtuuri perustui sequence to sequence-malliin, jolloin lopullinen botin neuroverkko koostui kahdesta eri neuroverkosta (ks. Kuvio 2). Näitä kahta neuroverkkoa kutsutaan encoderiksi ja decoderiksi. Kummatkin neuroverkot koostuvat omista Long Short Term Memory (LSTM) -soluista, joita voitiin pinota päällekkäin näin halutessaan. Tämä mahdollistaa verkolle tarkemman datankäsittelyn sekä havainnoinnin, joka johtaa parempaan oppimiseen. Tässä kuitenkin pitää ottaa huomioon se, että mikäli verkko on liian suuri botti ei enää suoranaisesti opettele uusia asioita vaan pyrkii varastoimaan kaiken tiedon LSTM-soluihin. LSTM-solujen pinoamistoiminto oli helppo toteuttaa, sillä Tensorflowista löytyi valmis funktio tälle. Funktiolle vain tarvitsi kertoa haluttujen kerrosten määrä ja syöttää määritelty LSTM-solu. Sequence to sequence-malli valittiin toteutustavaksi, koska sillä oli hyvä toteuttaa sekä keskusteluominaisuus, kuin myös kielenkääntäminen. Sequence to sequence-malli onkin yksi käytetyimmistä tavoista luoda keskustelubotteja, kuin myös tehdä omia kielikääntäjiä.



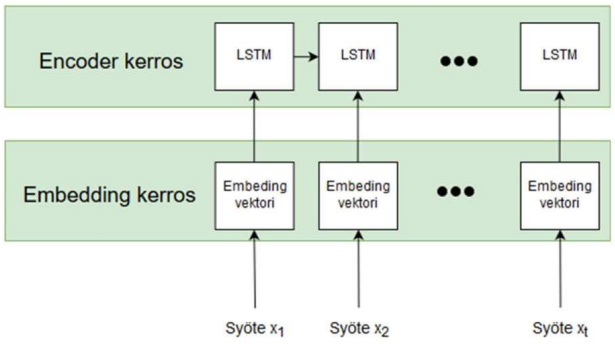
Kuvio 2. Encoder ja Decoder mallin kuvaus

Ennen encoderin LSTM-solu kerrosta koulutusdata syötettiin ensin embedding-kerrokselle, jonka toiminnallisuutena oli muodostaa sanoista sanavektoreja. Näin neuroverkko pystyisi helpommin hahmottamaan lauseiden rakennetta sekä lauseen kontekstia. Embedding-kerros mahdollistaa sanojen samankaltaisuuden tutkimisen, sillä samankaltaiset sanat esiintyvät kolmiulotteisessa koordinaatistossa lähempänä toisiaan verrattuna muihin sanoihin (ks. Kuvio 3). Esimerkkinä sanat kuningas ja mies esiintyisivät lähempänä toisiaan verrattuna sanoihin kuningatar ja nainen.



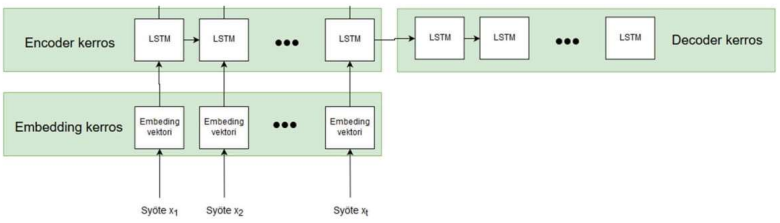
Kuvio 3. Sanat kolmiulotteisessa koordinaatistossa

Samalla embedding-kerroksella saavutettiin syötteen saaminen pienempään muotoon, sillä mikäli embedding-kerrosta ei olisi käytetty syöte olisi jouduttu muotoilemaan neuroverkolle one-hot-enkoodattuna vektorina. One-hot-enkoodattujen vektorien ongelmaksi chattiboteissa muodostuu niiden koko, jotka voivat kasvaa suuriksi, mikäli sanavarasto on suuri. One-hot-enkoodattu vektori myöskään ei ole laskennallisesti järkevä verrattuna embedding matriisiin. Embedding-kerrokselle olisi mahdollista ladata valmiiksi koulutettuja embedding-kerroksen painotuksia käyttäen esimerkiksi global vectors for word representationia (GloVe). Päädyimme kuitenkin testaamaan botin itsenäistä oppimista, jolloin kerroksen kouluttaminen aloitettiin tyhjästä. Näin ollen, kun bottia aloitetaan kouluttamaan embedding-kerros oppi sanoja ja näiden merkityksiä toisistaan koulutuksen yhteydessä. Embedding-kerroksen jälkeen, kun sanat on saatu muunnettua sanavektoreiksi, annetaan kyseinen syöte encoderin LSTM-soluille sanavektori kerrallaan (ks. Kuvio 4).



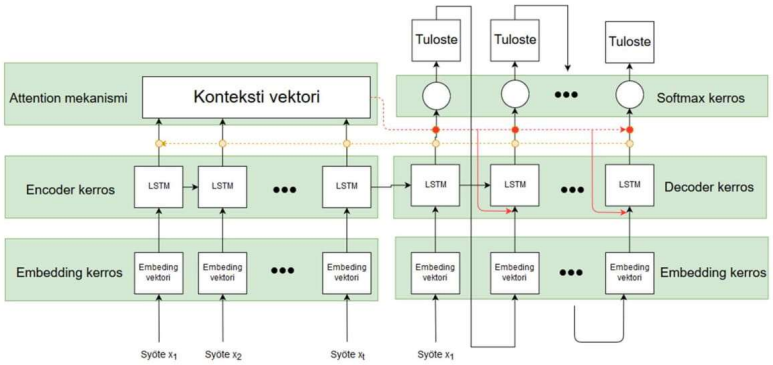
Kuvio 4. Embedding- ja LSTM kerros

LSTM-solu käsittelee syötteen ja laittaa muistiin asioita, joita se pitää tärkeänä. Tämän jälkeen LSTM-solu siirtää tilansa seuraavalle LSTM-solulle. Tätä tilaa sittemmin seuraava LSTM-solu käyttää hyväksi muodostaessaan tulostetta. Näin LSTM-solut saavat hyödynnettyä edellisien sanojen järjestystä sekä merkitystä lauseeseen. Encoderin avulla pystymme analysoimaan RNN-neuroverkolle tulevia syötteitä ja antamaan kaiken opitun tiedon decoderin LSTM-soluille (ks. Kuvio 5), jotka tuottavat vastauksen annetulle syötteelle. Näin decoderin ensimmäinen LSTM-solu, joka aloittaa vastauksen tuottamisen tietää encoderin syötteiden sisällöt ja tärkeimmät poimitut asiat syötteistä.



Kuvio 5. Encoderin tilan antaminen decoderin LSTM-solulle

Decoderi-kerrosta käytettäessä koulutusdata syötetään decoderin omalle embedding-kerrokselle. Toimenpide on täysin samanlainen kuin encoderin embedding-kerroksella, mutta syötettävä data on nimikkodatajoukko. Nimikkodatajoukko on siis niin sanotusti oikea vastaus minkä botin haluttaisiin antavan encoderilta saadun tilan perusteella. Embedding-kerroksen jälkeen decoderin ensimmäinen LSTM-solun tila alustetaan encoderin viimeisen LSTM-solun tilalla. Enne kuin decoderin LSTM-solu tuottaa lopullisen tulosteen sen hetkisestä LSTM-solusta saatava tuloste syötetään luong attention mekanismille. Attention mekanismia hyödynnettäessä encoderin LSTM-solujen tulosteisiin yhdistetään decoderin sen hetkisen LSTM-solun tuloste. Tämän jälkeen nämä yhdistevektorit pisteytetään ja luodaan niin sanottu kontekstivektori, josta saadaan selville mihin encoderin LSTM-solun tulosteen sanaan keskitetään eniten huomiota. Tämä on hyödyllistä sillä lauseen kannalta jokin encoderin sana saattaa olla useamman kerran tärkeämpi decoderin huomioida kuin muut sanat. Kontekstivektoria decoderi käyttää hyväksi, kun se syöttää lopullisen tulosteen softmax-kerrokselle. Tämän jälkeen konteksti vektori myös yhdistetään seuraavan LSTM-solun syötteen kanssa, jolloin sen hetkinen decoderin LSTM-solu tietää mihin encoderin sanaan viimeksi keskitettiin enemmän huomiota (ks. Kuvio 6). Tämä tapahtuu jokaisessa decoderin LSTM-solun vaiheessa, kunnes decoder on saanut tulostettua koko lauseen.

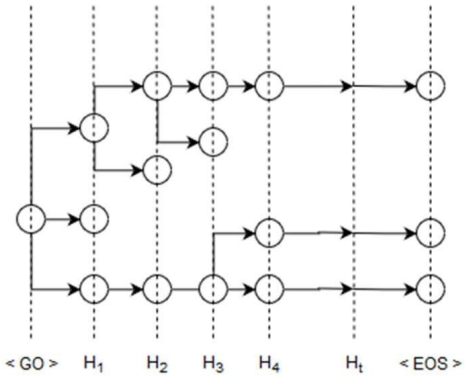


Kuvio 6. Attention mekanismi

Decoderin softmax-kerroksesta otetaan aina todennäköisin sana, joka on lopullinen lauseen sana. Suurimman arvon omaava sana ei välttämättä ole aina paras mahdollinen sana lauseen kannalta ja näin ollen tässä olisi voitu käyttää beam search menetelmää, jolloin sanoja olisi voitu poimia lauseen kannalta järkevimmin.

2.1.1 Chattibotin jatkokehitys

Suurimpana jatkokehityksenä chattibotille olisi beam search menetelmän lisääminen greedy search menetelmän sijasta. Beam search olisi tuonut etuna paremman lauseen muodostamisen, sillä se ottaa huomioon sanoille sopivia sanapareja. Beam search luo puumaisen rakenteen sanoille, jossa haarat ovat parhaimpia ehdotuksia lauseen seuraavaksi sanaksi. Yksinkertaisuudessaan beam search toimii niin että decoderin ensimmäinen LSTM-solun tuloste syötetään normaalisti softmax-kerrokseen, josta saadaan sanojen todennäköisyysvektori. Tästä vektorista tallennetaan muistiin ennalta määritelty määrä sanoja. Tämän jälkeen decoderin seuraavalle LSTM-solulle syötetään muistiin tallennetut sanat yksi kerrallaan ja vertaillaan softmax kerroksen todennäköisyysvektoreja. Näistä vektoreista poimitaan määritelty määrä todennäköisimpiä sanoja, jotka tallennetaan muistiin kyseisen aloitussanan jatkeeksi. Tätä jatketaan niin kauan, kunnes neuroverkko luo lopetusmerkin lauseille. Beam search voi muodostaa yhdelle sanalle useamman haaran ehdotettuja sanoja, mutta jokaisessa vaiheessa kerätään kuitenkin vain määritelty määrä sanoja (ks. Kuvio 7).



Kuvio 7. Beam search toiminta

Näin ollen, mikäli beam search tapaa olisi käytetty olisi botin lauseen muodostaminen saatu hienosäädettyä. Samalla oltaisiin mahdollisesti saatu saavutettua monipuolisempi vastauskokonaisuus botille kuin mitä greedy search vaihtoehto tarjosi. (Mishra 2018.)

Chattibotin kannalta myös oikeanlainen datajoukon rakentaminen on erittäin tärkeää, jotta botti osaisi tuottaa järkevää sekä ihmismäistä keskusteludataa. Loppujen lopuksi botin vastauskyky on kokonaan riippuvainen koulutusdatan laadusta sekä sen sisällöstä. Ihanteellisessa tapauksessa datajoukko tulisi koostua monipuolisista keskusteluista. Näin Botti oppii paremmin tunnistamaan ihmismäistä keskustelua sekä oppii tunnistamaan lauseista sanoja tai sana järkeistyksiä, jotka tarkoittavat samaa mutta ovat esitettynä eri muodossa. Samalla myös botti oppii tunnistamaan mitkä pääsanat muodostavat kyseisen lauseen ja kuinka sen tyyppisiin lauseisiin kuuluisi vastata. Datajoukko olisi myös hyvä koostua eripituisista lauseista. Mikäli kaikki lauseet ovat samanpituisia tai koostuvat pelkästään pitkistä lauseista saattaa botin olla vaikea tuottaa omaa vastausta, kun se saa suuresti poikkeavan syötteen verrattuna koulutusdataan. Samalla botti myös oppisi, että vaikka syötelause saattaa olla lyhyt tähän luotava vastaus saattaa olla pitkä. Koulutusdata määrältään olisi hyvä koostua suurikokoisista tekstitiedostoista, jotka olisivat sisällöltään kuitenkin laadukkaita. Mitä enemmän botille pystytään opettamaan sen parempaan ja hienosäädetympään lopputulokseen

päästäisiin. Hyvänä tutkimuksena olisi myös selvittää valmiiksi koulutettujen embedding-kerroksien vaikutus botin keskusteluominaisuuteen, kun koulutusdatajoukko on suurikokoinen. Keskusteluominaisuuden neuroverkossa oltaisiin voitu myös kokeilla yhteistä embedding-kerrosta. Tämä olisi nopeuttanut sekä lisännyt embedding-kerroksen koulutusta. Yhteinen embedding-kerros toimisi keskusteluominaisuudessa, sillä sekä encoderi että decoderi käsittelevät saman kielen lauseita ja sanoja.

2.2 Kielenkääntäjä

Botin kielenkääntämisen ominaisuus käytti tismalleen samaa sequence to sequence-mallin neuroverkkoa kuin keskusteluominaisuus. Tähän päädyttiin siitä syystä, että sequence to sequence toimii erinomaisesti kieltenkääntämisessä sen vaiheittaisen datan analysoinnin takia. Googlen oma kielenkääntäjä myös pohjautuu sequence to sequence malliin. Samaa mallia voitiin myös soveltaa siitä syystä, että keskusteluominaisuus sekä kielenkääntämisominaisuus ovat käytännössä samanlaisia toimintoja, jossa keskusteluominaisuus antaa vastauksen ja kääntäjä kääntää lauseen toiselle kielelle. Kielenkääntäminen tapahtuu, kun botti saa GNU social-palvelimelta suomenkielisen viestin. Viestien kieli koitetaan tunnistaa käyttäen pythonin kielentunnistus kirjastoa. Botti kääntää ensimmäiseksi suomenkielisen viestin englanninkielelle. Tämän jälkeen botti tuottaa vastauksen käyttäen englanninkielistä käännöstä. Lopuksi botti kääntää oman englanninkielisen vastauksensa suomeksi käyttäen kääntäjäänsä. Kielenkääntäminen jouduttiin toteuttamaan tällä tavalla siksi, että keskusteluominaisuus koulutettiin pelkästään käyttäen englanninkielistä koulutusdataa.

Botin kielenkääntämisen koulutusdata kerättiin monista eri lähteistä. Osa keskusteluominaisuuden koulutusdatasta myös käännettiin suomeksi käyttäen Google kääntäjää. Tämä tehtiin siitä syystä, että kääntäjän sanavarastoon saatiin kerättyä sanoja, jotka esiintyisivät keskusteluissa. Näin varmistuttiin siitä, että botin tuottaman vastauksen sanoista suurin osa pystyttäisiin tunnistamaan ja kyseinen lause näin ollen kääntämään järkevästi. Kielenkääntämisen koulutusdata käsiteltiin tismalleen samalla tavalla kuin keskusteluominaisuuden koulutusdata. Ensimmäiseksi data siis ladattiin tekstitiedostosta muistiin. Tämän jälkeen kummastakin kielestä poimittiin uniikit sanat, jotka tallennettiin sanalistaan ja samalla sanoille annettiin oma uniikki kokonaislukuarvo. Seuraavaksi

tekstimuotoiset lauseet muunnettiin numeraaliseen muotoon, jonka jälkeen lauseet tasattiin samanpituisiksi. Tässä vaiheessa siis data oli siinä muodossa, että sitä voitaisiin käyttää koulutuksessa. Encoderin data koostui käännettävän kielen sanoista ja decoderin data koostui käännetyn kielen sanoista.

2.2.1 Kielenkääntäjän jatkokehitys

Botin kielenkääntämisessä käytetty datajoukon oikeaoppinen lauserakenne eri lauseissa hieman vaihteli, sillä osa datasta käännettiin käyttäen Google kääntäjää ja kaikkia lauseita ei ehditty tarkistamaan vaan luotettiin Googlen kääntäjän käännösten oikeellisuuteen. Keskusteluominaisuuden datajoukkoon verrattuna kielenkääntäjän datajoukko oli kuitenkin parempi laatuinen, sillä se sisälsi paljon erilaisia lauseita eri aihealueista. Kielenkääntämisen datajoukko oli niin sanotusti helpompi kasata, sillä tarvittiin vain saada kasattua paljon erilaisia lauseita, joiden ei tarvinnut liittyä tiettyyn aihealueeseen. Kielenkääntäjässä myös beam search ominaisuus olisi tuonut paremman lopputuloksen kääntämiselle. Näin oltaisiin myös saatu luotua mahdollisia käännös variaatioita, jotka tarkoittavat samaa mutta ovat esitettynä eri tavoin.

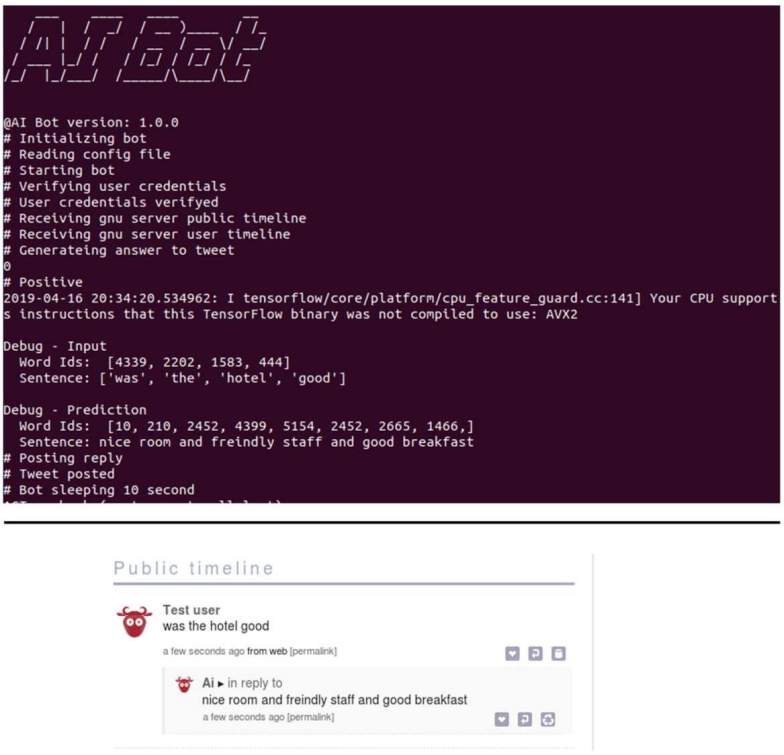
Kielenkääntäjän kannalta monipuolisen datajoukon rakentaminen lisää botin kielenkääntämisen tarkkuutta sekä parantaa monimutkaisien lauseiden kääntämistä. Kääntäjän datajoukko tulisi koostua monipuolisista lauseista, joiden ei tarvitse välttämättä kuulua tiettyyn aihealueeseen. Kielenkääntämisen datajoukossa on tärkeätä saada kasattua paljon dataa, jolloin botti oppii tunnistamaan esimerkiksi englanninkielen sanalle suomenkielisen vastineen ja kuinka sana kuuluu taivuttaa tai sijoittaa lauseeseen. Myös kielenkääntämisen datajoukko olisi hyvä koostua eripituisista lauseista samalla tavalla kuin keskusteluominaisuuden datajoukko. Tässäkin voitaisiin tutkia valmiiden embedding-kerroksien vaikutus koulutusnopeuteen sekä oppimiskykyyn. Tässä kuitenkin pitäisi huomioida se, että encoder sekä decoder vaatisivat omat valmiiksi koulutetut embedding-kerrokset. Tämä johtuu siitä, että kummankin embedding-kerrokset koostuvat eri kielien sanoista.

2.3 Toiminta

Botin käynnistämiseen tarvittiin komentoriviltä ajaa käynnistystiedosto. Botin keskusteluominaisuuden koulutus tapahtui ajamalla training kansiosta löytyvän training python

tiedoston. Koulutusdatajoukot sijoitetaan data kansioon, jossa input tekstitiedosto on encoderille syötettävät lauseet ja output tekstitiedosto on decoderille syötettävät lauseet. Koulutuksesta saatavat tiedostot voidaan siirtää tämän jälkeen modules kansiosta löytyviin moduuleihin. Itse bottia käytettäessä tarvittiin ajaa main python tiedosto, joka käynnisti botin toiminnan. Asetustiedostoon voitiin määritellä erilaisia asetuksia, kuten haluttu GNU social -palvelimen osoite sekä käyttäjä kyseiseen palveluun. Samalla voitiin myös määritellä mahdollinen haluttu propaganda sanalista, jolloin mikäli botti tunnistaa keskusteluista kyseisiä sanoja vaihtaisi botti käyttämään propagandalla koulutettua neuroverkkoa kyseiseen viestiin.

Kun botti oli käynnissä, se nouti määritellyn väliajoin viestejä GNU social-palvelimelta käyttäen omaa API-kirjastoaan (ks. kKuvio 8). Botti myös samalla piti kirjaa mitä viestejä oli haettu, jolloin se ei hakenut aina kaikkia viestejä. Vastausta luodessa botti kutsuu persoonallisuus interface python tiedostoa. Tässä python tiedostossa botti pilkkoo haetun viestin sanat listaan. Tämän jälkeen lista muunnetaan numeraaliseen muotoon käyttäen koulutusvaiheessa luotua sanastolistaa. Mikäli noudettu viesti sisälsi tuntemattomia sanoja, joita botin koulutusdatassa ei esiintynyt nämä merkittiin <UNK> merkillä. Kyseinen merkki on vain täytesana, jotta neuroverkko voisi käsitellä sanoja, joita se ei entuudestaan tiedä. Tämän jälkeen muunnettu sanalista syötetään neuroverkolle, ja sanoja poimitaan lauseeseen niin kauan kunnes neuroverkko tuottaa <EOS> lopetusmerkin. Tämän jälkeen tuotettu tuloste lähetetään GNU social-palvelimelle käyttäen botin API-kirjastoa.



Kuvio 8. Botin käyttäminen terminaalista

2.4 Ohjelmointirajapinta

Botti käyttää kommunikoimiseen täysin omaa API-kirjastoaan. Kirjastoon kerättiin tarpeelliset polut, joita botti tarvitsi kommunikoimiseen. Ensimmäiseksi kun botti käynnistetään, tarkistetaan GNU social-palvelimen tunnuksien toiminta käyttäen verify credentials polkua, joka ottaa parametreina käyttäjän salasanan sekä käyttäjätunnuksen (ks. kKuvio 9).

```
POST /api/account/verify_credentials.json
```

Parametrit:

Attribuutti	Tyyppi	Pakollinen	Seloste
username	String	Kyllä	Käyttäjätunnus
password	String	Kyllä	Käyttäjän salasana

Kuvio 9. Verify credentials polku

Gnu social-palvelimen julkiset viestit voitiin noutaa käyttäen APIn public timeline polkua. Kyseisen polun kutsuun voitiin myös määrittää kuinka monta viestiä haetaan palvelimelta (ks. kKuvio 10).

```
GET /api/statuses/public_timeline.json
```

Parametrit:

Attribuutti	Tyyppi	Pakollinen	Seloste
count	Integer	Ei	Haettavien viestien määrä
since_id	Integer	Ei	Haettavat viestit joiden ID on suurempi kuin annettu ID
max_id	Integer	Ei	Haettavat viestit joiden ID on pienempi kuin annettu ID

Kuvio 10. Public timeline polku

Viesteihin vastaaminen tapahtui käyttäen status update polkua, johon määriteltiin lähetettävä viesti sekä vastattavan viestin uniikki tunniste (ks. kKuvio 11).

```
POST /api/statuses/update.json
```

Parametrit:

Attribuutti	Tyyppi	Pakollinen	Seloste
status	String	Kyllä	Lähetettävä viesti
in_reply_to_status_id	Integer	Kyllä	Viestin ID johon vastataan

Kuvio 11. Viestiin vastaamisen polku

API-kirjastolla oli myös mahdollista botin lähettää omia viestejä käyttäen post status update polkua. Kirjastolla oli myös mahdollista hakea tietyn käyttäjän viestit käyttäen käyttäjän uniikkia tunnistetta. Näitä ominaisuuksia ei kuitenkin suoranaisesti käytetty, muutoin kuin viestien suodattamiseen joihin botti oli jo vastannut.

3 Keras versio

Botin ensimmäinen versio toteutettiin käyttäen Keras koneoppimiskirjastoa. Keras on korkeamman tason rajapinta, joka käyttää pohjana Tensorflow, Theano tai CNTK koneoppimiskirjastoja niin että itse näiden mainittujen kirjastojen toimintaan ei tarvitse itse perehtyä. Keras pääsääntöisesti pyrkii helpottamaan ja yksinkertaistamaan koneoppimiskirjastojen käyttöä. Vaikka Keras paljon helpottaa neuroverkkojen luomista havaittiin sen käyttö myös ongelmalliseksi myöhemmässä vaiheessa. Suurimmaksi ongelmasi Kerasissa muodostui tarpeellisien valmiiden koneoppimisfunktioiden puuttuminen ja näiden vaikea lisääminen Keras-neuroverkon malliin. Muutama näistä funktioista olivat beam search sekä eri attention mekanismit, jotka olivat käytettävissä Tensorflow koneoppimiskirjastossa.

Keras versiossa koulutusdata käsiteltiin samanlailla kuin Tensorflow versiossa. Koulutusdatan lauseet ladattiin siis ensin muistiin, jonka jälkeen näistä poimittiin uniikit sanat. Uniikeista sanoista luotiin

sanasto, jossa jokaiselle sanalle annettiin uniikki numeraalinen arvo. Tämän jälkeen lauseet muunnettiin numerolistoiksi, jossa numeron arvo edusti sanastolistasta löytyvää uniikkia numeroarvoa. Tämän jälkeen muunnetut lauselistat tasattiin samanpituisiksi käyttäen Kerasista valmiiksi löytyvää padding funktiota. Decoderin koulutusdatan lauseiden alkuun ja loppuun lisättiin aloitus- sekä lopetusmerkit.

Neuroverkko muodostui sequence to sequence mallin mukaisesti, jolloin chattibotti koostui encoder neuroverkosta ja decoder neuroverkosta. RNN-soluina käytettiin LSTM-soluja. Encoderin ensimmäinen kerros oli embedding-kerros, joka muunsi sanat sanavektoreiksi. Embedding-kerroksen jälkeen sanavektorit syötettiin LSTM-kerrokselle. Tensorflowin versioon poiketen LSTM-kerroksia ei voitu pinota päällekkäin. Kun encoderin LSTM-kerros on käsitellyt lauseet ja siirtänyt opitun tiedon aina seuraavalle LSTM-solulle, kerroksen viimeinen LSTM-solu siirsi opitut tiedot decoderin LSTM-kerrokselle. Decoderi koostui myös omasta embedding-kerroksesta, joka syötti sanavektorit decoderin LSTM-kerrokselle. Decoderin ensimmäinen LSTM-solu hyödyntää encoderin LSTM-soluista saatua tilaa luodessaan tulostetta. Decoderin LSTM-solu siirtää sittemmin opitun tiedon aina seuraavalle LSTM-solulle, kunnes lause on saatu kokonaan käsiteltyä. LSTM-solun tulosteet syötetään softmax-kerroksen lävitse, josta saadaan lopullisen vastaus. Tätä vastausta voidaan verrata sanalistan arvoihin, jolloin numeraalinen arvo saadaan muunnettua takaisin sanaksi. Kerasista ei löytynyt suoraan tukea sequence to sequence malliin kuten Tensorflowissa, joten tämä jouduttiin itse määrittelemään. Tämä tarkoitti sitä, että kerroksien syötteet jouduttiin käsin koodillisesti antamaan. Encoderin embedding-kerrokselle syötettiin siis muunnetut koulutuslauseet, jonka jälkeen embedding-kerroksen tuloste syötettiin LSTM-solulle. LSTM-solusta otettiin talteen sen tuloste sekä sisäinen tila, jotka asetettiin decoderin LSTM-solun tiloiksi. Decoderin embedding-kerrokselle määriteltiin syötteeksi nimikoidun datan koulutuslauseet, jonka jälkeen decoderin embedding-kerroksen tuloste syötettiin decoderin LSTM-solulle. Lopuksi määriteltiin softmax-kerros, joka antaa yhden vastauksen. Tämä vastaus sittemmin voitiin muuntaa sanaksi käyttäen sanalistaa. Keras version neuroverkko myös tallensi verkon joka koulutuskerta käyttäen Kerasin omaa checkpoint funktiota. Mikäli neuroverkkoa haluttiin jatkokouluttaa, voitiin aikaisempi koulutuksen tila ladata verkolle ja näin ollen jatkaa sen kouluttamista.

Toiminnaltaan siis Keras version neuroverkko ei poikennut Tensorflowin versioon nähden muutoin kuin, että se ei käyttänyt attention mekanismia. Näin ollen encoderin neuroverkko joutui luottamaan, että viimeinen LSTM-solu siirtää kaiken opitun tiedon decoderin ensimmäiselle LSTM-solulle. Myöskään decoderin kerros ei pystynyt samalla tavalla huomioimaan encoderin sanojen tärkeyttä luodessaan decoderin LSTM-solujen tulosteita verrattuna attention mekanismia hyödyntävään neuroverkkoon. Tämä ongelma ei välttämättä näy niinkään kieltä kääntäessä sillä yleisimmin sanoille on vain yhdentyyppinen vastaus, kun taas keskusteluja luodessa voidaan ajatella vastausten määrän olevan rajaton, jolloin ei ole olemassa yhtä ainoata oikeata vastausta. Myöskään beam search ominaisuutta Kerasista ei löytynyt, jolloin softmax-kerrokselta jouduttiin aina ottamaan suurimman arvon omaava sana. Tämä ei keskusteluja luodessa välttämättä ole paras tapa verrattuna beam searchin tuomaan puurakenne hakuun. Sillä Tensorflow sisälsi kyseiset ominaisuudet, päädyttiin botti toteuttaa käyttäen pelkästään Tensorflow koneoppimiskirjastoa.

4 Pohdinta

Mielestäni onnistuimme saamaan botin toimimaan odotusten mukaisesti, vaikka emme saaneetkaan beam search ominaisuutta toimimaan. Botti pystyi itsenäisesti kääntämään sekä luomaan vastauksia kysymyksiin, mikä oli haluttu lopputulos. Tutkimuksen aikana huomasimme, että varsinkin suomenkielisiä datajoukkoja oli hankala löytää. Myöskin propagandamaista datajoukkoja oli hankala löytää valmiina. Hotelliarvosteluilla oli helppoa esittää propagandamainen vastaaminen, mutta toki normaalissa käytössä se ei ole välttämättä hyödyllinen. Bottia kuitenkin voidaan opettaa millä tahansa koulutusdatalla ja sen vastauslaatu riippuneekin täysin koulutusdatan laadusta. Tästä huolimatta koimme, että pääsimme tavoitteisiimme sekä saimme tietoa mitä bottien tekemiseen vaaditaan. Samalla saimme tietoa minkälaista datajoukkoa vaadittaisiin, mikäli halutaan saada aikaiseksi tietyntyyppinen chattibotti. Myöskin lopullisessa Tensorflow versiossa attention mekanismin hyödyntäminen neuroverkossa testien mukaan paransi huomattavasti botin vastauksen tuottamista verrattuna Keras versioon missä sitä ei käytetty. Botti osasi monipuolisemmin vastata erilaisiin syötteisiin sekä muotoilla lauseita järkevämmiksi.

Lähteet

About Us. N.d. JYVSECTEC, verkkosivu .Viitattu 3.4.2019 https://jyvsectec.fi/about/overview/

Mishra, P. 2018. Beam Search - A Search Strategy. Hackernoon sivuston artikkeli. Viitattu 20.2.2019. https://hackernoon.com/beam-search-a-search-strategy-5d92fb7817f