Riku Kukkaniemi

Eerik Lehtomäki

Luonnollisen suomen kielen ymmärtäminen koneellisesti

Tietotekniikan pro gradu -tutkielma

19. huhtikuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijät: Riku Kukkaniemi ja Eerik Lehtomäki

Yhteystiedot: kukkaniemi.riku@gmail.com ja eerik.lehtomaki@gmail.com

Ohjaajat: Paavo Nieminen ja Sami Äyrämö

Työn nimi: Luonnollisen suomen kielen ymmärtäminen koneellisesti

Title in English: Understanding of natural Finnish language by computer

Työ: Pro gradu -tutkielma

Opintosuunta: Ohjelmistotekniikka

Sivumäärä: 64+7

Tiivistelmä: Tässä tutkimuksessa selvitettiin, miten luonnollisen kielen ymmärtämiseen rakennetut teknologiat soveltuvat suomen kielen käsittelyyn. Tutkimusosuuksissa selvisi, että vain harvat teknologioista tukevat suomen kieltä. Kielten tukitaso vaikutti perustuvan täysin palveluntarjoajien omaan käsitykseen kielituen laajuudesta.

Teknologioiden isoimmaksi ongelmaksi muodostui suomen kielen kohdalla taivutusmuodossa olevien sanojen käsittely. Teknologiat pystyivät käsittelemään sanoja ainoastaan siinä muodossa, jossa sanat oltiin teknologioille opetettu. Tämä tarkoittaa sitä, että teknologioiden toiminta suomen kielellä vaatisi kattavan opetusdatan, jossa tulisi ottaa tunnistettavien sanojen lisäksi huomioon kaikki sanojen taivutusmuodot. Tutkimuksessa tähän ongelmaan löytyi ratkaisu lemmauksesta, jonka avulla sanat pystyttiin muuttamaan perusmuotoon ennen teknologioiden käsittelyä.

Avainsanat: Luonnollisen kielen käsittely, Luonnollisen kielen ymmärtäminen, NLP, NLU, Tekoäly, Keskustelubotti, Intentio, Entiteetti, Koneoppiminen, Neuroverkko, Sanaluokittelu, Saneistus, Normalisointi, Lemmaus, Stemmaus, Dialogflow, Wit.ai, LUIS, Watson Assistant, Amazon Lex, Recast.ai, Rasa, Snips

Abstract: This study investigated how technologies built for understanding natural language are applicable to Finnish language processing. The research revealed that only a few technologies support the Finnish language. The level of language support seemed to be based entirely on service providers' own perception of the scope of language support.

The biggest problem with technologies in the Finnish language was the processing of inflectional forms of words. Technologies could only handle words in the form in which the words were taught to the technologies. This means that the operation of technologies in the Finnish language would require comprehensive instructional data, which should include not only identifiable words but also any possible inflectional form. The study found a solution to this problem in lemmatisation, which allowed words to be transformed into their basic form before the technologies processed them.

Keywords: Natural language processing, Natural language understanding, NLP, NLU, Artificial intelligence, Chatbot, Intent, Entity, Machine learning, Neural network, Part of speech tagging, Tokenization, Normalization, Lemmatisation, Stemming, Dialogflow, Wit.ai, LUIS, Watson Assistant, Amazon Lex, Recast.ai, Rasa, Snips

Termiluettelo

Affiksi	Sanan runkoon liitettävä osa, esimerkiksi taivutusmuodot.
Agglutinatiivinen kieli	Kieli, jossa sanojen vartaloon liittyy useita osia.
Algoritmi	Algoritmi on yksiselitteinen kuvaus tai ohje, jota seuraamalla ongelma ratkaistaan.
Antonyymi	Sana, joka on jonkin toisen sanan vastakohta.
API	Ohjelmointirajapinta (engl. *Application Programming Interface*).
Entiteetti	Intentiota tarkentava avainsana.
Esperanto	Luonnolliseen kieleen perustuva keinotekoinen apukieli.
Homofoni	Sana, joka ääntyy samalla tavalla toisen sanan kanssa, mutta jolla on eri merkitys.
Homografi	Sana, joka on kirjoitusasultaan toisen sanan kaltainen, mutta äänteellisesti ja merkitykseltään siitä poikkeava.
Intentio	Käyttäjän pyrkimys tai asia, jota ilmaisussa tavoitellaan.
Iteraatio	Iteroinnissa toistettavaa työvaihe.
Iterointi	Yleinen nimitys menetelmille, joissa samoja työvaiheita toistetaan, kunnes haluttu tulos on saavutettu.
Keinotekoinen kieli	Kieli, jonka kielioppi ja sanasto on luotu tietoisesti.
Kielikunta	Ryhmä kieliä, jotka periytyvät samasta kantakielestä.
Kontekstuaalinen	Asiayhteyttä koskeva.
Leksikaalinen	Sanakirjan mukainen.
Lemma	Sanamuotoja yhdistävä perusmuoto.
Lemmaus	Normalisointialgoritmi, joka etsii sille määritellystä tietokannasta sanojen lemmat.
Luonnollinen kieli	Kieli, jolla ihmiset yleisesti kommunikoivat keskenään.
Matriisi	Matemaattinen taulukko, jossa rivien ja sarakkeiden alkiot ovat lukuja tai lausekkeita.
Morfeemi	Morfeemi on kielen pienin merkitystä kantava yksikkö. Vartalo, monikon tunnus ja sijapääte ovat morfeemeja.
Morfologinen analyysi	Analyysi kielen sisältämistä morfeemeista.

Neuroverkko	Laskennan malli, joka perustuu yhdistävään laskentaan. Matkii aivojen hermoston toimintaa.
NLG	Luonnollisen kielen tuottaminen (engl. *Natural Language Generation*).
NLP	Luonnollisen kielen käsittely (engl. *Natural Language Processing*).
NLU	Luonnollisen kielen ymmärtäminen (engl. *Natural Language Understanding*).
Päätöspuu	Päätöspuu koostuu solmuista, jotka kuvaavat mahdollisia päätöksiä ja ulostuloja.
Sanaluokittelu	Jäsennysmenetelmä, jonka tavoitteena on ratkaista, mikä sanaluokkakoodi kullekin sanalle kuuluu.
Sane	Sane on sanamuodon esiintymä. Saneiden määrällä voidaan mitata tekstin pituutta.
Saneistus	Tekstin paloittelu saneiksi.
SDK	Kokoelma työkaluja tai ohjelmia, joita kehittäjät voivat käyttää sovellusten rakentamiseen tietyille alustoille (engl. *Software Development Kit*).
Stemmaus	Normalisointialgoritmi, joka poistaa sanoista niiden taivutusmuotoon viittaavat kirjaimet.
Tekoälyteknologia	Järjestelmä tai kokonaisuus, joka kykenee älykkäiksi pidettäviin toimintoihin.
Tensori	Tensori on matematiikassa tietyn tyyppinen geometrinen kokonaisuus tai vaihtoehtoisesti yleinen suure.
Vektori	Vektori on vektoriavaruuden alkio, joka voidaan esittää kiinteänä määränä reaalilukuja eli komponentteja.
Vektoriavaruus	Matemaattinen joukko, johon on määritelty alkioiden summa ja skalaarilla kertominen tietyllä tavoin.

Kuviot

Kuvio 1. Tilakaavio NLU:ta ja NLG:tä hyödyntävästä keskustelubotista. 4
Kuvio 2. Google Translate osaa käsitellä suomenkielistä puhetta ja tekstissä esiintyviä virheitä. 15
Kuvio 3. Älykaiuttimet ovat yleistyneet ihmisten kotona. Kuvassa Google Home Mini -älykaiutin. Kuva Eerik Lehtomäki. 16
Kuvio 4. Seurarobotti Pepperiä on testattu jo muutamissa terveyskeskuksissa. Kuva (*Pepper Robot* 2020). 17
Kuvio 5. Lauseen muuttaminen vektoriksi one-hot encoding -menetelmällä. 21
Kuvio 6. Upotusvektorit kaksiulotteisessa vektoriavaruudessa. 22
Kuvio 7. Kontekstin muodostus Skip-Gram -mallissa. 23
Kuvio 8. Dialogflown käyttöliittymä 09/2018. 25
Kuvio 9. Wit.ai:n käyttöliittymä 10/2018. 26
Kuvio 10. LUIS käyttöliittymä 10/2018. 28
Kuvio 11. Watson Assistantin käyttöliittymä 10/2018. 29
Kuvio 12. Amazon Lex käyttöliittymä 1/2019. 31
Kuvio 13. Recast.ai käyttöliittymä 1/2019. 32
Kuvio 14. Snipsin käyttöliittymä 1/2019. 35
Kuvio 15. Suomen kielellä testatut NLU-teknologiat. 38
Kuvio 16. Tilakaavio testaustyökalun toiminnasta. 43
Kuvio 17. Testaustyökalun käyttöliittymä. 44
Kuvio 18. Testaustyökalun toiminta. 46
Kuvio 19. Entiteettien tunnistustarkkuus syötteistä. 47
Kuvio 20. Syötteiden prosessointinopeudet keskiarvoina ja -hajontoina. * $p < 0{,}05$ ja ** $p > 0{,}05$. 48

Sisältö

1 JOHDANTO 1

2 LUONNOLLISEN KIELEN KÄSITTELY 3
2.1 Historia 4
2.2 Saneistus 5
2.3 Sanaluokittelu 8
2.4 Normalisointi 9
2.4.1 Stemmaus 10
2.4.2 Lemmaus 11
2.4.3 Suomen kielen normalisointi 12

3 LUONNOLLISEN KIELEN YMMÄRTÄMINEN 14
3.1 Käyttökohteet 14
3.2 Teknologioiden opetus 18
3.3 Keskustelubottien kehitys 19

4 TEKNOLOGIOIDEN TOIMINTA SUOMEN KIELELLÄ 24
4.1 Dialogflow 24
4.2 Wit.ai 26
4.3 LUIS 27
4.4 Watson Assistant 29
4.5 Amazon Lex 30
4.6 Recast.ai 32
4.7 Rasa 33
4.8 Snips 34
4.9 Johtopäätökset 36

5 WATSON ASSISTANTIN JA WIT.AI:N SUORITUSKYKY SUOMEN KIELEN KÄSITTELYSSÄ 40
5.1 Suorituskykyä mittaavat tutkimukset 40
5.2 Tutkimusasetelma 42
5.3 Testaustyökalu 43
5.4 Entiteettien tunnistustarkkuus 46
5.5 Prosessointinopeus 47
5.6 Johtopäätökset 49

6 YHTEENVETO JA POHDINTA 51

LÄHTEET 53

LIITTEET 57
A Tutkimuksessa teknologioille opetetut entiteetit 57
B Tutkimuksessa käytetyt syötteet 58
C Rajapintatoteutus Wit.ai ja Watson Assistant -teknologioille 62

D Rajapintatoteutus Finnish-dep-parserille 63

1 Johdanto

Ensimmäisten tietokoneiden myötä tutkijoita alkoi kiinnostamaan ajatus siitä, pystyisikö tietokone ajattelemaan ja käyttäytymään kuten ihminen. Jotta tietokoneen ja ihmisen välinen keskustelu olisi mahdollista, täytyisi tietokoneen ensin osata käsitellä ihmisten käyttämää luonnollista kieltä. (Turing 2009) Viime vuosikymmenen aikana useat teknologiayritykset ovat kehittäneet luonnollisen kielen tunnistamista (engl. Natural Language Understanding, NLU) helpottavia NLU-teknologioita, joilla ihmisten käyttämä kieli muutetaan tietokoneelle ymmärrettävään muotoon (Hänninen ym. 2018).

NLU-teknologioiden soveltuvuutta käsitellä suomen kieltä on toistaiseksi tutkittu erittäin vähän. Watson Health Cloud Finland -projektissa Jyväskylän yliopiston tutkimusryhmä toteutti vuoden 2018 aikana viisi prototyyppisovellusta, joiden tarkoituksena oli tutkia pystytäänkö terveydenhuollon prosesseja ja ihmisten terveyttä edistämään erilaisilla tekoälyteknologioilla (Hänninen ym. 2018). Tutkimusryhmän toteuttamissa prototyyppisovelluksissa hyödynnettiin NLU-teknologioita suomen kielen käsittelyssä (Hänninen ym. 2018). Tämä tutkielma jatkaa Watson Health Cloud Finland -projektin (Hänninen ym. 2018) tutkimusta tuottaen omalta osaltaan uutta tietoa siitä, miten NLU-teknologiat soveltuvat suomen kielen käsittelyyn.

Watson Health Cloud Finland -projektin (Hänninen ym. 2018) yhtenä loppupäätelmänä oli se, että projektissa käytetyt NLU-teknologiat pystyvät käsittelemään suomen kieltä, mutta niiden vaatiman opetusdatan täytyy olla kattava, ja kattavan opetusdatan toteuttaminen on todella työlästä. Tässä tutkielmassa selvitettiin Watson Health Cloud Finland -projektissa (Hänninen ym. 2018) käytettyjen Wit.ai- ja Watson Assistant -teknologioiden lisäksi muiden yleisesti käytössä olevien NLU-teknologioiden kykyä käsitellä suomen kieltä. Watson Health Cloud Finland -projektissa (Hänninen ym. 2018) myös todettiin, että NLU-teknologioiden opetusdatan tarvetta voitaisiin vähentää normalisoimalla syötteiden sanat niiden perusmuotoon. Tämän huomion perusteella tutkielmassa haluttiin selvittää, miten teknologioiden suorituskyky muuttuu normalisoinnin avulla.

Tutkielman tutkimusosuus on jaettu kahteen osaan. Ensimmäisessä osuudessa NLU-teknolo-

gioista annetaan kattava kuvaus sovelluskehittäjää kiinnostavasta näkökulmasta. Käsiteltäviksi NLU-teknologioiksi valittiin Dialogflow, Wit.ai, LUIS, Watson Assistant, Amazon Lex, Recast.ai, Rasa ja Snips. Jokaisella tutkimukseen valitulla teknologialla toteutettiin suomen kielellä toimiva keskustelubotti, joka koulutettiin sekä testattiin suomenkielisellä testiaineistolla. Tutkimusosuudessa vastattiin tutkimuskysymykseen "Minkälaisia ominaisuuksia NLU-teknologiat sisältävät ja miten teknologiat tukevat suomen kieltä?".

Toisessa tutkimusosuudessa testattiin Watson Assistantin ja Wit.ai:n suorituskykyä käsitellä suomen kieltä. Tutkimusosuudessa testattaville teknologioille annettiin erilaisia syötteitä, joista teknologioiden tuli tunnistaa opetettuja entiteettejä. Tutkimusosuudessa tutkittiin entiteettien tunnistustarkkuutta ja syötteiden prosessointinopeutta. Tutkimusosuudessa myös selvitettiin, miten teknologioiden suorituskyky muuttuu, kun syötteet normalisoidaan ennen teknologioille vientiä. Toisessa tutkimusosuudessa vastattiin tutkimuskysymykseen "Miten Watson Assistant ja Wit.ai suoriutuvat suomen kielellä ja miten normalisointi vaikuttaa niiden suoriutumiseen?".

Luvussa 2 kerrotaan, mitä luonnollisen kielen käsittely tarkoittaa ja minkälaisia menetelmiä se pitää sisällään. Luvussa 3 syvennytään siihen, mihin luonnollisen kielen ymmärtämistä käytetään, miten luonnollista kieltä ymmärtäviä teknologioita opetetaan ja miten kieltä ymmärtäviä keskustelubotteja voidaan kehittää. Luku 4 on tutkielman ensimmäinen tutkimusosuus, jossa tutkitaan NLU-teknologioiden ominaisuuksia ja miten teknologiat tukevat suomen kieltä. Luku 5 on tutkielman toinen tutkimusosuus, jossa tutkitaan Watson Assistantin ja Wit.ai:n suorituskykyä käsitellä suomen kieltä, sekä miten teknologioiden suoriutuminen muuttuu normalisoinnin myötä. Luvussa 6 kerrataan tutkielman keskeisimmät asiat ja käydään läpi tuloksiin liittyvää pohdintaa.

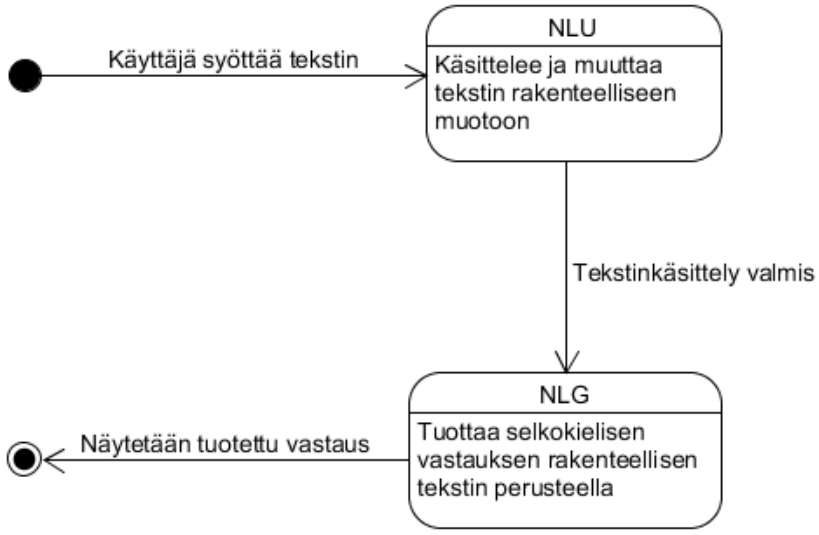
Lehtomäki vastasi luonnollista kieltä ymmärtävien teknologioiden kartoituksesta ja ensimmäisestä tutkimusosuudesta, joka sisälsi teknologioiden toimintojen sekä keskustelubottien suomen kielen tuen selvityksen. Kukkaniemi perehtyi luonnollisen kielen käsittelyn menetelmiin ja vastasi toisesta tutkimusosuudesta eli Watson Assistantin ja Wit.ai:n suorituskyvystä käsitellä suomen kieltä. Tutkielma tehtiin vastuualueista huolimatta tiiviissä yhteistyössä.

2 Luonnollisen kielen käsittely

Luonnollisella kielellä tarkoitetaan kieltä, jolla ihmiset kommunikoivat keskenään joko puheen tai merkkien avulla (Lyons 1991). Luonnollinen kieli on kehittynyt ihmisten käytössä ja siirtynyt sukupolvelta toiselle sosiaalisen kanssakäymisen kautta (Lyons 1991). Luonnollinen kieli on laaja käsite, minkä vuoksi termin sisältö kannattaa aina määritellä sen käytön yhteydessä. Tässä tutkielmassa luonnollista kieltä tarkastellaan vapaamuotoisena puhekielenä tai tekstinä, jolla ihmiset kommunikoivat sosiaalisissa kanssakäymisissä. Luonnollinen kieli nähdään tutkielmassa jatkuvasti muokkaantuvana sen mukaan, miten ihmiset päättävät kommunikoida.

Luonnollisen kielen vastakohtana pidetään usein keinotekoista kieltä (Stenlund 1990). Keinotekoisia kieliä ovat esimerkiksi ohjelmointikielet sekä erilaiset elokuviin tai sarjoihin luodut kielet kuten Star Trekissä käytetty klingonin kieli. Keinotekoisen kielen merkittävin ero luonnollisen kieleen on se, että keinotekoisen kielen kielioppi ja sanasto on luotu tietoisesti, kun taas luonnollinen kieli on kehittynyt ihmisten vuorovaikutuksen kautta (Lyons 1991; Stenlund 1990).

Luonnollisen kielen käsittelyllä (engl. *Natural Language Processing, NLP*) tarkoitetaan tietokoneen kykyä prosessoida ihmisten tuottamaa luonnollista kieltä (Sumathy ja Chidambaram 2013). Luonnollisen kielen käsittely ei rajoitu ainoastaan tekstin käsittelyyn, vaan siihen kuuluu sekä kirjallisen että puhutun kielen analysointi (Sumathy ja Chidambaram 2013). Luonnollisen kielen käsittely on tutkimusalueena laaja, minkä vuoksi se jaetaan yleensä pienempiin alakategorioihin. Yleisimmät alakategoriat ovat luonnollisen kielen ymmärtäminen ja luonnollisen kielen tuottaminen (engl. *Natural Language Generation, NLG*). NLU:ssa tietokone pyrkii tulkitsemaan luonnollista kieltä, löytämään kielestä erilaisia asiayhteyksiä ja ymmärtämään kielen sisältöä (Navigli 2018). NLG:ssä tietokone pyrkii rakenteisen tiedon avulla tuottamaan ihmiselle selkolukuista luonnollista kieltä (Perera ja Nand 2017). NLU:ta ja NLG:tä voidaan käyttää esimerkiksi keskustelubotissa, jossa NLU:lla pyritään ymmärtämään ihmisen syöttämää tekstiä ja sen perusteella NLG:llä tuottamaan ihmiselle selkokielinen vastaus. Kuviossa 1 on esitetty tilakaavio siitä, miten keskustelubotti voisi käsitellä käyttäjän syötteen NLU:n ja NLG:n avulla.



Kuvio 1. Tilakaavio NLU:ta ja NLG:tä hyödyntävästä keskustelubotista.

2.1 Historia

Ensimmäisiä kielen kääntämiseen tarkoitettuja koneita aloitettiin patentoimaan 1930-luvulla, jolloin Georges Artsrouni esitteli yksinkertaisella reikänauhalla toimivan laitteen, joka toimi saman tapaisesti kuin kaksikielinen sanakirja. Vuonna 1933 Peter Troyanskii esitti Artsrounin laitetta hieman tarkemman kielen kääntämiseen tarkoitetun laitteen. Troyanskiin laite sisälsi kaksikielisen sanakirjan lisäksi menetelmän, joka sovelsi keinotekoisen esperanto kielen kielioppeja. (Hutchins 2004) Edellä mainittuja sekä muita 1930-luvun puolivälissä kielen käsittelyyn kehitettyjä laitteita voidaan pitää alkuna luonnollisen kielen käsittelyn kehittymiselle (Hutchins 2004, 2003).

Erilaisten laitteiden keksimisen lisäksi vuonna 1957 kielitieteilijä Noam Chomskyn julkaisemaa teosta Syntactic Structures voidaan pitää merkittävänä luonnollisen kielen käsittelyn kehitykselle (Chomsky 1957). Chomskyn teoksessa esiteltiin syntaktisiin rakenteisiin sekä sääntöihin perustuvaa kieliopillista näkemystä kielen rakentumisesta, mikä auttoi yhä useampia ymmärtämään kielen rakenteita (Chomsky 1957; Knuth 2003). Syntaktisten rakenteiden kehittyessä huomattiin tietoteknisten laitteiden kyvykkyys kielen käsittelyssä, mikä toi kielitieteen sekä tietotekniikan aloja lähemmäksi toisiaan (Chomsky 1957; Knuth 2003).

Useimmat NLP-järjestelmistä perustuivat pitkään monimutkaisiin käsin kirjoitettuihin sääntöihin, kunnes 1980-luvulla luonnollisen kielen ymmärtämisen saralla alkoi vallankumous erilaisten koneoppimisalgoritmien sekä metodien kehittymisen myötä. Näihin aikoihin esiteltiin muun muassa takaisinkytkeytyvä neuroverkko (Hopfield 1982), vastavirta-algoritmi (Rumelhart, Hinton ja Williams 1986) sekä vahvistusoppiminen (Rumelhart, Hintont ja Williams 1986). Edellä lueteltujen algoritmien kehittyminen oli suuri askel luonnollisen kielen käsittelylle, vaikkakin nykyisissä kielen tutkimuksissa keskitytäänkin yhä enemmän koneoppimisesta tuttuihin puoliksi ohjattuihin sekä ohjaamattomiin oppimisalgoritmeihin (Russell ja Norvig 2016). Koneoppimisella viitataan tutkimusalaan, jonka tarkoituksena on kehittää tietokoneohjelmille kykyä oppia ilman erillistä ohjelmointia (Samuel 1959).

Yksi syy puoliksi ohjattujen sekä ohjaamattomien oppimisalgoritmien käytölle luonnollisen kielen käsittelyssä on se, että kyseisiä algoritmeja voidaan kehittää helpommin opetusaineistoa lisäämällä (Russell ja Norvig 2016). Ohjaamattomassa oppimisessa algoritmia voidaan kehittää jopa sellaisella opetusaineistolla, johon ei ole merkitty ennalta oikeita tai vääriä vastauksia (Russell ja Norvig 2016; Lee, Shin ja Realff 2017). Ohjaamattomat oppimisalgoritmit tuottavat tyypillisesti kuitenkin vähemmän tarkkoja tuloksia pienellä määrällä opetusdataa verrattuna ohjattuihin ja puoliksi ohjattuihin oppimisalgoritmeihin (Russell ja Norvig 2016; Lee, Shin ja Realff 2017). Tekstisisältöistä opetusaineistoa on kuitenkin usein runsaasti saatavilla, mikä on eduksi ohjaamattomalle oppimiselle (Lee, Shin ja Realff 2017).

2.2 Saneistus

Suurimmassa osassa NLP-tutkimuksista keskitytään tekstin analysoinnin tai tuottamisen menetelmiin, jättäen kuitenkin huomioimatta perusyksiköt eli miten yksittäiset sanat tulisi käsitellä. Ilman perusyksiköiden käsittelyä on mahdotonta suorittaa monimutkaista tekstin analyysia tai tekstin tuottamista. (Webster ja Kit 1992) Tämän vuoksi saneistus (engl. *tokenization*) on yleensä ensimmäinen menetelmä, joka luonnollisen kielen käsittelyssä tekstille toteutetaan (Proisl ja Uhrig 2016; Hassler ja Fliedl 2006; Habert ym. 1998; Jiang ja Zhai 2007).

Saneistuksen tavoitteena on paloitella käsiteltävä teksti pienemmiksi helpommin käsiteltä-

viksi yksiköiksi (Hassler ja Fliedl 2006; Habert ym. 1998; Jiang ja Zhai 2007; Proisl ja Uhrig 2016). Saneistuksen ensimmäisessä vaiheessa käsiteltävä teksti jaetaan lauseisiin ja toisessa tunnistetaan lauseiden saneet (Proisl ja Uhrig 2016). Saneeksi luokitellaan merkkijono, joka on tulostettavissa oleva kokonaisuus. Sane voi olla esimerkiksi sana, numero, internet-osoite tai lyhenne. Sane ei voi olla kuitenkaan tyhjä merkkijono, välilyönti tai rivinvaihto. (Hassler ja Fliedl 2006)

Saneistukselle ei yleensä anneta huomiota tutkimuksissa, sillä sitä pidetään liian yksinkertaisena tai jopa triviaalina prosessina (Webster ja Kit 1992; Proisl ja Uhrig 2016; Hassler ja Fliedl 2006; Habert ym. 1998). Yleisesti voidaan olettaa, että saneet saadaan määriteltyä lauseesta erottelemalla lauseen sanat toisistaan välilyöntien ja rivinvaihtojen perusteella (Hassler ja Fliedl 2006; Jiang ja Zhai 2007). Saneiden määrittelyssä tulisi kuitenkin huomioida kieleen liittyvät sanariippuvuudet ja muut kielen erityispiirteet, minkä vuoksi saneistuksen tulisi perustua jonkinlaiseen kaavaan tai malliin (Webster ja Kit 1992; Hassler ja Fliedl 2006).

Saneistusalgoritmi aloittaa jakamalla tekstin lauseet omiksi kokonaisuuksiksi. Ihmiset pystyvät helposti hahmottamaan, milloin lause alkaa ja milloin se päättyy, mutta sen opettaminen tietokoneelle ei ole niin yksinkertaista kuin voisi olettaa. Lause alkaa isolla alkukirjaimella ja loppuu päätemerkkiin, joka voi olla piste, huutomerkki tai kysymysmerkki. Piste voi kuitenkin esiintyä lauseen lopettamisen lisäksi esimerkiksi lyhenteissä, desimaalimerkkinä, internet-osoitteessa tai lauseen kesken jäämisen välimerkissä eli kolmessa pisteessä. Kun nämä erilaiset pisteen esiintymismahdollisuudet otetaan huomioon, saneistuksesta tuleekin yhtäkkiä huomattavasti haasteellisempi tehtävä jopa ihmiselle. (Proisl ja Uhrig 2016)

Saneiden määrittelyä vaikeuttavat homografit, yhdyssanat ja morfeemiset sanat (Webster ja Kit 1992). Esimerkiksi englannin kielessä osa yhdyssanoista kirjoitetaan erikseen kuten huonekaluliike “furniture shop” tai jäätelö “ice cream”. Jos saneistus jakaisi tällaisessa tapauksessa sanat erilleen, niin sanojen ja samalla koko lauseen merkitys muuttuisi (Webster ja Kit 1992; Proisl ja Uhrig 2016). Ihminen pystyy hahmottamaan helposti erikseen kirjoitetut yhdyssanat samaksi sanaksi, mutta tietokoneen toteuttaessa saneistusta tämän kaltaisten sanojen käsittely tulisi ottaa huomioon (Webster ja Kit 1992; Proisl ja Uhrig 2016). Suomen kielen saneistuksessa erikseen kirjoitetut yhdyssanat eivät ole ongelma, sillä yhdyssanat kir-

joitetaan lähtökohtaisesti yhteen.

Saneistuksessa on tärkeää ottaa huomioon konteksti, missä saneet esiintyvät. Jos käsitellään esimerkiksi englannin kielessä merkkijonoa “No.”, niin ilman kontekstia ei voida tietää minkälaisesta saneesta on kyse. Merkkijono “No.” voi tarkoittaa johonkin kysymykseen annettua lyhyttä kieltävää vastausta “Do you want coffee? – No.” tai se voi olla myös lyhenne sanalle numero “No. 6”. Kieltosanan tapauksessa kyseessä olisi siis kaksi sanetta eli kieltosana ja piste, kun taas jälkimmäisessä tapauksessa numero kuusi olisi vain yksi sane. Tämän kaltaista ongelmaa ei voida ratkaista ainoastaan saneistuksen avulla, vaan ongelmanratkaisuun tarvitaan avuksi myös muita NLP-menetelmiä. (Proisl ja Uhrig 2016)

Saneistuksen toteutus ei onnistu samalla tavalla kaikilla maailman kielillä. Esimerkiksi kiinan ja japanin kielissä teksti perustuu erilaisiin merkkeihin, jotka sisältävät useita erilaisia sanoja tai jopa kokonaisia lauseita. (Webster ja Kit 1992) Ilman selkeitä saneiden erottelumerkkejä on hankalaa määritellä saneistukselle sääntöjä, mistä sane alkaa ja mihin se loppuu (Webster ja Kit 1992; Haruechaiyasak ja Kongthon 2013). Tällaisten kielten tapauksessa saneistuksessa on erikseen luokiteltava tunnistettavat saneet esimerkiksi sanakirjapohjaisilla tai koneoppimiseen perustuvilla lähestymistavoilla (Webster ja Kit 1992; Haruechaiyasak ja Kongthon 2013). Ilman hyvää saneiden luokittelua merkkeihin perustuvissa kielissä olisi mahdotonta tunnistaa käsiteltäviä saneita, mikä vaikeuttaisi myös muita saneistuksen jälkeisiä NLP-menetelmiä (Webster ja Kit 1992).

Saneistukseen liittyvissä tutkimuksissa on perinteisesti käytetty apuna virallisia asiakirjoja ja uutistekstejä saneistusalgoritmin rakentamiseksi (Haruechaiyasak ja Kongthon 2013). Osassa tutkimuksissa on käsitelty myös sosiaalisen median ja lääketieteen tekstejä, joissa kirjoitustyyli ja termistö eroaa huomattavasti verrattuna perinteisiin asiateksteihin (Haruechaiyasak ja Kongthon 2013; Jiang ja Zhai 2007). Sosiaalisen median tekstit ovat yleensä lyhyitä ja voivat sisältää lyhennettyjä sekä homofonisesti muunneltuja sanoja (Haruechaiyasak ja Kongthon 2013). Lääketieteellisessä tekstissä puolestaan esiintyy usein erikoistermejä kuten geenien, proteiinien sekä kemikaalien nimiä. Nämä termit sisältävät usein erikoismerkkejä kuten numeroita, tavuviivoja, vinoviivoja ja hakasulkeita (Jiang ja Zhai 2007). Onkin selvää, että edellä mainittuja perinteisiä asiatekstejä varten rakennettu saneistusalgoritmi ei suoriudu yhtä hyvin sosiaalisen median tai lääketieteellisten tekstien saneistuksesta (Haruechaiyasak

ja Kongthon 2013; Jiang ja Zhai 2007).

Vaikka saneistusalgoritmin toteutuksessa tuleekin ottaa huomioon pieniä yksityiskohtia, suurin osa saneistuksen toiminnasta on hyvin yksinkertaista. Saneiden erottelu välilyöntien ja muiden yleisten välimerkkien perusteella johti Proislin ja Uhrigin 2016 toteuttamassa tutkimuksessa saneiden 96,73 %:n erottelutarkkuuteen. Tutkimuksessa todettiin, että 100 %:n erottelutarkkuuden saavuttaminen on todella haastavaa ja vaatii useiden pienten yksityiskohtien ottamisen huomioon. (Proisl ja Uhrig 2016) Tekstin täydellinen saneistus vaatii yleensä tiedon lauserakenteesta, mikä voidaan selvittää sanaluokittelumenetelmällä (Proisl ja Uhrig 2016; Haruechaiyasak ja Kongthon 2013).

2.3 Sanaluokittelu

Sanaluokittelu (engl. *part of speech tagging*) on tärkeä esikäsittelyvaihe useimmissa NLP-sovelluksissa (Biemann 2009; Nguyen ym. 2016; Petrov, Das ja McDonald 2012; Lv, Liu ja Dong 2010). Sanaluokittelussa määritellään tekstin sanoille tunnisteet, jotka kertovat, mitä sanaluokkaa kukin sana edustaa (Nguyen ym. 2016). Sanaluokkia ovat esimerkiksi substantiivit, verbit, pronominit ja adverbit (Lv, Liu ja Dong 2010). Kun tekstin sanojen sanaluokat on merkattu, niin tekstiä voidaan hyödyntää esimerkiksi puheentunnistuksessa, kielenkäänTämisessä tai tiedonhaussa (Nguyen ym. 2016; Lv, Liu ja Dong 2010).

Aina ei ole selvää, mitä sanaluokkaa käsiteltävä sana edustaa, sillä osalla sanoista sanaluokka määräytyvät lauseen kontekstin perusteella (Nguyen ym. 2016). Monet sanat voivat toimia sekä substantiivina että adjektiivina riippuen siitä, miten sanoja käytetään lauseessa. Esimerkiksi lauseessa “Tuolla juoksee suomalainen.” sana “suomalainen” toimii substantiivina, mutta lauseessa “Tuolla juoksee suomalainen mies.” sana “suomalainen” toimii adjektiivina ja sana “mies” substantiivina.

Sanaluokittelijan toteuttamiseksi tarvitaan tietokanta, joka sisältää kaikki mahdolliset sanojen sanaluokkatunnisteet. Tietokannan lisäksi sanaluokittelijasta tulee löytyä mekanismi, jolla tunnisteet merkitään tekstin sanoille. (Biemann 2009) Sanaluokittelijan tarkkuus riippuu siitä, millä tavoin se on koulutettu tunnistamaan sanaluokkia (Nguyen ym. 2016). Sanaluokittelualgoritmit on perinteisesti toteutettu joko sääntöpohjaisella tai tilastollisella lähesty-

mistavalla (Lv, Liu ja Dong 2010; Nguyen ym. 2016).

Sääntöpohjaisessa lähestymistavassa sanaluokat määräytyvät esimerkiksi sanojen leksikaalisen käytön ja sanoista löytyvien taivutusmuotojen perusteella (Lv, Liu ja Dong 2010). Sääntöpohjaisessa sanaluokittelussa sanaluokkien määrittäminen tapahtuu iteraatioissa, joissa pyritään tunnistamaan sanojen kontekstuaalinen merkitys käymällä lausetta läpi pienissä osissa. Iteraatioiden myötä sanojen sanaluokat saattavat muuttua riippuen lauseen muiden sanojen sanaluokista. (Nguyen ym. 2016)

Tilastoihin perustuvassa lähestymistavassa sanaluokat määräytyvät todennäköisyyksien perusteella (Lv, Liu ja Dong 2010; Petrov, Das ja McDonald 2012). Tilastoihin perustuva lähestymistapa käyttää yleensä sanaluokittelussa päätöspuuta. Päätöspuussa edetään erilaisten kysymysten avulla ja selvitetään, mihin sanaluokkaan kukin lauseen sana kuuluu. Päätöspuun avulla lauseen sanojen sanaluokat selviävät sanojen käyttötapausten perusteella. (Lv, Liu ja Dong 2010)

Päätöspuuta hyödyntävän sanaluokittelijan toteutuksessa käytetään yleensä koneoppimisen menetelmiä, sillä tällainen sanaluokittelija vaatii suuren määrän annotoitua dataa toimiakseen (Lv, Liu ja Dong 2010; Nguyen ym. 2016). Koneoppimisalgoritmi vaatii riittävästi ennalta opetettua aineistoa, jotta algoritmia pystytään soveltamaan myös uudelle tekstille (Biemann 2009). Koneoppimista hyödyntävä tilastoihin perustuva sanaluokittelu voi tuottaa hyviä tuloksia, sillä se pystynyt jopa 97,3% tarkkuuteen englanninkielisen tekstin sanaluokittelussa (Petrov, Das ja McDonald 2012).

2.4 Normalisointi

Normalisointi on yksi NLP-menetelmistä, jossa käsiteltävän tekstin sanat muutetaan niiden perusmuotoon (Toman, Tesar ja Jezek 2006; Korenius ym. 2004). Tekstin normalisoinnilla halutaan helpottaa muita NLP-menetelmiä siten, että kaikkien menetelmien ei tarvitsisi pystyä tunnistamaan sanojen erilaisia taivutusmuotoja vaan ainoastaan niiden perusmuodot (Toman, Tesar ja Jezek 2006). Esimerkiksi hakukoneet yleensä normalisoivat niille syötetyn tekstin, jotta avainsanojen määrittäminen hakutulokselle olisi helpompaa (Korenius ym. 2004).

Normalisointiin on kehitetty 1960-luvulta lähtien algoritmeja, joilla on haluttu nopeuttaa ja tarkentaa tietokoneen kykyä normalisoida tekstiä (Lovins 1968; Toman, Tesar ja Jezek 2006). Normalisoinnin käytetyimpiä algoritmeja ovat stemmaus (engl. *stemming*) ja lemmaus (engl. *lemmatisation*). Stemmauksen ja lemmauksen normalisointitekniikka eroaa huomattavasti toisistaan, vaikka molemmilla algoritmeilla onkin sama tehtävä. (Korenius ym. 2004; Porter 2006; Toman, Tesar ja Jezek 2006; Lovins 1968; Plisson, Lavrac ja Mladenic 2004)

2.4.1 Stemmaus

Stemmauksessa käsiteltävät sanat pyritään muuttamaan perusmuotoon poistamalla sanojen lopusta taivutusmuotoon viittaavat kirjaimet. Stemmaukselle on ominaista se, että poistettavat taivutusmuodot ovat ennalta määriteltyjä ja taivutusmuotojen poistaminen tapahtuu sääntöpohjaisesti. (Porter 2006) Algoritmi määritellään etsimään tiettyjä taivutusmuotoon liittyviä kirjainyhdistelmiä, joita sanoissa yleensä esiintyy. Esimerkiksi englannin kielessä sanapäätteet “-ed” ja “-ing” viittaavat sanan taivutusmuotoon, joten stemmausalgoritmi etsii ja poistaa nämä käsiteltävistä sanoista. (Lovins 1968) Stemmauksessa käsiteltävän sanan ei tarvitse olla sanakirjasta löytyvä sana, sillä algoritmi on kiinnostunut ainoastaan sanan taivutusosuudesta (Porter 2006).

Joskus pelkän taivutusmuodon poistaminen jättäisi sanan väärään muotoon, minkä vuoksi poistamisen sijaan stemmausalgoritmi muuntaa taivutusmuotoon viittaavia kirjaimia toisiin kirjaimiin (Porter 2006). Esimerkiksi englannin kielessä sanapääte “-ies” tulisi muuntaa kirjaimeksi “y”, jotta “-ies” päätteiset sanat kuten “babies” ja “memories” normalisoituisivat niiden oikeaan perusmuotoon (Lovins 1968). Stemmaus toteuttaa normalisointiprosessia yleensä iteraatioissa, joissa onnistuneen poisto-operaation jälkeen algoritmi aloittaa sanan tarkastelun uudestaan varmistaen, että sana normalisoitui oikeaan muotoon (Lovins 1968; Porter 2006).

Jokaisessa taivutustyypissä tulisi olla vain yksi mahdollinen lopputulos, joten stemmausalgoritmin toteuttajan täytyy määritellä, miten erilaiset taivutustyypit käsitellään algoritmin toimesta (Lovins 1968). Sääntöihin perustuvalla normalisoinnilla ei pystytä normalisoimaan kaikkia sanoja niiden perusmuotoon, koska jotkut säännöistä voivat aiheuttaa sanojen vää-

ränlaisen normalisoinnin. Esimerkiksi stemmaus poistaa sanojen lopusta taivutusmuotoon viittaavia sanapäätteitä “-ed” ja “-ing”, mikä johtaa englannin kielen sanat “computing” ja “computed” muotoon “comput” vaikkakin sanan oikea perusmuoto olisi “compute”. (Korenius ym. 2004)

Stemmauksen yksinkertaisen logiikan ansiosta se on helposti toteutettavissa ja samalla maailmanlaajuisesti käytetyin tekstin normalisointitekniikka (Korenius ym. 2004). Stemmaus on tehokas kielillä, joissa lauserakenne on suhteellisen yksinkertainen (Porter 2006). Stemmaus toimii erityisen hyvin esimerkiksi englannin, sloveenin, ranskan, modernin kreikan, arabian ja ruotsin kielillä (Lovins 1968; Korenius ym. 2004; Porter 2006).

2.4.2 Lemmaus

Lemmauksen normalisointitekniikka perustuu siihen, että se etsii sille määritellystä tietokannasta sanojen perusmuodot eli lemmat. Tietokannassa tulee olla merkattuna jokaisen taivutusmuodossa olevan sanan perusmuoto. (Korenius ym. 2004) Esimerkiksi tietokantaan tulisi olla merkattuna, että sanojen “koululaisen”, “koululaiselta” ja “koululaisella” lemma on “koululainen”. Kattavan sanatietokannan rakentamiseen menee aikaa ja se on haastavaa, minkä vuoksi lemmaus ei ole yhtä suosittu normalisointitekniikka stemmaukseen verrattuna (Korenius ym. 2004; Plisson, Lavrac ja Mladenic 2004).

Jotta sanojen löytäminen sanakirjasta olisi tehokkaampaa, lemmaus toteuttaa tekstille morfologisen analyysin, jonka tarkoituksena on selvittää tekstin lauserakenne ja samalla sanojen sanaluokat (Porter 2006). Sanat voidaan jakaa tietokantaan sanaluokittain, jolloin lemmaus pystyy etsimään sanoja nopeammin sanaluokan perusteella. Lemmaus hyödyntää morfologisen analyysin muodostamisessa luvussa 2.2 käsiteltyä saneistusta ja luvussa 2.3 käsiteltyä sanaluokittelua (Porter 2006). Lemmaus on hyvä esimerkki NLP-menetelmästä, joka hyödyntää muita NLP-menetelmiä omassa prosessissaan.

Lemmaus on stemmausta hitaampi prosessi, mutta lemmauksella pystytään normalisoimaan myös sellaisia sanoja, joita stemmauksella ei pystytä (Korenius ym. 2004; Toman, Tesar ja Jezek 2006; Lovins 1968). Morfologisen analyysin ansiosta lemmaus pystyy tunnistamaan sellaisia sanoja, jotka tarkoittavat eri asiaa erilaisessa kontekstissa (Korenius ym. 2004; Por-

ter 2006). Esimerkiksi sana “tee” voi tarkoittaa juomaa tai tehdä-verbin käskymuotoa. Lauserakenteen avulla lemmauksen tulisi pystyä tunnistaa, mihin sanaluokkaan kukin lauseen sana kuuluu ja täten pystyä normalisoimaan sanat niiden oikeaan perusmuotoon (Korenius ym. 2004; Porter 2006).

2.4.3 Suomen kielen normalisointi

Normalisointialgoritmien suoriutumiseen vaikuttaa käsiteltävän tekstin kieli ja kieleen liittyvät erityispiirteet (Korenius ym. 2004; Porter 2006). Suomen kielen sanataivutukset ovat huomattavasti monimutkaisempia verrattuna indoeurooppalaisiin kieliin (Korenius ym. 2004). Suomen kielessä sanojen taivutukset ovat sanan lopussa siinä missä indoeurooppalaisissa kielissä sanat taipuvat prepositioiden ja postpositioiden kautta. Esimerkiksi sanan “huone” taivutusmuoto “huoneessa” on englannin kielessä vastaavasti “in the room”. Suomen kielessä useat eri yksityiskohdat voivat vaikuttaa sanamuotoon kuten numerot, lauserakenne, aikamuoto, persoonamuoto ja muut morfologiset yksityiskohdat (Korenius ym. 2004). Suomen kielen monimutkaista taivutusta kuvastaa hyvin se, että sanoilla on paljon erilaisia sanamuotoja (Korenius ym. 2004).

Korenius ym. (2004) toteutti tutkimuksen, jossa testattiin neljää hierarkkista klusterimetodia ja todettiin, että lemmaus on stemmausta parempi suomenkielisen tekstin käsittelyyn. Tutkimuksessa käytettiin 5000 suomenkielistä dokumenttia, jotka oltiin kerätty suuresta uutisartikkelikokoelmasta. Tutkimuksen mukaan suurin ero lemmauksen ja stemmauksen välillä oli se, että stemmaus ei pystynyt käsittelemään oikein suomenkielisiä yhdyssanoja, joissa taivutus on sanan keskiosassa. (Korenius ym. 2004) Esimerkiksi yhdyssanassa “nuorelleparille” taivutus on sanan keskiosassa. Suomen kielessä on yhteensä noin 200 000 sanaa, joista kaksi kolmasosaa ovat yhdyssanoja, minkä vuoksi stemmaus ei ole lemmaukseen verrattuna kovin luotettava normalisointialgoritmi suomen kieleen (Korenius ym. 2004).

TurkuNLP[1] on kehittänyt NLP-työkalun nimeltä Turku-neural-parser-pipeline[2] (TNPP), joka tarjoaa lemmauksen lisäksi myös saneistus- ja sanaluokittelualgoritmien käyttöä. TNPP

[1]TurkuNLP. Lisätietoa: https://turkunlp.org
[2]Turku neural parser pipeline. Lisätietoa: https://turkunlp.org/Turku-neural-parser-pipeline

pystyy suomen kielen lisäksi lemmaamaan yli 50:tä erilaista kieltä 95 %:n onnistumistarkkuudella ja se on menestynyt hyvin muiden vastaavien työkalujen vertailuissa[2]. TurkuNLP:n tuottamaa työkalua tullaan hyödyntämään luvun 5 tutkimusosuudessa.

3 Luonnollisen kielen ymmärtäminen

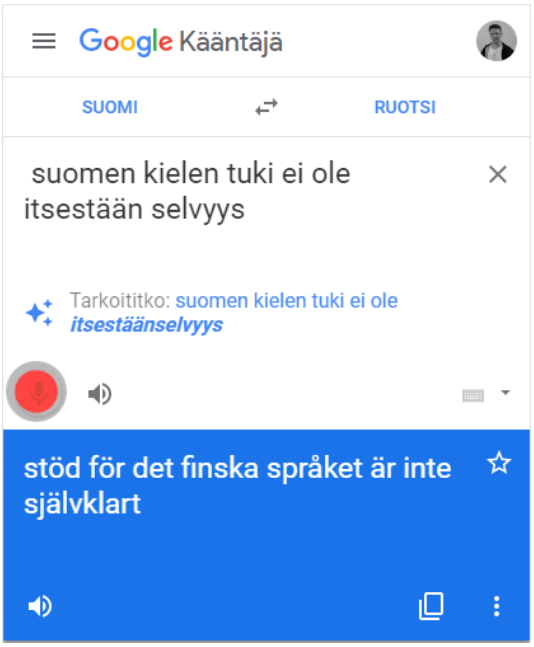
Luonnollista kieltä ymmärtävillä teknologioilla eli NLU-teknologioilla tarkoitetaan tässä tutkimuksessa palveluita ja ohjelmia, jotka ovat suunniteltu luonnollista kieltä ymmärtävien käyttöliittymien rakentamiseen. Tutkimuksen NLU-teknologioissa kielisyötteet käsitellään siten, että teknologia pystyy prosessoimaan sekä ymmärtämään syötteiden sisällön. Syötteiden käsittely on teknologioille kuitenkin haastavaa, sillä luonnollinen kieli sisältää semanttisen eli lauseen merkitystä koskevan tason (Navigli 2018).

Semanttisen tason muuntamisesta tietokoneelle ymmärrettävään muotoon tekee hankalaa se, että puhutussa kielessä esiintyy usein epäselvyyksiä sanojen sekä lauseiden merkityksissä. Epäselvyydet voivat johtua esimerkiksi kuuntelijoiden erilaisesta käsityksestä sanojen sisällöstä. Teknologioiden ongelmana onkin selvittää, miten syötteiden epäselvyydet tulisi ratkaista. Ihmiset kykenevät käsittelemään virheellistä tai hieman normaalista poikkeavaa kieltä myös vaivatta, kun taas tietokoneille nämä voivat tuottaa hankaluuksia. (Navigli 2018)

3.1 Käyttökohteet

Monet nykypäivän kuluttajista ovat saaneet ensikosketuksensa luonnollisen kielen teknologioihin käyttämällä käännöskoneita, joista yksi tunnetuimmista on Google Translate (ks. kuvio 2). Käännöskoneiden suosiota ei tarvitse enää perustella nykypäivän yhteiskunnassa, jossa vieraita kieliä kohdataan lähes päivittäin. Tämän seurauksena älypuhelimissa on yleistynyt myös erilaiset puheentunnistusta hyödyntävät sovellukset, jotka mahdollistavat puheen kääntämisen eri kielille jopa reaaliajassa.

Muita älypuhelimissa yleistyneitä puheentunnistuksella toimivia sovelluksia ovat virtuaaliavustajat, jotka monessa tapauksessa osaavat hoitaa myös käännöskoneiden työn. Virtuaaliavustajat osaavat muun muassa keskustella käyttäjänsä kanssa sekä toteuttaa erilaisia äänikomentoja kuten merkinnän asettamisen kalenteriin tai navigoinnin aloituksen. Virtuaaliavustajia ei ole tehty toimimaan ainoastaan älypuhelimissa, vaan myös kotiin sijoitettavissa älykaiuttimissa.



Kuvio 2. Google Translate osaa käsitellä suomenkielistä puhetta ja tekstissä esiintyviä virheitä.

Älykaiuttimet ovat puheohjauksella toimivia kaiuttimia, joiden avulla käyttäjät voivat esimerkiksi ohjata älykotinsa laitteita, soittaa musiikkia sekä hyödyntää muita virtuaaliavustajista tuttuja toimintoja (ks. kuvio 3). Eri laitevalmistajat ovat toteuttaneet myös älykaiuttimien kanssa yhteensopivia laitteita kuten valoja, siivousrobotteja, kameroita, lämpömittareita, kahvinkeittimiä sekä muita kodin laitteita. Älykaiuttimista on saatavilla monia eri versioita, jotka vaihtelevat sekä kooltaan että ulkonäöltään. Osasta älykaiuttimista on saatavilla myös näytöllisiä versioita, joissa näyttöä voidaan hyödyntää esimerkiksi kodin valvontakameroiden kanssa.

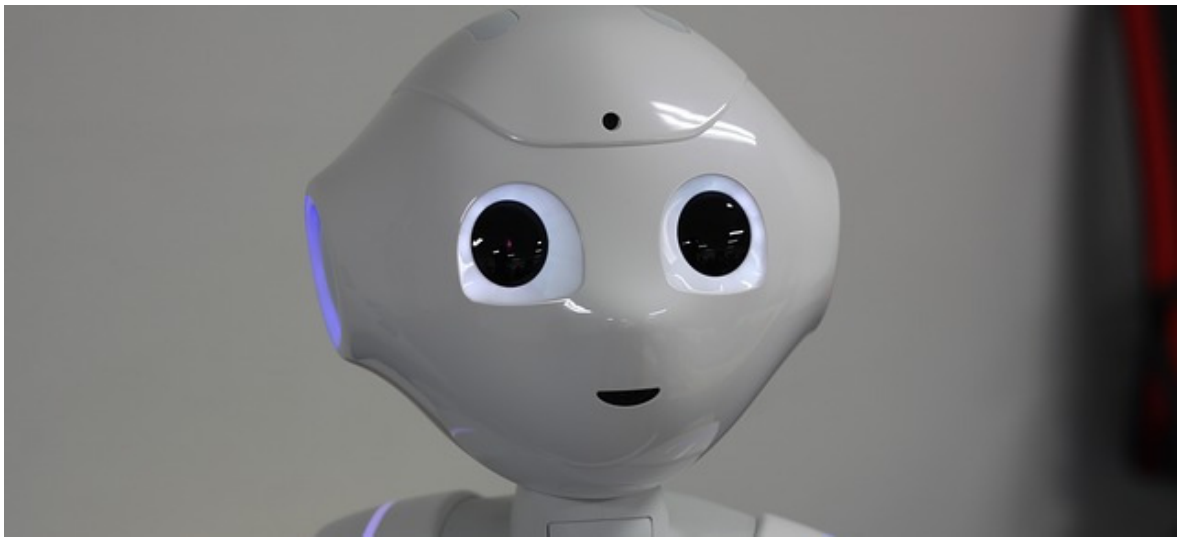
Kuvio 3. Älykaiuttimet ovat yleistyneet ihmisten kotona. Kuvassa Google Home Mini -älykaiutin. Kuva Eerik Lehtomäki.

Yritykset ovat aloittaneet luonnollisten kielten teknologioiden hyödyntämisen varsinkin asiakaspalvelun sekä data-analyysin puolella. Puhelinvastaajien, tukipuhelimien sekä verkkosivujen neuvonnan automatisointi teknologioiden avulla on yleistynyt erityisesti suurissa palvelualan yrityksissä, joissa asiakkailta tulee suuria määriä yhteydenottoja päivittäin. Käymällä läpi asiakkailta tulevat kysymykset ensin koneellisesti ja ehdottamalla näihin kysymysten sisällön perusteella automaattisesti ratkaisuja yritykset voivat karsia henkilökohtaisen asiakaspalvelun määrän tarvetta.

Asiakkaat tuottavat usein paljon arvokasta dataa, jota analysoimalla yritykset voivat esimerkiksi kohdistaa tuotteidensa sekä palveluidensa myyntiä paremmin tietyille asiakasryhmille. Prosessoimalla käyttäjien hakuhistoriaa tai keskusteluissa esiintyneitä sanoja yritykset pystyvät keräämään tietoa, mistä asiakkaat ovat mahdollisesti sillä hetkellä kiinnostuneita ja mukauttamaan tämän perusteella yrityksensä markkinointia asiakasryhmille sopivaksi. Samalla tavalla yritys voi seurata asiakkaidensa keskuudessa vallitsevia trendejä sekä kartoittaa uusia tarpeita tuotteille, joita yrityksellä ei ole vielä sillä hetkellä tarjolla.

Luonnollisen kielen teknologian hyödyntäminen yrityksissä ei rajoitu pelkästään yrityksen sisäisiin käyttötarkoituksiin, vaan yritykset ovat alkaneet käyttämään luonnollisen kielen tek-

nologioita myös tarjoamissaan palveluissa sekä tuotteissa. Yhä useampien puhelinsovellusten hakukentistä löytyy esimerkiksi mahdollisuus tehdä hakuja puhetta käyttäen ja hakutulokset mukautuvat sen perusteella, mitä muuta käyttäjät ovat etsineet samankaltaisilla hakuehdoilla. Tuotteiden käyttömukavuutta on parannettu myös lisäämällä mahdollisuus ääniohjauksen käyttöön, mikä antaa kuluttajille entistä vapaammat kädet kodinelektroniikan kanssa. Robotiikan kehitys on avannut uusia mahdollisuuksia ääniohjauksen hyödyttämiselle erityisesti terveydenhuollossa (ks. kuvio 4), jossa laitteiden helppo käytettävyys on ensiarvoisen tärkeää (Alho ym. 2018).



Kuvio 4. Seurarobotti Pepperiä on testattu jo muutamissa terveyskeskuksissa. Kuva (*Pepper Robot* 2020).

Tiedeyhteisöissä luonnollisen kielen tutkimuksen edistysaskeleet sekä tämän myötä syntyneet teknologiat ovat mahdollistaneet aikaisempaa kattavampien data-analyysien ja tieteellisten kokeiden toteuttamisen erilaisille luonnollisen kielen aineistoille. Data-analyysin avulla voidaan esimerkiksi pyrkiä löytämään yhteneväisyyksiä ihmisten kirjoittamista teksteistä sekä heidän poikkeavista toimintatavoistaan. Tulkitsemalla tekstin sisältöä on kehitetty esimerkiksi kokeita, joissa annetun tekstin sisällön pohjalta on pyritty tuottamaan halutunlaista kuvaa, ääntä tai muuta aineistoa. Kokeet tavoittelevat usein ihmisten kaltaisen ajattelun siirtämistä koneille sekä algoritmeille, jotta koneet pystyisivät toteuttamaan aikaisemmin vain ihmiselle mahdollisia tehtäviä. (Bisk, Yuret ja Marcu 2016)

3.2 Teknologioiden opetus

Eri NLU-teknologioiden opetuksessa oli havaittavissa paljon samankaltaisuuksia. Tässä tutkimuksessa käsiteltyjen teknologioiden opetus tapahtui syöttämällä teknologialle syötteitä, joita keskustelubotin halutiin kykenevän käsittelemään. Tämän jälkeen teknologialle kerrottiin, millä tavalla botin haluttiin reagoivan annettuihin syötteisiin ja mitä syötteen osia botin tuli erityisesti ottaa huomioon päätöksenteossa. Opetusvaiheen aikana keskusteluboteille määriteltiin syötteissä esiintyvät intentiot, eli mitä lauseen sisällöllä haluttiin tarkoittaa. Tämän lisäksi syötteestä voitiin valita entiteettejä, jotka olivat lauseen tarkoitusta tai kohdetta tarkentavia sanoja. Useissa teknologioissa oli tarjolla myös valmiita opetuspohjia, joissa yleisimmin käytetyt intentiot ja entiteetit olivat valmiiksi määriteltyjä.

Intentio kuvaa teknologioiden kohdalla käyttäjän pyrkimystä eli asiaa tai toimintaa, jota käyttäjä tavoittelee antamallaan syötteellään. Käyttäjä voi viitata samaan intentioon kuitenkin usealla eri tavalla.[3,4] Esimerkiksi seuraavien lauseiden alut “Haluaisin ostaa...”, “Voisinko saada...”, “Tilaisin...” ja “Saisinko...” voisivat kuulua saman intention alle nimeltä “ostaminen”. Tunnistamalla syötteistä intention teknologia kykenee reagoimaan syötteeseen käyttäjän toivomalla tavalla ja antamaan halutunlaisen vastauksen[4].

Entiteeteillä tarkoitetaan syötteestä löytyviä avainsanoja, jotka tarkentavat syötteen sisältämää intentiota. Entiteetit edustavat syötteessä olevaa tietoa, joka on merkittävä syötteen tarkoituksen kannalta. Jos intentio on syötteessä verbi, entiteetti edustaa usein substantiivia eli toiminnan kohdetta tai kontekstia[4]. Jos syötteen tarkoituksena on ostaminen, tarvitaan ensin tieto ostettavasta asiasta ennen kuin ohjelma voi toteuttaa ostotoiminnon[3]. Lauseen “Haluaisin ostaa...” entiteettejä voisivat olla esimerkiksi “leipäpussin”, “pehmiksen” tai “hampurilaisen”. NLU-teknologioissa koneoppimista käytetäänkin usein, sillä tämä on hyödyllinen koulutusvaiheen toteutuksessa. Teknologia voidaan opettaa koneoppimista hyödyntäessä lähes automaattisesti toimimaan sitä paremmin, mitä enemmän opetusdataa koulutuksessa syötetään.

Koneoppiminen voidaan jakaa opetuksen osalta kolmeen eri kategoriaan, joita ovat ohjaa-

[3]LUIS Docs. Lisätietoa: https://docs.microsoft.com/fi-fi/azure/cognitive-services/luis
[4]Watson Assistant Docs. Lisätietoa: https://cloud.ibm.com/docs/services/assistant

maton oppiminen, vahvistusoppiminen ja ohjattu oppiminen. Ohjatussa oppimisessa kone ei automaattisesti päättele syötteelle oikeaa ratkaisua, vaan kehittäjän pitää itse kertoa ongelman ratkaisu. Vahvistusoppimisessa kone pyrkii ensin tekemään mahdollisimman hyvän arvauksen ratkaisusta, jonka kehittäjä tarkistaa ja korjaa tarvittaessa. Ohjaamattomassa oppimisessa kone päättelee ratkaisun datasta löytyvien säännönmukaisuuksien pohjalta ja ratkaisun selvittäminen riippuu täysin ratkaisuja muodostavasta algoritmista. (Russell ja Norvig 2016; Lee, Shin ja Realff 2017)

Teknologiat voivat hyödyntää intentioiden sekä entiteettien tunnistuksessa koneoppimismalleja, jotka mahdollistavat teknologioiden koulutuksen esimerkkisyötteiden avulla. Koneoppimismallien ansiosta teknologiat oppivat toimimaan sitä paremmin mitä enemmän teknologioille annetaan esimerkkisyötteitä. Teknologiat pystyvät koulutuksen myötä havaitsemaan intentiot sekä entiteetit paremmin syötteistä ja tämän myötä antamaan tarkempia tuloksia. Teknologiat voivat koulutuksen myötä oppia tunnistamaan esimerkiksi käyttäjien kirjoitusvirheitä ja onnistua tämän avulla suoriutumaan myös virheellisistä syötteistä.

Opetusdata on usein teknologialle tavallisesti ratkaistavaksi annettavia syötteitä, joiden sisällön tarkoitus on valmiiksi selvitetty. Myöhemmässä vaiheessa keskustelubotin opetus voi tapahtua myös niin, että keskustelubotti pyrkii edellä saadun opetusdatan pohjalta arvaamaan uuden syötteen tarkoituksen, minkä käyttäjä varmistaa sekä korjaa tarvittaessa oikeaksi. Keskusteluboteilla pyritään selvittämään syötteiden sisällön tarkoitus sekä tarkoitusta tarkentavat osat, joiden perusteella keskustelubotti osaa toimia käyttäjän haluamalla tavalla. Tätä varten keskustelubotti on opetettu löytämään syötteestä toteutettavan toiminnan määrittelevät intentiot sekä entiteetit. (Russell ja Norvig 2016)

3.3 Keskustelubottien kehitys

NLU-teknologiat ovat tehneet keskustelubotin toteuttamisesta helppoa suurimmalle osalle puhutuimmista kielistä. Jos mikään NLU-teknologia ei kuitenkaan tue haluttua kieltä tai sisällä tarvittavia ominaisuuksia, niin oman keskustelubotin kehittäminen voi olla ainoa ratkaisu. Tähän päätökseen tuli muun muassa suomalainen Onerva yritys, joka kehitti oman

suomenkielisen keskustelubotin vanhustenhoitoa varten.[5]

Luonnollista kieltä ymmärtävän keskustelubotin kehittäminen on haastava, minkä takia jokainen käsiteltävä tapaus kannattaa rajata omaksi suppeaksi kokonaisuudeksi. Tällöin keskustelubotin tekoälyn tarvitsee ymmärtää keskustelun merkitys vain rajatusta aihepiiristä, mikä mahdollistaa paremman suorituskyvyn. Käyttötapauksia voidaan lisätä myös asteittain, jolloin näitä vastaava ongelma-avaruus kasvaa hallitusti.[5]

Keskustelubotin kehityksen alussa tulee kerätä mahdollisimman kattava esimerkkilauseista koostuva opetusaineisto botin koulutusta varten. Esimerkkilauseet ovat yleensä keskustelubotille syötettäviä lauseita, joiden merkitys on ennalta selvitetty. Pelkät esimerkkilauseet eivät tee keskustelubotista kuitenkaan älykästä, vaan opetuksen onnistumiseen tarvitaan myös matemaattisia malleja, joita toteutetaan usein koneoppimisella. Yksi koneoppimisessa käytettävistä matemaattisista malleista on neuroverkko, joka matkii toiminnaltaan aivojen hermoston toimintaa. Neuroverkon etu verrattuna moneen muuhun informaatiota käsittelevään malliin piilee siinä, että neuroverkon avulla koneet voivat ymmärtämään myös syötteitä, joita näille ei ole aiemmin opetettu. Tällöin syötteistä tulee kuitenkin löytyä joitain samankaltaisuuksia aikaisemmin opetettujen syötteiden kanssa, jotta neuroverkko pystyy päättelemään oikean vastauksen.[5]

Laadukasta koulutusaineistoa varten tarvitaan suuri määrä esimerkkisyötteitä, jotka ihminen on tulkinnut keskustelubotille sopivaksi. Yksi tapa helpottaa koulutusaineiston hankkimista on automatisoida työtä luomalla uutta aineistoa jo kerätystä aineistosta. Tämä tapahtuu antamalla koneen korvata olemassa olevien esimerkkien sanoja sekä lausahduksia näiden synonyymeillä tai antonyymeillä. Tällä tavoin voidaan myös tasapainottaa eri tyyppisten esimerkkien jakaumaa aineistossa luomalla harvinaisia vastaustyyppejä mahdollisimman paljon ja yleisimpiä ainoastaan vähän tai ei ollenkaan.[6]

Aluksi keskustelubotin neuroverkko ei ymmärrä mitään tälle syötettävistä lauseista, minkä takia keskustelubotin antamat vastaukset ovat aluksi täysin satunnaisia. Tällöin suurin osa

[5]Onerva. Lisätietoa: https://onervahoiva.fi

[6]Miten kouluttaa tekoäly ymmärtämään luonnollista kieltä? – osa 5. Lisätietoa: https://onervahoiva.fi/miten-kouluttaa-tekoaly-ymmartamaan-luonnollista-kielta-osa-5

vastauksista on myös todennäköisesti väärin, mitkä voidaan korjata jälkikäteen kertomalla botille oikeat vastaukset. Tämän seurauksena botin koulutusta hoitava algoritmi lähettää korjaussignaalin etenemään neuroverkossa taaksepäin eli merkityksestä kohti lauseen sisältöä. Korjaussignaalin edetessä keinotekoiset hermosolut eli "neuronit" korjaavat omaa käyttäytymistään sekä välittävät korjauksen eteenpäin, jolloin koko neuroverkon käyttäytyminen muuttuu hieman oikeampaan suuntaan. Kun "yritys-erehdys-korjausta" toistetaan tuhansia tai miljoonia kertoja, niin neuroverkko alkaa oppia tiettylaisia syötteitä ja tämän myötä myös antaa oikeita vastauksia.[7](Goldberg 2016)

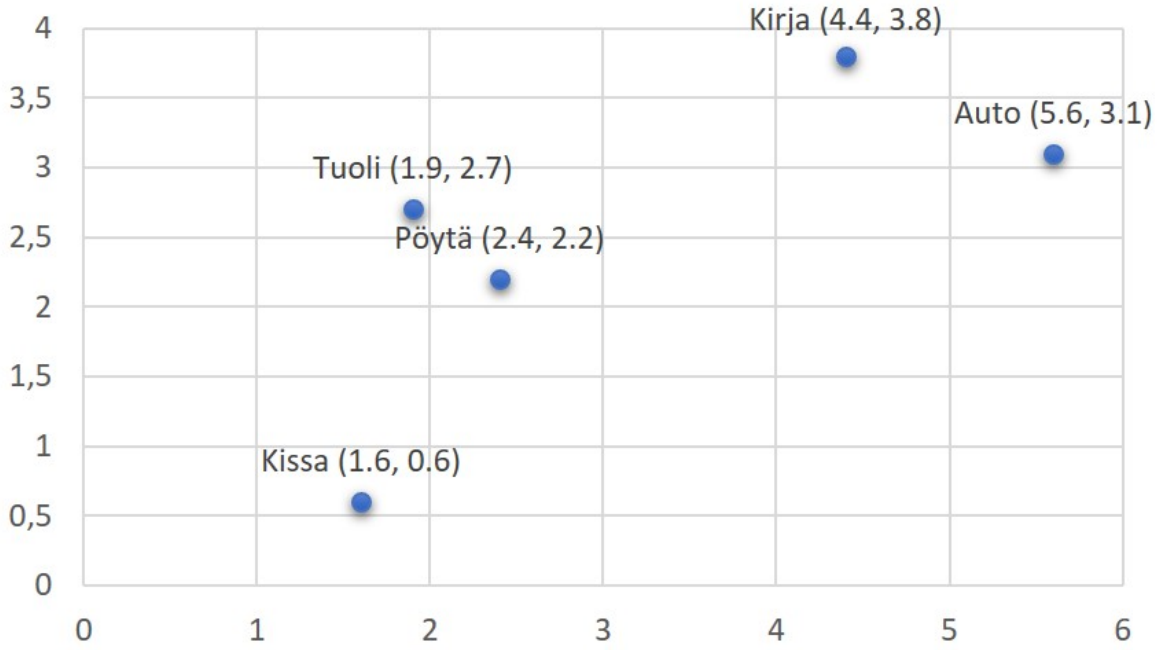
Jotta neuroverkko pystyy käsittelemään lauseita, niin lauseiden sisältö tulee olla ensin neuroverkolle sopivassa muodossa. Neuroverkolla voidaan käsitellä numeroina esitettyä sisältöä kuten vektoreita, matriiseja tai tensoreita. Lauseet voidaan muuttaa sarjaksi vektoreita tunnistamalla ensin lauseen sanat luvussa 2.2 esitetyn saneistuksen mukaisesti ja sitten muuntamalla jokainen sana omaksi vektoriksi. Yksinkertaisimmillaan sanoista voidaan luoda vektoreita one-hot encoding -menetelmällä, jossa vektorin muut elementit ovat nollia paitsi sanan paikkaa vastaava elementti on arvoltaan yksi. One-hot -vektoreiden pituus määräytyy käsiteltävän lauseen sanojen lukumäärän mukaan. Kuviossa 5 esitetty esimerkkilause "Lauseiden sisällön muuttaminen vektoreiksi on helppoa" sisältää kuusi sanaa, joten one-hot-vektorin pituus on tässä tapauksessa kuusi. (Maind ja Wankar 2014; Goldberg 2016)

Alkuperäinen	Perusmuoto	"One-hot" vektori
Lauseiden	lause	[1, 0, 0, 0, 0, 0]
sisällön	sisältö	[0, 1, 0, 0, 0, 0]
muuttaminen	muuttaa	[0, 0, 1, 0, 0, 0]
vektoreiksi	vektori	[0, 0, 0, 1, 0, 0]
on	on	[0, 0, 0, 0, 1, 0]
helppoa	helppo	[0, 0, 0, 0, 0, 1]

Kuvio 5. Lauseen muuttaminen vektoriksi one-hot encoding -menetelmällä.

[7]Miten kouluttaa tekoäly ymmärtämään luonnollista kieltä? Lisätietoa: https://onervahoiva.fi/miten-kouluttaa-tekoaly-ymmartamaan-luonnollista-kielta

One-hot -vektorin perusteella ei voida kertoa kuitenkaan mitään sanan luonteesta, minkä takia word embedding -malleja hyödynnetään usein one-hot vektorien kanssa. Word embedding -malleissa sanat muutetaan upotusvektoriksi (engl. *embedding*), jonka elementtien määrä on huomattavasti pienempi kuin one-hot -vektorin. Upotusvektorien kohdalla saman tyyliset sanat ovat arvoiltaan lähempänä toisiaan, mikä mahdollistaa myös sanojen luonteiden selvittämisen.[8] Esimerkiksi kissa, tuoli, pöytä, auto ja kirja sanojen tapauksessa voisi olla, että huonekaluihin viittaavat sanat pöytä sekä tuoli sijoittautuisivat lähimmäksi toisiaan kuten kuviossa 6.

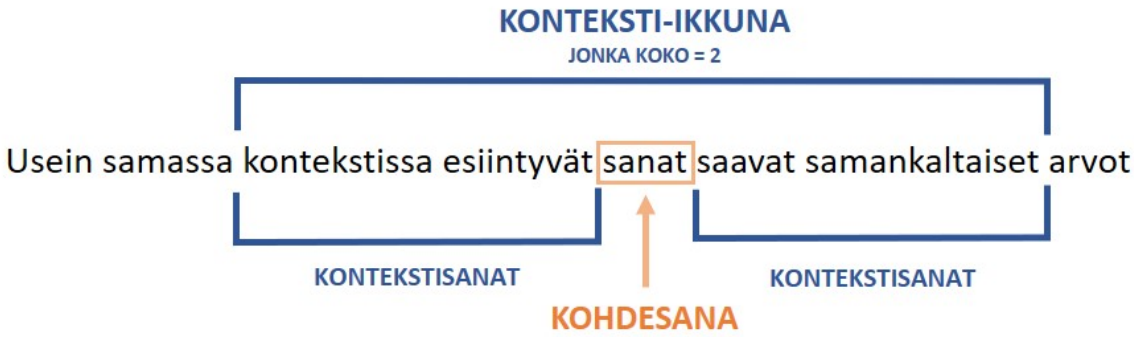


Kuvio 6. Upotusvektorit kaksiulotteisessa vektoriavaruudessa.

Yksi tapa muodostaa vektoriarvot on käyttää Skip-Gram -mallia, jossa samassa kontekstissa esiintyville sanoille annetaan samankaltaiset arvot. Samassa kontekstissa esiintyviä sanoja kohdesanasta enintään konteksti-ikkunan koon päässä olevat sanat. Kuviossa 7 esitetyssä Skip-Gram -mallin esimerkissä lauseeksi on valittu “Usein samassa kontekstissa esiintyvät sanat saavat samankaltaiset arvot”, konteksti-ikkunan kooksi kaksi ja kohdesanaksi “sa-

[8]Miten kouluttaa tekoäly ymmärtämään luonnollista kieltä? – osa 2. Lisätietoa: https://onervahoiva.fi/miten-kouluttaa-tekoaly-ymmartamaan-luonnollista-kielta-osa-2

nat”. Esimerkissä kohdesanan vasemmalle puolelle jäävät kontekstisanat “kontekstissa” sekä “esiintyvät” ja oikealle “saavat” sekä “samankaltaiset”. Lauseessa esiintyvät sanat on muutettu esimerkissä perusmuotoon, jolloin sanojen eri taivutusmuodot jätetään huomioimatta. Vektoriesitys muodostetaan kohdesanalle yksinkertaisen neuroverkon avulla niin, että kohdesana toimii neuroverkolle annettavana syötteenä ja konteksti-ikkunan sisälle jäävät sanat vasteena.[8](Mikolov ym. 2013)



Kuvio 7. Kontekstin muodostus Skip-Gram -mallissa.

Kun neuroverkolle syötetään koulutusaineistoa Skip-Gram -mallissa, niin samassa kontekstissa esiintyvien sanojen välille muodostuu neuroverkossa vahvempia kytköksiä. Mitä useammin sanat esiintyvät samassa kontekstissa, niin sitä vahvempi kytkös sanojen välille muodostuu. Kun koulutuksen jälkeen neuroverkolle syötetään jokin koulutusaineistossa löytyvistä sanoista, niin sanan upotusvektori aktivoituu ja neuroverkko palauttaa sanan kontekstin. Skip-Gramin kaltaisissa malleissa koulutus voidaan suorittaa ohjaamatonta oppimista hyödyntäen, minkä ansiosta koulutusaineisto voidaan syöttää suoraan koneelle ja kone pystyy itsenäisesti oppimaan aineiston. Opetuksessa voidaan käyttää myös suurempia datamassoja ja luoda kattavampia upotuksia pienemmällä työmäärällä ohjaamattomaan oppimiseen verrattuna.[8]

4 Teknologioiden toiminta suomen kielellä

Ensimmäisen tutkimusvaiheen tarkoituksena oli selvittää, minkälaisia ominaisuuksia NLU-teknologiat sisältävät ja kuinka suomen kielellä toimivan keskustelubotin toteutus teknologioilla tapahtuu. Tutkimukseen valittiin teknologiat, jotka olivat tutkimuksen aikana yleisesti käytössä. Tutkimusvaiheen aikana jokaisella valitulla teknologialla toteutettiin suomen kielellä toimiva keskustelubotti, jota koulutettiin ja testattiin suomenkielisellä tuotenimiä sisältävällä testiaineistolla. Tutkimuksessa hyödynnettiin teknologioiden dokumentaatioita hinnoittelun, rajapintojen, suomen kielen tuen sekä muiden tuettujen kielten määrän selvityksessä.

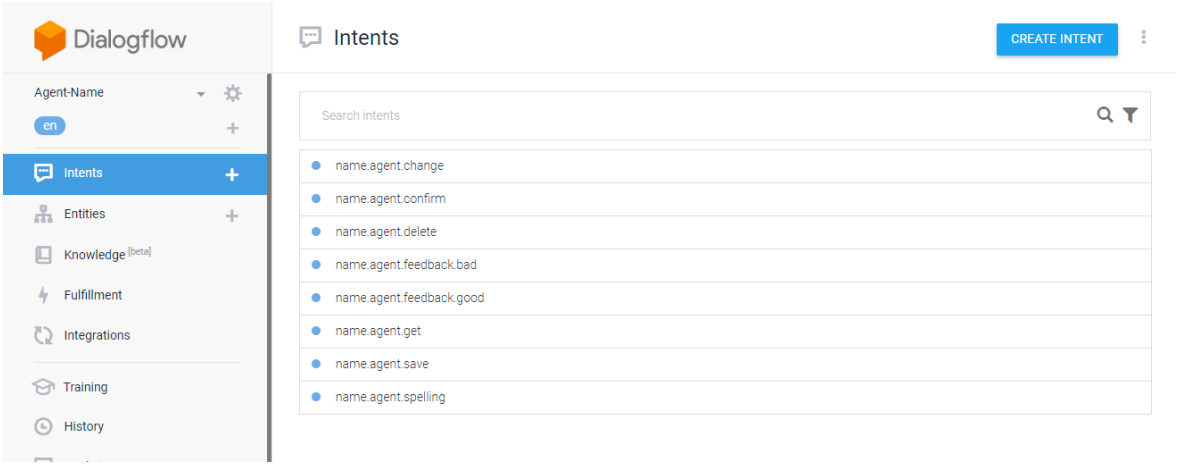
Keskustelubottien toteutuksella selvitettiin sekä teknologioiden käyttöliittymien toimintaa että keskustelubottien toteutuksen läpivientiä. Keskustelubotit koulutettiin kuvitteellisten tuotteiden myyntiä varten ja niiden toimintaa testattiin tuotenimiä sisältävillä tekstisyötteillä. Useimmissa tapauksissa keskustelubottien kieleksi jouduttiin valitsemaan englanti suomen kielen sijaan, koska suomen kieltä ei löytynyt useimpien keskustelubottien kielivalikoimasta. Keskustelubottien koulutus ja testaus tapahtui kuitenkin aina samalla suomenkielisellä aineistolla valitusta kielestä riippumatta.

4.1 Dialogflow

Dialogflow on Googlen omistama NLU-teknologia, joka julkaistiin ensimmäisen kerran vuonna 2011 nimellä Speaktoi[9]. Dialogflow'n käyttöönotto tuntui kokeilun aikana vaivattomalta tämän käyttöliittymän (ks. kuvio 8) sekä kattavan dokumentaation puolesta[10]. Dialogflow'ssa käyttäjän ei tarvinnut osata ohjelmoida lainkaan, sillä teknologia tarjosi toiminnon, jolla helpotettiin keskustelubotin integraatiota. Toiminnon ajatus oli, että käyttäjät voivat yhdistää keskustelubottinsa toimimaan helposti esimerkiksi useimmissa sosiaalisen median alustoissa yhdellä napinpainalluksella. Dialogflow tarjosi keskustelubotin luonnin yhteydessä myös

[9]Build your own Siri: Api.ai offers voice integration for all. Lisätietoa: https://thenextweb.com/dd/2014/09/16/build-your-own-siri-api-ai-offers-voice-integration-for-all
[10]Dialogflow. Lisätietoa: https://cloud.google.com/dialogflow

tietojen hakua helpottavan ominaisuuden, jota voitiin hyödyntää tiedon hakemiseen kolmannen osapuolen palveluista.[10]



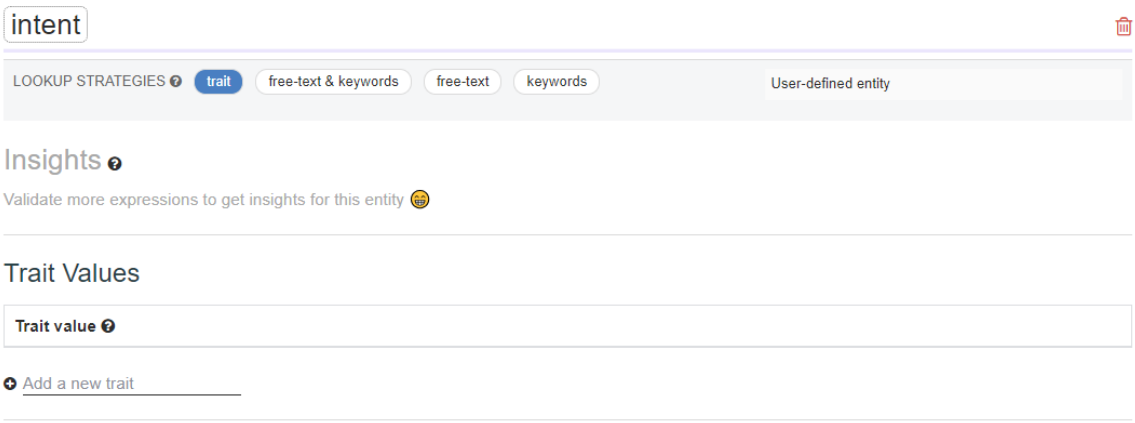
Kuvio 8. Dialogflown käyttöliittymä 09/2018.

Dialogflow'ssa keskustelubotin koulutus ja toteutus keskittyi "agentti" nimisiin peruskomponentteihin, jotka muuttivat syötetyt vastaukset herätteiksi sekä hallitsivat käyttäjän ja keskustelubotin välisen vuorovaikutuksen etenemistä[10]. Dialogflow'ssa käyttäjä pystyi hyödyntämään joko valmiiksi luotuja agentteja tai luomaan näitä itse alusta loppuun. Agenttien luonnin yhteydessä oli mahdollista syöttää esimerkkilauseita, joiden mukaan agentit aktivoituivat intentioiden tapaan. Koulutus voitiin toteuttaa komentorivipohjaisesti visuaalisen käyttöliittymän sijaan, mikä vaatii kuitenkin hieman syvempää perehtymistä teknologian toimintaan.

Helppokäyttöisyyden lisäksi Dialogflow'n hyviin puoliin lukeutui kattava alustatuki ja halpa hinnoittelu. Tutkimuksen aikana Dialogflow'sta oli saatavana kaksi eri versiota, joista toinen oli ilmaisversio ja toinen käytön mukaan laskutettava. Dialogflow'n ilmaisversiossa prosessikutsujen määrää oli rajoitettu, mikä ei vaikuttanut olevan kuitenkaan haitaksi useimmille ilmaisversiota käyttäville kehittäjille tai pienyrityksille. Isommille sovellusratkaisuille suunnatussa maksullisessa "Pay as you go" -versiossa ei rajoitettu suoritettavien prosessikutsujen määrää kuten ilmaisversiossa, mutta jokaisesta suoritettavasta prosessikutsusta perittiin pieni maksu. Dialogflow kertoi tukevansa 18:a eri kieltä ja kykenevänsä käsittelemään näistä 15:ttä samanaikaisesti yhdellä agentilla.[10]

4.2 Wit.ai

Wit.ai on vuonna 2013 toimintansa aloittanut NLU-teknologia, jonka Facebook osti omistukseensa vuonna 2015[11]. Wit.ai:ssa ensimmäisen projektin luonti oli tehty hyvin helpoksi visuaalisella käyttöliittymällä sekä selkeällä animoidulla dokumentaatiolla. Wit.ai sisälsi uusien syötteiden opetusta varten tekstikentän, jossa teknologia merkkasi tekstisyötteisiin jo opetettujen intentioiden ja entiteettien sanat (ks. kuvio 9). Keskustelubotin luonnissa oli mahdollista hyödyntää myös ennalta opetettuja malleja, joilla pystyi tunnistamaan testisyötteistä automaattisesti numeroita, linkkejä sekä sähköpostiosoitteita. Wit.ai:n käyttämä rajapinta perustui HTTP-protokollaan, jonka ansiosta teknologia oli helppo yhdistää toimimaan omassa ohjelmassa.[12]



Kuvio 9. Wit.ai:n käyttöliittymä 10/2018.

Wit.ai:ssa keskustelubotin opetus tapahtui antamalle käyttöliittymälle koulutettavia syötteitä, joista Wit.ai pyrki automaattisesti havaitsemaan entiteettejä. Jos Wit.ai ei tunnistanut haluttua entiteettiä syötteestä, niin entiteetin pystyi myös itse valitsemaan sekä yhdistämään haluamaansa intentioon. Koulutusvaiheessa oli mahdollista luoda myös oma etsintästrategia entiteeteille syöttämällä entiteettejä vastaavia hakusanoja intentioiden luonnin yhteydessä. Valmiita etsintästrategioita entiteeteille oli neljä eri vaihtoehtoa, joita olivat piirre, vapaateksti, avainsanat ja vapaateksti & avainsanat. Piirre-etsintästrategiassa lauseessa olevien sa-

[11]Wit.ai is joining Facebook. Lisätietoa: https://medium.com/wit-ai/wit-ai-is-joining-facebook-deff3745fcf5
[12]Wit.ai Docs. Lisätietoa: https://wit.ai/docs

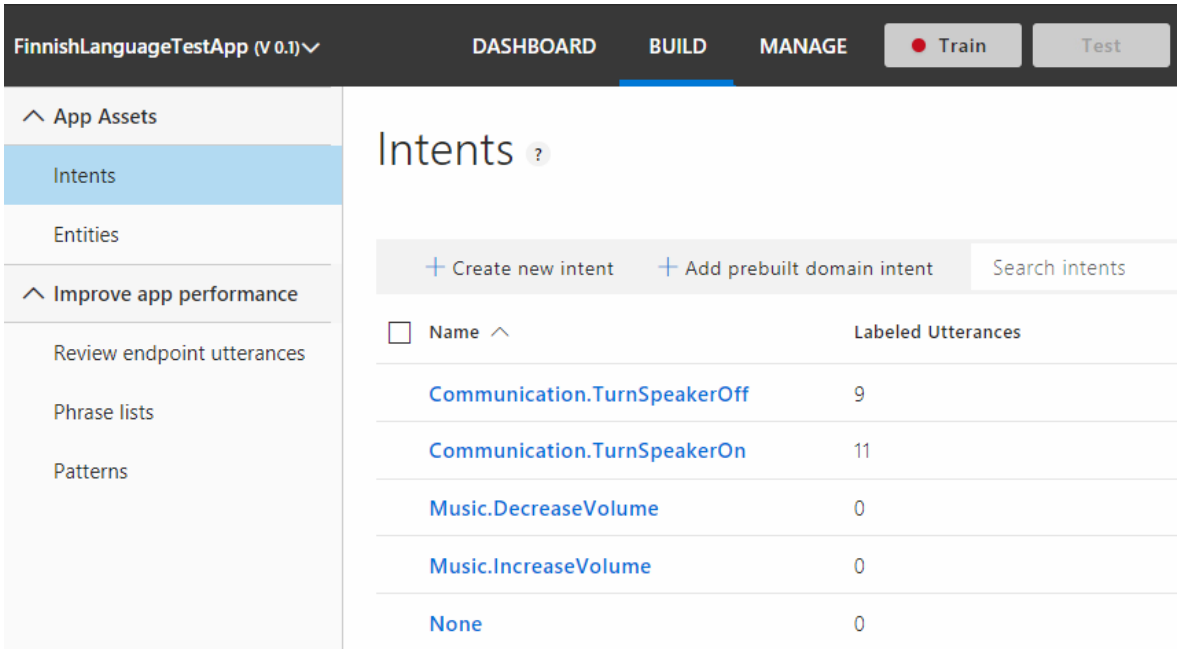
nojen välillä ei ollut selvää yhteyttä, vaan lausetta tarkasteltiin kokonaisuutena. Vapaatekstistrategiassa lauseesta ei pyritty löytämään pelkästään tiettyjä sanoja, vaan sanojen tuli olla lauseessa tietyssä järjestyksessä. Avainsanastrategiassa lauseesta tuli löytyä riittävä määrä syötettyjä avainsanoja ja vapaateksti & avainsanat -strategia oli yhdistelmä kahta edellä mainittua etsintästrategiaa.[12]

Wit.ai:n käyttö oli ilmaista ei-kaupallisessa ja kaupallisissa käyttötarkoituksissa, minkä takia teknologia oli hyvä vaihtoehto sekä yksityishenkilöiden että yritysten käyttöön. Palveluun lähetettävien prosessikutsujen määrää ei oltu rajoitettu, mutta teknologian ylläpitäjä suositteli kertomaan, jos aikomuksena oli toteuttaa suuria määriä prosessikutsuja kerralla[13]. Wit.ai:ssa oli toteutettu Node.js-, Python- ja Ruby-ohjelmointikielille viralliset rajapinnat, jotka helpottivat teknologian käyttöönottoa[14]. Wit.ai-käyttäjät olivat luoneet ajan saatossa myös omia epävirallisia rajapintoja muille ohjelmointikielille, mikä antoi positiivisen kuvan teknologian ympärille muodostuneesta kehittäjäyhteisöstä. Wit.ai kertoi tukevansa suomen kieltä ja kaiken kaikkiaan 73:a eri kieltä. Vaikka Wit.ai lupasi tuen useille eri kielille, niin monen kielen kohdalla tukea ei ollut kehitetty kovin pitkälle.[12]

4.3 LUIS

LUIS (Language Understanding Intelligent Service) on Microsoftin vuonna 2015 julkaisema koneoppimista hyödyntävä NLU-teknologia, joka toimi osana Microsoftin Azure-pilvipalvelua. LUIS julkaistiin alun perin osana Oxford-projektia, jonka yhteydessä Microsoft julkaisi myös muita koneoppimista hyödyntäviä Azure-teknologioita[3]. Keskustelubotin luonti alkoi LUIS:n kohdalla määrittelemällä botin nimi sekä käytettävä kieli. Tämän jälkeen edettiin suoraan intentioiden sekä entiteettien määrittelyyn, jossa oli mahdollisuus hyödyntää valmiiksi koottuja intentio- ja entiteettikokoelmia. Teknologian päänäkymä ei sisältänyt paljon ohjeistusta, mutta sovelluksen dokumentaatiosta löytyi kaikki tarvittava tieto teknologian käyttöön. Käyttöliittymä LUIS:n kohdalla oli varsin pelkistetty ja välillä olisi toivonutkin, että ohjeistusta sovelluksen käyttöön olisi tullut käyttöliittymän puolelta (ks. kuvio 10).

[13]Wit.ai Terms. Lisätietoa: https://wit.ai/terms
[14]Wit.ai GitHub. Lisätietoa: https://github.com/wit-ai



Kuvio 10. LUIS käyttöliittymä 10/2018.

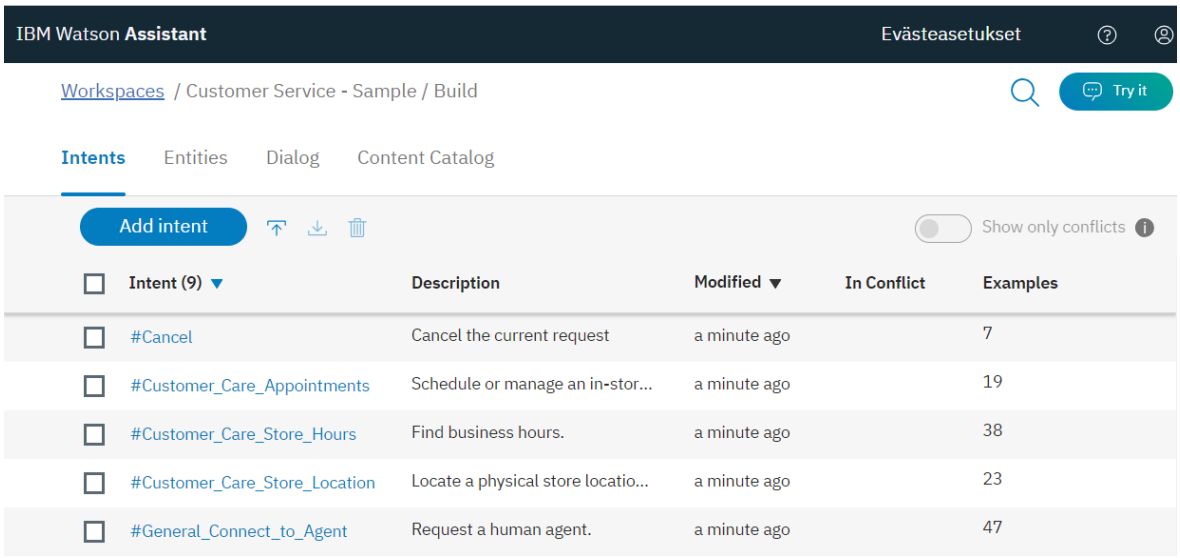
Intentioiden, entiteettien ja esimerkkilauseiden syötön jälkeen painamalla opetus-painiketta LUIS loi koulutetun mallin syötettyjen tietojen pohjalta. LUIS tarjosi mallin integrointiin "Bot Builder SDK" -paketin, joka mahdollisti teknologian helpomman integroinnin Node.js- ja .NET-alustoille. LUIS-teknologiasta oli tarjolla sekä ilmainen että maksullinen versio. Ilmaisversiossa suoritettavien API-pyyntöjen määrä sekunnissa oli rajoitettu viiteen ja viiteenkymmeneen maksullisessa. Ilmaisversiossa oli mahdollistaa toteuttaa ainoastaan tekstisisältöisiä prosessikutsuja, kun taas maksullisessa versiossa myös äänisisältöiset kutsut olivat mahdollisia. Ilmaisversion toiminta oli rajattu vain yhteen maantieteelliseen alueeseen, kun taas maksullinen versio ei sisältänyt maankohtaisia rajoitteita. Hinnoittelu maksullisessa versiossa tapahtui toteutettujen prosessipyyntöjen määrän ja sisällön perusteella[3].

LUIS:ssa keskustelubotti voitiin asettaa toimimaan yhdellä valitulla kielellä, jota ei ollut mahdollista muuttaa kesken koulutusprosessin. Tutkimuksen aikana LUIS listasi tukevansa 12:ta eri kieltä vaihtelevalla tasolla, mutta suomen kieltä ei tältä listalta löytynyt. Suomen kielen sijaan LUIS vaikutti huomioineen varsin hyvin maailmalla eniten puhutuimmat kielet kuten englanti ja kiina[3].

4.4 Watson Assistant

Watson Assistant on IBM:n kehittämä teknologia, joka luotiin ymmärtämään luonnollista kieltä sisältäviä keskusteluita sekä syötteitä. Watson Assistant julkaistiin vuonna 2016 osana IBM Watson -pilvipalvelua, joka sisälsi myös muita koneoppimista hyödyntäviä teknologioita[15].

Watson Assistantin visuaalisesta käyttöliittymästä löytyi tyypilliset ominaisuudet kuten intentioiden ja entiteettien luonti (ks. kuvio 11). Assistant tarjosi hyvän ohjeistuksen keskustelubotin toteutukseen sekä mahdollisuuden hyödyntää valmiiksi luotuja intentio- ja entiteettikokoelmia botin koulutuksessa. Teknologia sisälsi myös hyvin toimivan dialogityökalun, jonka avulla keskustelubotille voitiin rakentaa useita erilaisia keskustelupolkuja.



Kuvio 11. Watson Assistantin käyttöliittymä 10/2018.

Watson Assistantin koulutus tapahtui luomalla kykyjä (engl. *skills*), jotka sisälsivät koulutuksen tiedot sekä logiikan. Käytännössä tämä tarkoitti intentioiden, entiteettien sekä dialogien luomista kykyihin, joiden pohjalta Watson Assistant loi opetetun mallin keskustelubottia varten. Aina kun kykyihin lisättiin uusia komponentteja tai komponenttien sisältöön tehtiin muutoksia, niin koulutusprosessi käynnistyi uudelleen luoden päivitetyn mallin botis-

[15]Watson Conversation is moving from Experimental to General Availability. Lisätietoa: https://www.ibm.com/blogs/bluemix/2016/07/watson-conversation-general-availability

ta.[16]

Watson Assistantista oli tarjolla ilmais- ja normaaliversio sekä edistynyt versio. Ilmaisversiossa prosessikutsujen määrä oli rajoitettu 10 000 kutsuun kuukaudessa, kun taas maksullisissa normaalin ja edistyneen tason versioissa toteutettavien prosessikutsujen määrää ei ollut rajoitettu. Edistyneen tason versio poikkesi normaaliversiosta tarjoamalla hieman paremman tietoturvallisuuden ja datasiirron.[17] Kaikissa versioissa Assistant tuki useampia eri ohjelmistoalustoja ja ohjelmointikieliä oman rajapintansa sekä omien virallisten SDK-työkalujen kautta. Watson Assistant kertoi tukevansa 13:a eri kieltä, mutta suomen kieltä ei löytynyt tästä joukosta[16]. Yksinkertaisen keskustelubotin toteutus onnistui kuitenkin myös suomen kielellä, vaikka teknologian opetuskieleksi oli valittuna englanti.

4.5 Amazon Lex

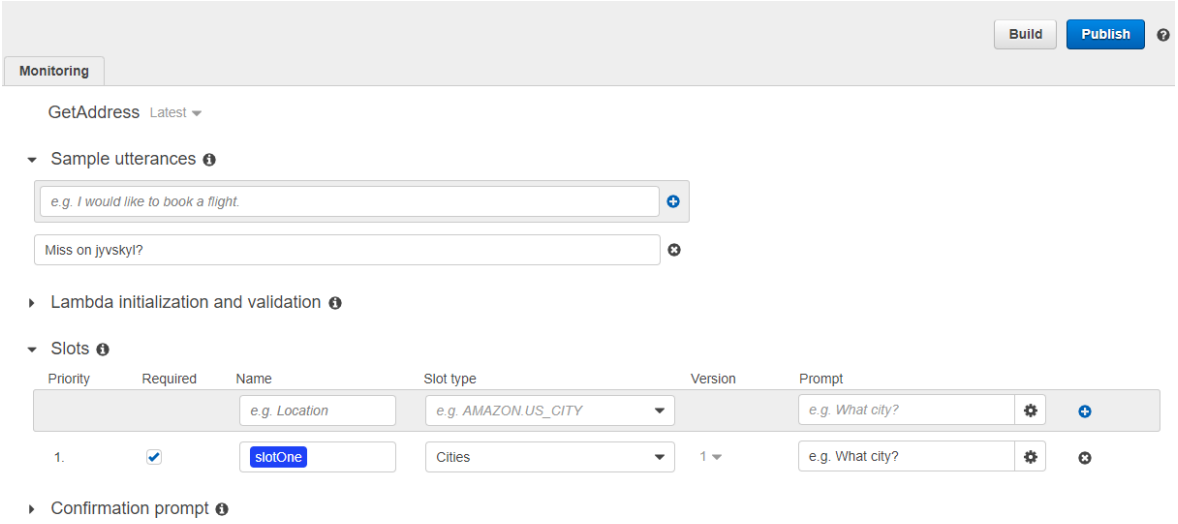
Amazon Lex on vuonna 2017 julkaistu NLU-teknologia, jolla oli mahdollista rakentaa puhetta sekä tekstiä ymmärtäviä keskustelubotteja. Testauksen aikana Amazon Lex käytti samaa keskustelumoottoria, mitä Amazon hyödynsi tarjoamassaan Amazon Alexa -laitteessa. Teknologiasta löytyi keskustelubotin luonnin lisäksi automaattinen puheentunnistus, jonka avulla englanninkielinen puhe voitiin helposti muuttaa tekstimuotoon[18]. Amazon Lex oli mahdollista integroida toimimaan monien muiden Amazonin tarjoamien teknologioiden kanssa AWS-alustan kautta. AWS-alusta oli samankaltainen Microsoftin tarjoaman Azure-palvelualustan[19] kanssa.

Amazon Lexin käyttöönotto tapahtui luomalla keskustelubotti joko tyhjän tai valmiin mallin pohjalle. Botin luonnissa tuli aluksi määritellä tämän asetukset kuten nimi, yhteyden maksimiaika sekä säilöttävän tiedon sisältö. Tämän jälkeen botin koulutus voitiin aloittaa luomalla intentioita sekä määrittelemällä näistä seuraavat toiminnot[18].

Amazon Lexin kohdalla keskustelubottien rakennus onnistui ilman aikaisempaa kokemusta

[16]Watson Assistant Docs. Lisätietoa: https://www.ibm.com/watson/developercloud/assistant/api/v1
[17]Watson Assistant Pricing. Lisätietoa: https://www.ibm.com/cloud/watson-assistant/pricing
[18]Amazon Lex Docs. Lisätietoa: https://docs.aws.amazon.com/lex/latest/dg/what-is.html
[19]Microsoft Azure. Lisätietoa: https://azure.microsoft.com

bottien luonnista. Botin koulutus tapahtui syöttämällä botille käyttäjiltä todennäköisesti saatavia esimerkkilauseita, joiden perusteella teknologia muodosti keskustelubotin opetusmallin (ks. kuvio 12). Koulutetun opetusmallin avulla botti kykeni vastaamaan tälle syötettyihin lauseisiin joko äänen, tekstin tai muiden erikseen määriteltyjen toimintojen kautta.



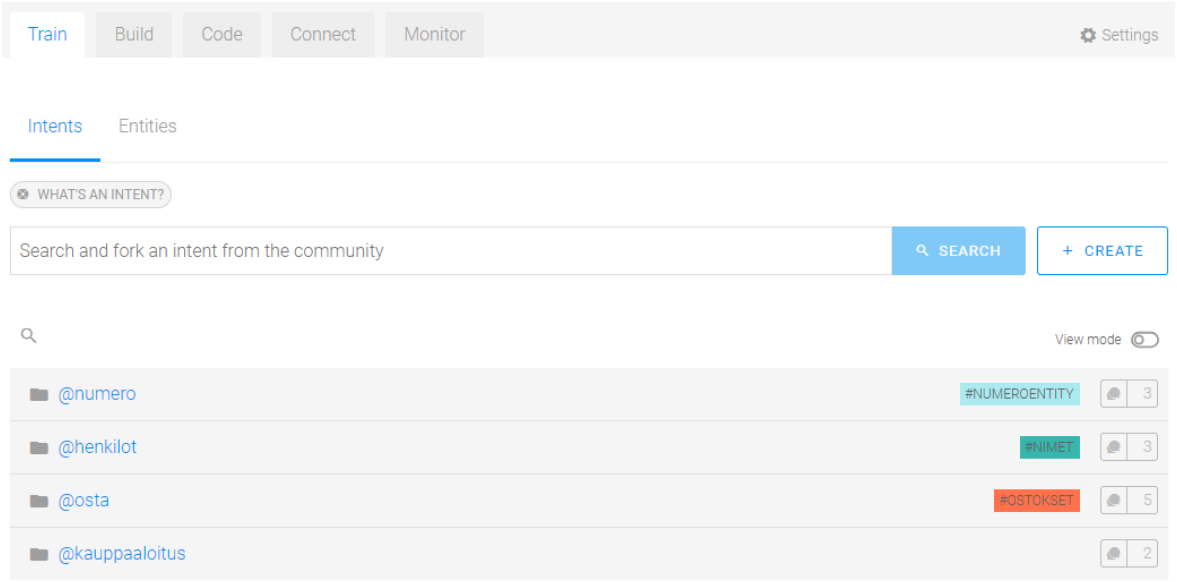
Kuvio 12. Amazon Lex käyttöliittymä 1/2019.

Amazon Lex tarjosi yli kymmenelle eri ohjelmointikielelle omaa SDK-työkalua keskustelubotin integrointia varten. Lexiä voitiin käyttää täysin ilmaiseksi ensimmäisen vuoden ajan, mutta tällöin keskustelubotilla voitiin toteuttaa enintään 10 000 tekstipyyntöä sekä 5000 puhekyselyä kuukaudessa. Maksullisessa versiossa veloitus tapahtui kuukauden aikana käsiteltyjen teksti- ja äänipyyntöjen määrän mukaan.[18]

Amazon Lexin kielituki oli tutkimuksessa käsitellyistä teknologioista suppein, koska Amazon Lex ilmoitti tukevansa ainoastaan englannin kieltä[18]. Amazon Lex olikin muista teknologioista poiketen keskittynyt ainoastaan englannin kielen tuen kehittämiseen useamman kielen tuen sijaan.

4.6 Recast.ai

Recast.ai on vuonna 2015 julkaistu teknologia, jonka avulla oli mahdollista luoda keskustelubotteja useammalla eri kielellä[20]. Keskustelubotin pystyi toteuttamaan Recast.ai:ssa myös monikielisenä, jolloin samalla keskustelubotilla voitiin käsitellä useamman eri kielen syötteitä[20]. Recast.ai sisälsi visuaalisen käyttöliittymän (ks. kuvio 13), jonka kautta botteja oli mahdollista luoda, kouluttaa, käyttää sekä seurata.



Kuvio 13. Recast.ai käyttöliittymä 1/2019.

Keskustelubotin luonti alkoi Recast.ai:ssa valitsemalla valmiiksi koulutettuja kykyjä, joita botin haluttiin osaavan heti alusta alkaen. Seuraavaksi keskustelubotille annettiin nimi ja toimintakieli, joka voitiin myöhemmin myös muuttaa monikieliseksi. Keskustelubotin luonti jatkui määrittelemällä käyttäjäryhmät ja näiden käyttöoikeudet. Keskustelubotin käyttö voitiin rajata esimerkiksi vain itselle, kehittämisessä mukana olleille henkilöille tai kaikille avoimeksi Seuraavaksi voitiin aloittaa itse botin koulutus visuaalisella käyttöliittymällä, jossa intentioiden, entiteettien ja kykyjen määrittely tapahtui.[20]

Keskustelubotin koulutus voitiin aloittaa luomalla uusia intentioita tai valitsemalla ne valmiiksi luotujen intentioiden joukosta. Intentioiden luonnissa botille syötettiin esimerkkisa-

[20]Recast. Lisätietoa: https://recast.ai

noja tai -lauseita, joilla intentioiden haluttiin aktivoituvan. Kuten intentiot, niin myös entiteetit voitiin valita joko valmiiksi määriteltyjen entiteettikokoelmien joukosta tai määritellä ne itse. Keskustelubotille voitiin ohjelmoida myös erilaisia toimintoja, joiden tuli tapahtua, kun intentio havaittiin. Toiminnot olivat esimerkiksi tekstiä, puhetta tai ohjelmalle lähetettäviä viestejä. Keskustelubotin toimivuutta voitiin helposti testata teknologian omassa käyttöliittymässä ennen botin varsinaista käyttöönottoa.

Recast.ai oli mahdollista yhdistää toimimaan useimpiin suosituimpiin sosiaalisen median keskustelusovelluksiin kuten Facebook Messengeriin ja Twitteriin. Teknologia tarjosi myös valmiit SDK-rajapinnat Ruby-, Javascript-, Python-, PHP- ja Java-ohjelmointikielille. Teknologian käyttö oli täysin ilmaista, jos sovelluksesta lähetettävien prosessikutsujen määrä ei ylittänyt kolmea kutsua sekunnissa. Aktiivisemman käytön hinnoittelusta oli mahdollista neuvotella palveluntarjoajan kanssa.[20]

Recast.ai kertoi tukevansa 19:ä eri kieltä ja tarjoavansa suomen kielelle tavallisen tukitason[20]. Tukitaso sisälsi intentioiden luokittelun, kustomoidut entiteetit sekä kielen tunnistuksen[20]. Teknologialla toteutettu keskustelubotti ei pystynyt kuitenkaan tunnistamaan suomenkielisiä sanoja samoiksi, vaikka sanat olisivat eronneet toisistaan vain hieman taivutukseltaan.

4.7 Rasa

Rasa NLU oli Rasan julkaisema keskusteluohjelmien kehittämiseen tehty avoimen lähdekoodin NLU-kirjasto, jota käytettiin tutkimuksessa yhdessä Rasa Core -alustan kanssa. Rasa Core oli avoimen lähdekoodin alusta kehittyneiden keskusteluohjelmien luomiselle ja sitä voitiin käyttää joko Rasa NLU:n tai muun NLU-kirjaston kanssa. Yhdessä Rasa NLU ja Core mahdollistivat hyvinkin monipuolisten keskustelubottien rakentamisen jopa omalla tietokoneella ilman yhteyttä palveluntarjoajaan.[21]

Koska Rasa NLU ja Core oli tehty käytettäväksi omalta tietokoneelta, niin alusta sekä kirjasto oli asennettava ennen teknologian käyttöä. Rasan käyttö ja asennus tapahtui tietokoneen

[21] Rasa Docs. Lisätietoa: https://rasa.com/docs

komentorivin kautta, eikä visuaalista käyttöliittymää ollut tarjolla, mikä voi tuntua aluksi vaikealta komentorivin käyttöön tottumattomalle. Rasa tarjosi asennukseen kuitenkin hyvän ohjeistuksen kotisivullaan, minkä ansiosta varsin kokemattomankin käyttäjän oli mahdollista saada Rasa toimimaan omalla kotikoneellaan.[21]

Kun Rasa Core ja NLU oli saatu asennettua, niin ensimmäisten NLU-mallien koulutus voitiin aloittaa. Tämä tapahtui yksinkertaistetusti määrittelemällä aluksi intentiot sekä näiden koulutusaineisto nlu.md-tiedostoon Rasan dokumentaation ohjeistuksen mukaisesti. Tämän jälkeen luotiin keskustelubotin nluconfig.yml-tiedosto, jonne määriteltiin botin toimintaan liittyvät asetukset. Tämän jälkeen botti voitiin kouluttaa antamalla opetuskomento, joka käynnisti koneopetetun mallin luonnin tiedostojen ja näihin syötettyjen arvojen pohjalta. Koulutusta ja asetuksia voitiin parannella niin pitkään, kunnes haluttu lopputulos saavutettiin.[21]

Rasan parhaimpia puolia oli se, että keskustelubotti voitiin halutessa luoda toimimaan lähes millä tahansa kielellä[21]. Rasa NLU:ta ei oltu kuitenkaan valmiiksi koulutettu toimimaan eri kielillä, vaan kielten koulutus tuli toteuttaa itse. Kielen koulutusprosessi vaatikin enemmän työtä kuin muissa tutkimukseen valituissa NLU-teknologioissa, joissa tuettujen kielten kielikohtaisesta koulutuksesta iso osa oli ennalta toteutettu. Vaikka Rasa NLU ei ollut yhtä helppo ottaa käyttöön kuin muut teknologiat, niin teknologia oli mahdollista ohjelmoida toimimaan juuri käyttäjän haluamalla tavalla. Rasan ympärille oli tästä johtuen muodostunutkin laaja kehittäjäkunta, jonka ansiosta teknologialle oli saatavilla paljon valmiiksi opetettuja malleja eri kielillä.[22]

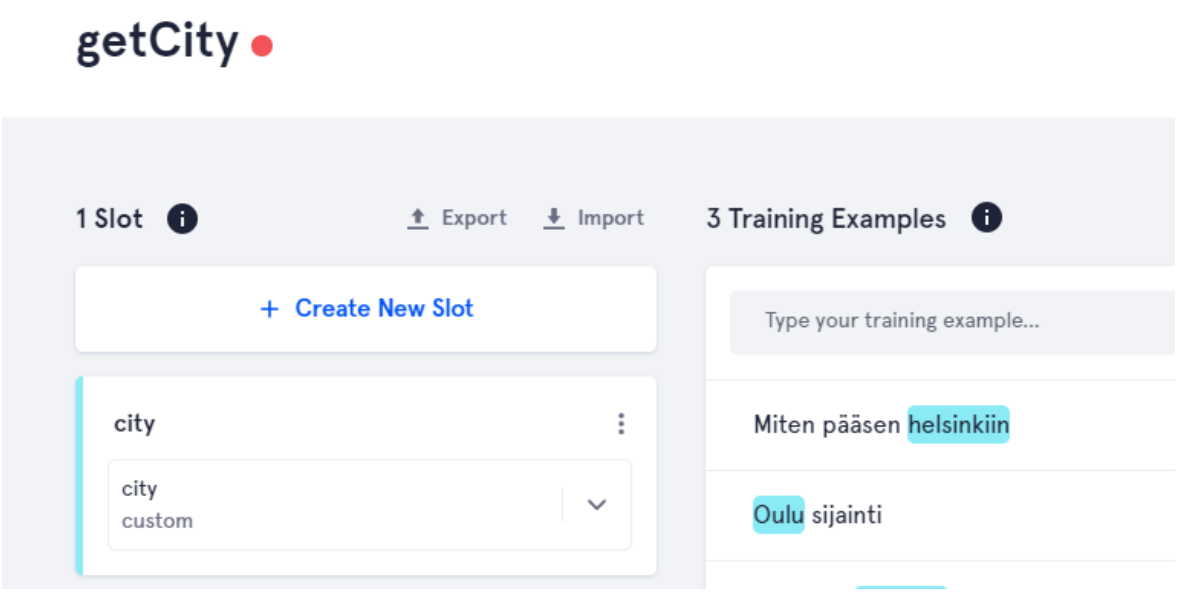
4.8 Snips

Useimmat NLU-teknologioista ovat pilvipohjaisia palveluita, jotka vastaanottavat keskustelubotille lähetetyn syötteen internetin yli ja käsittelevät tämän palvelimellaan. Snips oli tämän suhteen kuitenkin poikkeus, sillä Snips kehitti vuonna 2017 ilman internetyhteyttä toimivan NLU-teknologian. Teknologia mahdollisti botin käytön ilman internetyhteyttä käyttäjän omalla laitteella. Ainoa vaihe, missä Snips tarvitsi yhteyttä internettiin oli teknologian

[22]LanguageModels.io. Lisätietoa: http://www.languagemodels.io

koulutusvaihe. Koulutusvaihe ei kuitenkaan vaikuttanut palvelun käyttöön sen jälkeen, kun koulutuksen avulla tuotettu koneopetettu malli oli siirretty toimimaan keskustelubotille.[23]

Snipsissä keskustelubotin mallin luonti alkoi tämän nimen sekä kielen määrittelyllä käyttöliittymässä (ks. kuvio 14), jonka jälkeen mallin koulutus voitiin aloittaa joko tyhjän tai valmiiksi koulutetun mallin pohjalle. Uusien intentioiden kouluttaminen mallille tapahtui syöttämällä intentioihin liittyvää opetusdataa, johon merkattiin kuhunkin intentioon viittaavat avainsanat. Entiteettien määrittäminen tapahtui vuorostaan luomalla uusia slots-nimisiä komponentteja ja syöttämällä näiden sisältö erikseen käyttöliittymässä tai valitsemalla nämä syötetyn opetusdatan joukosta. Kun intentiot ja entiteetit oli saatu määriteltyä, niin mallin koulutus voitiin käynnistää. Koulutuksen jälkeen malli voitiin testata ja muuttaa JSON-tiedostomuotoon, joka pystyttiin siirtämään keskustelubotille.[23]



Kuvio 14. Snipsin käyttöliittymä 1/2019.

Snips antoi nettisivuilta ohjeistuksen teknologian käyttöönottoon Android-, iOS- ja Raspberry Pi -alustoille. Snips oli mahdollista saada toimimaan myös Windowsilla, mutta tämän toteutukseen ei ollut saatavilla ohjeistusta. Teknologian sisällyttäminen puhelinsovellukseen

[23]Snips. Lisätietoa: https://snips.ai

tapahtui Android-projektissa määrittelemällä aluksi teknologian sijainti projektissa ja lisäämällä tämä projektin riippuvuuksin. Tämän jälkeen koulutettu keskustelubotin malli ladattiin Snips-konsolista ja asetettiin projektissa samaan kansioon AndroidManifest.xml-tiedoston kanssa. Lopuksi projektissa tuli määritellä keskustelubotin käyttötapa ja -oikeudet. Jos sovelluksen haluttiin toimivan puheentunnistuksella, niin sovellukselle tuli ensin antaa oikeus käyttää puhelimen mikrofonia. Tämän jälkeen botti voitiin aktivoida puhelimen kautta esimerkiksi herätesanalla “Hey Snips”.[23]

Snips ei tarjonnut tukea suomen kielelle eikä kyennyt käsittelemään suomenkielisten syötteiden taivutusmuotoja. Sen sijaan Snips tarjosi tuen englannin, ranskan, saksan, italian, korean, espanjan ja japanin kielille. Snipsin vahvuus oli valmiin opetusmallin siirrettävyys erillisille alustoille verkkosivuilla tapahtuvan koulutuksen jälkeen.[23]

4.9 Johtopäätökset

Tutkimuksessa käsiteltyjen NLU-teknologioiden toiminnoissa havaittiin paljon samankaltaisuuksia. Kaikista tutkimuksessa käsitellyistä teknologioista löytyi jonkinlainen tapa intentioiden, entiteettien sekä laukaistavien toimintojen määrittelylle. Keskustelubotin koulutus tapahtui useimpien teknologioiden kohdalla hyvin samankaltaisesti, mikä nopeutti teknologiasta toiseen siirtymistä. Eniten eroja löytyi teknologioiden kielituesta, mikä näkyi siinä, että tuettujen kielten määrä vaihteli hyvinkin paljon teknologioittain. Teknologioiden tarjoama kielituki ei ollut kuitenkaan suosiollinen suomen kielen kohdalla, sillä tutkituista teknologioista ainoastaan Wit.ai ja Recast.ai kertoi tukevansa suomen kieltä. Toteutettujen keskustelubottien perusteella ei kuitenkaan huomattu, että Wit.ai:n ja Recast.ai:n suomen kielen tukitaso olisi huomattavasti eronnut muista teknologioista. Yksikään tutkituista teknologioista ei onnistunut esimerkiksi automaattisesti yhdistämään perusmuodossa olevia suomenkielisiä sanoja näiden taivutusmuotoisten sanojen kanssa.

Tutkimuksen aikana NLU-teknologioiden huomattiin kehittyvän nopeasti, mikä oli toisaalta odotettavissa, sillä useat NLU-teknologioista oltiin kehitetty vasta viime vuosikymmenen sisällä. Sen sijaan teknologioiden hinnoittelu oli yllätys, sillä kaikki käsitellyistä teknologioista tarjosivat palvelustaan myös ilmaisversion. Tutkimuksessa kävi ilmi, että tutkituilla NLU-

teknologioilla kehittäjä voi hyvinkin nopeasti luoda puheohjattuja käyttöliittymiä. Joissain tapauksissa valmiit NLU-teknologiat voivat kuitenkin rajoittaa keskustelubotin kehityksen viemistä pidemmälle, minkä vuoksi oman keskustelubotin kehittäminen esimerkiksi Rasan tarjoaman avoimen lähdekoodin päälle voi olla varteenotettava ratkaisu.

Teknologioiden tukemien kielten määrän ja tason huomattiin vaihtelevan hyvin paljon eri teknologioiden välillä. Osa teknologioista oli keskittynyt tukemaan vain puhutuimpia kieliä, kun taas useimmat tarjosivat tuen vähintään yli kymmenelle eri kielelle (ks. kuvio 15). Teknologioiden dokumentaatioiden perusteella vähiten kieliä tuki Amazon Lex, joka tarjosi kielituen ainoastaan englannin kielelle. Eniten kieliä löytyi Wit.ai:lta, joka kertoi tukevansa jopa 73:a kieltä eri tasoilla. Rasa poikkeisi kielitukensa osalta eniten muista teknologioista, sillä Rasa ei tukenut ainuttakaan kieltä heti käyttöönotossa. Rasan dokumentaation mukaan Rasa NLU:n tulisi kuitenkin pystyä tukemaan kaikkia maailman eri kieliä, jos käyttäjä antaa teknologialle kattavan kielen sanoista koostuvan opetusmallin. Tämä asetti Rasan myös erikoiseen asemaan muihin teknologioihin nähden, sillä muissa teknologioissa kielituki oli valmiiksi sisäänrakennettu. Tutkimuksen aikana syntyi kuitenkin vaikutelma, että teknologioiden dokumentaatiosta löytyvien tuettujen kielten määrittely perustui palveluntarjoajien omaan käsitykseen kielituen laajuudesta.

Kaikista tutkimuksen aikana käsitellyistä teknologioista oli saatavilla ilmaisversio (ks. kuvio 15). Ilmaisversioissa palveluun lähetettävien kutsujen määrä saattoi olla rajoitettu tai kaikki ominaisuudet eivät olleet käytettävissä maksulliseen versioon verrattuna. Kaikki tutkimuksen keskustelubotit pystyttiin toteuttamaan teknologioista saatavilla olevilla ilmaisversioilla ilman, että tällä olisi ollut vaikutusta teknologioilla toteutettuihin testeihin.

Teknologia	Tuettujen kielten määrä	Tuki suomen kielelle	Sisälsi ilmaisversion	Toimi pilvipalvelussa
Dialogflow	18	EI	KYLLÄ	KYLLÄ
Wit.ai	73	KYLLÄ	KYLLÄ	KYLLÄ
LUIS	12	EI	KYLLÄ	KYLLÄ
Watson Assistant	13	EI	KYLLÄ	KYLLÄ
Amazon Lex	1	EI	KYLLÄ	KYLLÄ
Recast.ai	19	KYLLÄ	KYLLÄ	KYLLÄ
Rasa	?	EI	KYLLÄ	EI
Snips	7	EI	KYLLÄ	EI

Kuvio 15. Suomen kielellä testatut NLU-teknologiat.

Teknologioista kaikki muut paitsi Rasa ja Snips toimivat pilvipalvelussa, mikä mahdollisti useimpien keskustelubottien testaamisen helposti joko teknologioiden omalla testausalustalla tai rajapintojen kautta (ks. kuvio 15). Pilvessä toimivien teknologioiden käyttöönotto koettiin testauksen aikana yleisesti helpommaksi kuin laitteelle asennettavien teknologioiden. Snipsin kohdalla keskustelubotin koulutusvaihe tapahtui kuitenkin pilvipalvelussa, minkä jälkeen koulutettu malli voitiin siirtää toimimaan omalle alustalle. Pilvipalveluita täysin hyödyntävien teknologioiden kanssa käyttäjä oli aina riippuvainen pilvipalvelun tarjoajasta, jolloin keskustelubottien käyttö ei ollut mahdollista esimerkiksi ilman toimivaa internetyhteyttä.

Keskustelubottien kokeilussa huomattiin, ettei yksikään teknologioista osannut yhdistää perusmuotoisia suomenkielisiä sanoja näiden taivutusmuotoisten sanojen kanssa, mistä johtuen sama sana voitiin joutua kouluttamaan hyvinkin usealla eri tavalla. Rasan kohdalla koulutusaineiston syötössä voitiin hyödyntää kuitenkin tiedostopohjaista hallintaa, minkä takia suomen kielen koulutus isossa mittakaavassa olisi luultavasti onnistunut kaikista parhaiten Rasan avulla. Rasa oli kuitenkin muihin teknologoihin verrattuna lähempänä keskustelubotin

tekoon tarkoitettua ohjelmistokirjastoa kuin valmista helposti käyttöönotettavaa teknologia-palvelua, minkä takia Rasa ei ollut kunnolla verrattavissa muihin käsiteltyihin teknologioihin.

5 Watson Assistantin ja Wit.ai:n suorituskyky suomen kielen käsittelyssä

Tutkielman toisessa tutkimusosuudessa mitattiin Watson Assistantin ja Wit.ai:n suorituskykyä käsitellä suomen kieltä. Tutkimukseen valituissa teknologioissa haluttiin olevan jonkin verran eroavaisuuksia, mutta samalla tarpeeksi paljon samankaltaisuuksia, jotta tuloksia olisi järkevä vertailla keskenään. Tutkimuksen ajankohdalla Watson Assistant ei virallisesti tukenut suomen kieltä siinä missä Wit.ai tuki (ks. luku 4.9). Luvussa 4.9 ja Watson Health Cloud Finland -projektissa (Hänninen ym. 2018) saatujen tutkimustulosten perusteella Wit.ai ja Watson Assistant pystyivät tunnistamaan syötteistä suomenkielisiä sanoja siinä määrin, kuin sanoja oli teknologioille opetettu.

Luvussa 4.9 ja Watson Health Cloud Finland -projektissa (Hänninen ym. 2018) myös todettiin, että NLU-teknologioiden on vaikeaa tunnistaa sanataivutuksista ja yhdyssanoista niille opetettuja entiteettejä. Tämän huomion perusteella tässä tutkimusosuudessa haluttiin tutkia, miten syötteiden normalisointi vaikuttaa teknologioiden suorituskykyyn käsitellä suomen kieltä. Tutkimuksen normalisointitekniikaksi valittiin lemmaus, joka luvussa 2.4.3 esitettyjen tutkimusten perusteella soveltuu hyvin suomenkielisten sanojen normalisointiin.

5.1 Suorituskykyä mittaavat tutkimukset

NLU-teknologioiden suorituskykyyn kantaaottavia vertaisarvioituja tutkimuksia on vähän, sillä teknologiat ovat vielä varsin uusi ilmiö ja ne muuttuvat nopealla tahdilla. Suurin osa NLU-teknologioiden suorituskykyvertailusta on toteutettu erilaisten yritysten toimesta. Suorituskyvyn mittaukset perustuivat yleensä siihen, miten hyvin teknologiat tunnistivat syötteistä niille opetettuja entiteettejä ja miten nopeasti teknologiat suoriutuivat syötteiden käsittelystä.

Münchenin teknillisen yliopiston tutkimuksessa (Braun, Mendez ja Matthes 2017) vertailtiin NLU-teknologioiden tarkkuutta löytää syötteistä entiteettejä. Vertailtavina NLU-teknologioina olivat LUIS, Watson Assistant, Dialogflow ja Rasa. Teknologioissa käytettiin täysin sa-

maa opetusdataa, millä haluttiin varmistaa yhdenmukaiset testausasetelmat kaikille teknologioille. Teknologioiden testaamiseen käytettiin 206 erilaista syötettä, joista teknologioiden tuli löytää ennalta opetetut entiteetit. Etsittävänä oli viisi erilaista entiteettiä, joista osa esiintyi syötteissä useasti ja osa harvoin. Syötteet olivat englanninkielisiä, mutta niissä esiintyi myös saksankielisiä teiden ja asemien nimiä. Etsittävistä entiteeteistä LUIS löysi 94,5 %, Dialogflow 87,1 %, Rasa 78,9 % ja Watson Assistant 73,8 %. (Braun, Mendez ja Matthes 2017)

Intento-nimisen yrityksen toteuttamassa tutkimuksessa[24] testattiin NLU-teknologioiden suorituskykyä tutkimalla teknologioiden oppimisnopeutta, prosessointinopeutta ja kykyä löytää entiteettejä syötteistä. Testattavina teknologioina olivat Watson Assistant, Wit.ai, Dialogflow, LUIS, Amazon Lex, Recast.ai ja Snips. Tutkimuksessa käytettiin laajaa englanninkielistä opetusdataa, jonka avulla teknologiat opetettiin samankaltaisesti. Tutkimuksessa teknologioiden tunnistamistarkkuus oli hyvä, sillä opetetuista entiteeteistä Dialogflow löysi 99,3 %, Watson Assistant 99,2 %, LUIS 98,8 %, Snips 98,8 %, Recast.ai 97,5 % ja Amazon Lex 96,3 %. Tunnistamistarkkuutta kokeiltiin aluksi myös pienemmällä opetusdatalla, jolloin Watson Assistant suoriutui muita teknologioita paremmin. Tunnistamistarkkuuden lisäksi tutkimuksessa mitattiin keskimääräistä prosessointinopeutta, joka teknologioilla kului syötteiden käsittelyyn. Prosessointinopeudet teknologioittain olivat LUIS 210 ms, Dialogflow 280 ms, Watson Assistant 350 ms, Snips 360 ms, Amazon Lex 430 ms, Wit.ai 960 ms ja Recast.ai 2060 ms.[24]

Tutkimuksissa esitetyt entiteettien tunnistustarkkuudet eroavat huomattavasti toisistaan, mikä johtuu todennäköisesti tutkimuksissa käytetyn opetusdatan ja syötteiden eroavaisuuksista. NLU-teknologioiden suorituskykyvertailua on toteutettu myös muiden yritysten toimesta, mutta vertailuissa käytettävä opetusdata ja syötteet saattavat suosia tiettyä NLU-teknologiaa, mikä näkyy myös vertailun tuloksissa[25]. Yritysten tuottamiin vertailutuloksiin kannattaakin siis suhtautua kriittisesti.

[24] NLU / Intent Detection Benchmark by Intento. Lisätietoa: https://www.slideshare.net/KonstantinSavenkov/nlu-intent-detection-benchmark-by-intento-august-2017

[25] NLU Comparison between Clare.AI, IBM Watson, and Dialogflow. Lisätietoa: https://blog.clare.ai/2019/03/22/nlu-comparison-between-clare-ai-ibm-watson-and-dialogflow/

5.2 Tutkimusasetelma

Tässä tutkimusosuudessa tullaan ottamaan mallia luvussa 5.1 esitetyistä tutkimuksista mittaamalla Watson Assistantin ja Wit.ai:n entiteettien tunnistustarkkuutta sekä prosessointinopeutta suomen kielellä. Tunnistustarkkuudella ilmaistaan prosenttiyksikköä, joka kertoo, kuinka monta ennalta opetettua entiteettiä teknologia löysi syötteistä. Prosessointinopeudella puolestaan kuvataan aikaa, joka teknologialla kului keskimäärin syötteen käsittelyssä. Tutkimusosuus ei kuitenkaan keskity täysin näiden kahden NLU-teknologian keskinäiseen vertailuun, vaan tutkimuksessa haluttiin myös vertailla, miten syötteiden lemmaus vaikuttaa NLU-teknologioiden suorituskykyyn.

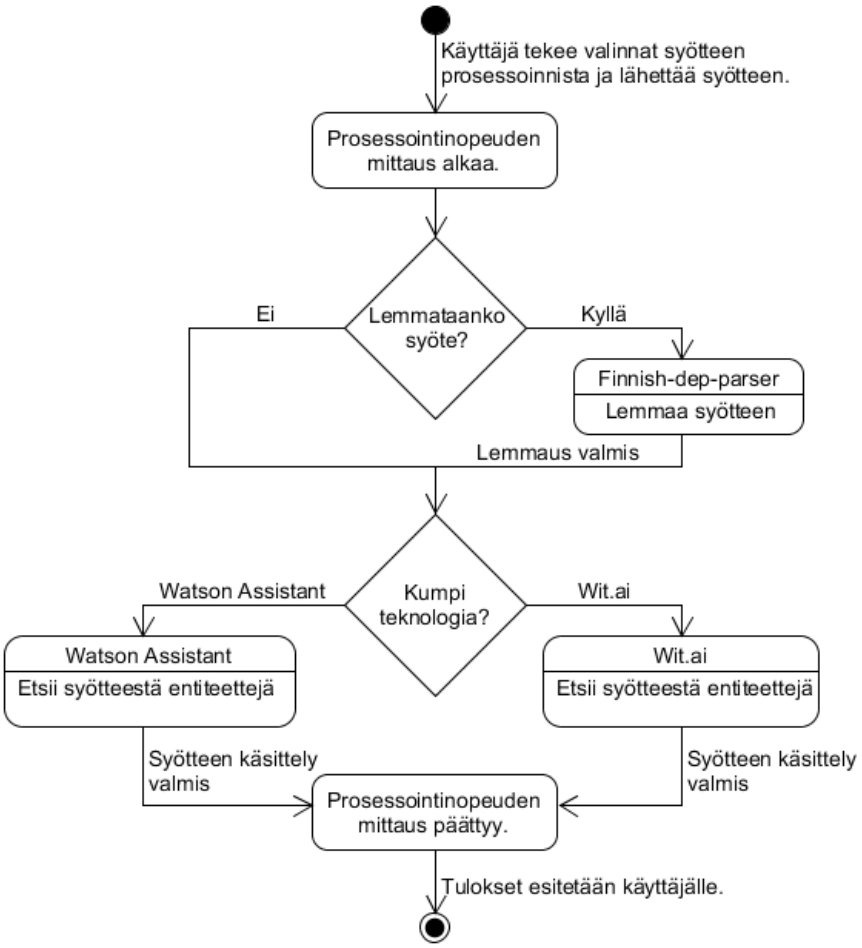
Tutkimusta varten suunniteltiin opetusdata, joka koostui 42:sta suomenkielisestä sanasta. Sanat opetettiin testattaville teknologioille samalla tavalla, minkä jälkeen teknologioiden tuli tunnistaa opetetut sanat entiteetteinä. Tutkimuksessa haluttiin vertailla kielituen ja normalisoinnin vaikutusta, minkä vuoksi sanat opetettiin teknologioille ainoastaan perusmuodossa. Teknologioille opetetut entiteetit löytyvät liitteestä A.

Teknologioiden kykyä tunnistaa entiteettejä testattiin 58 erilaisella testisyötteellä, joiden pituudet vaihtelivat yksittäisistä sanoista muutamiin lauseisiin. Jokaisessa teknologialle lähetetyssä syötteessä oli yhdestä seitsemään löydettävää entiteettiä. Esimerkiksi syötteestä “Haluaisin ostaa pippuria ja suolaa.” teknologioiden kuului löytää entiteetit “pippuri” ja “suola”. Puolet teknologioille lähetetyistä syötteistä olivat perusmuodossa ja puolet taivutusmuodossa. Taivutusmuodossa olevien syötteiden kohdalla haluttiin nähdä, pystyvätkö teknologiat löytämään taivutettujen sanojen entiteetit. Tutkimuksessa käytetyt syötteet löytyvät liitteestä B.

Teknologioiden testausprosessi toteutettiin kahdessa eri iteraatiossa. Ensimmäisessä iteraatiossa teknologiat käsittelivät syötteet liitteessä B esitetyssä muodossa. Toisessa iteraatiossa käytettiin samoja syötteitä, mutta syötteet lemmattiin ennen teknologioiden käsittelyä. Samoja syötteitä syötettiin teknologioille kymmenen kertaa, jotta dataa kertyisi mahdollisimman paljon prosessointinopeuden keskiarvon ja -hajonnan laskemiseksi. Datan kerääminen jaettiin viidelle eri päivälle, jotta mahdolliset internetyhteydestä johtuvat viiveet eivät näkyisi tuloksissa.

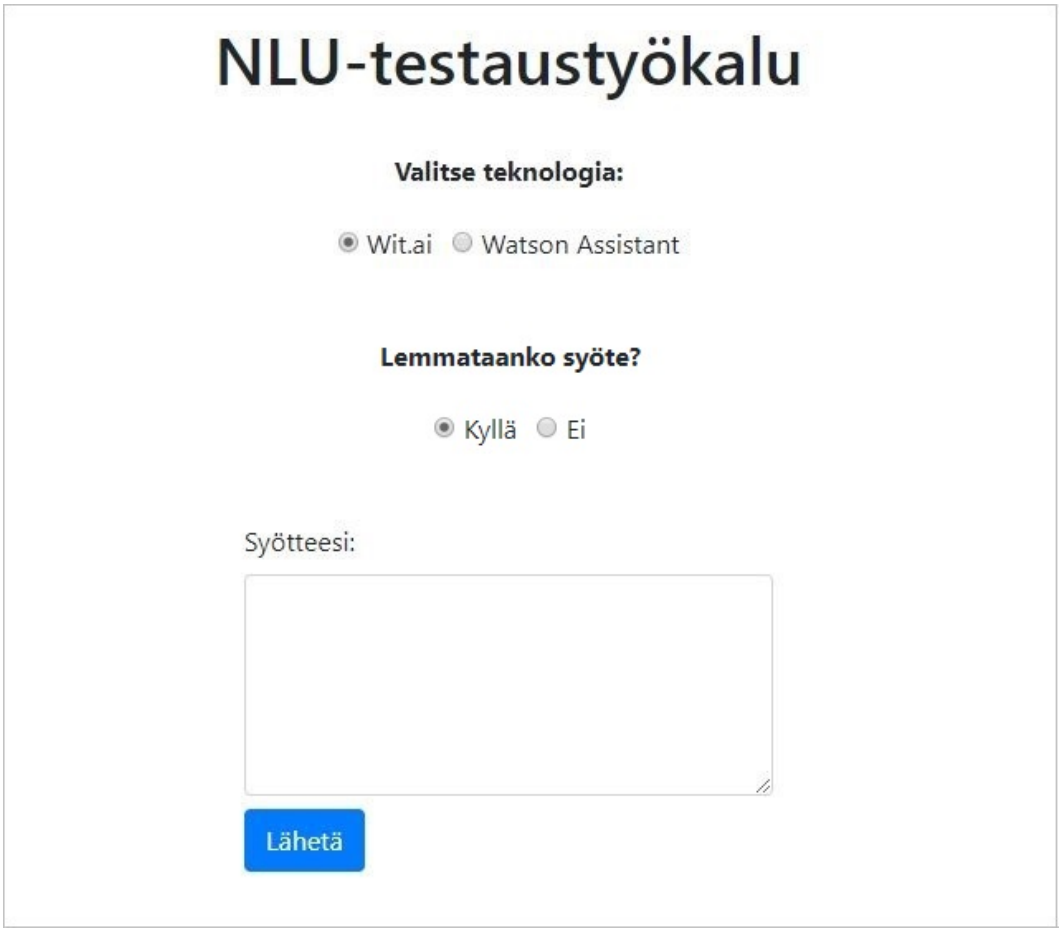
5.3 Testaustyökalu

Tutkimusta varten toteutettiin aluksi web-pohjainen testaustyökalu, joka mahdollisti NLU-teknologioiden suorituskyvyn mittaamisen. Testaustyökalulla haluttiin mahdollistaa syötteiden lähettäminen ja datan kerääminen graafisen käyttöliittymän avulla. Testaustyökalulla pystyttiin valitsemaan NLU-teknologia, jolle voitiin lähettää halutunlainen tekstisyöte. Testaustyökalussa pystyttiin valitsemaan toteutetaanko syötteelle lemmaus ennen NLU-teknologian käsittelyä. NLU-teknologian käsittelyn jälkeen testaustyökalu esitti teknologialta saadun palautteen, josta käy ilmi millaisia entiteettejä teknologia löysi syötteestä ja miten pitkään teknologialla meni syötteen käsittelyyn. Testaustyökalun toimintaprosessia on kuvattu tilakaaviolla kuviossa 16.



Kuvio 16. Tilakaavio testaustyökalun toiminnasta.

Testaustyökalun käyttöliittymä on tehty Vue.js-ohjelmointikirjastolla. Käyttöliittymässä teknologian ja lemmauksen valinta on toteutettu yksinkertaisten valintapainikkeiden avulla. Valintapainikkeiden lisäksi käyttöliittymässä on tekstikenttä, johon NLU-teknologialle lähetettävä syöte kirjoitetaan. Testaustyökalu toimii ainoastaan Wit.ai- ja Watson Assistant -teknologioiden testaamiseen, mutta testaustyökalu toteutettiin siten, että myös muiden teknologioiden lisääminen olisi mahdollisimman helppoa. Testaustyökalun käyttöliittymä on esitetty kuviossa 17.



Kuvio 17. Testaustyökalun käyttöliittymä.

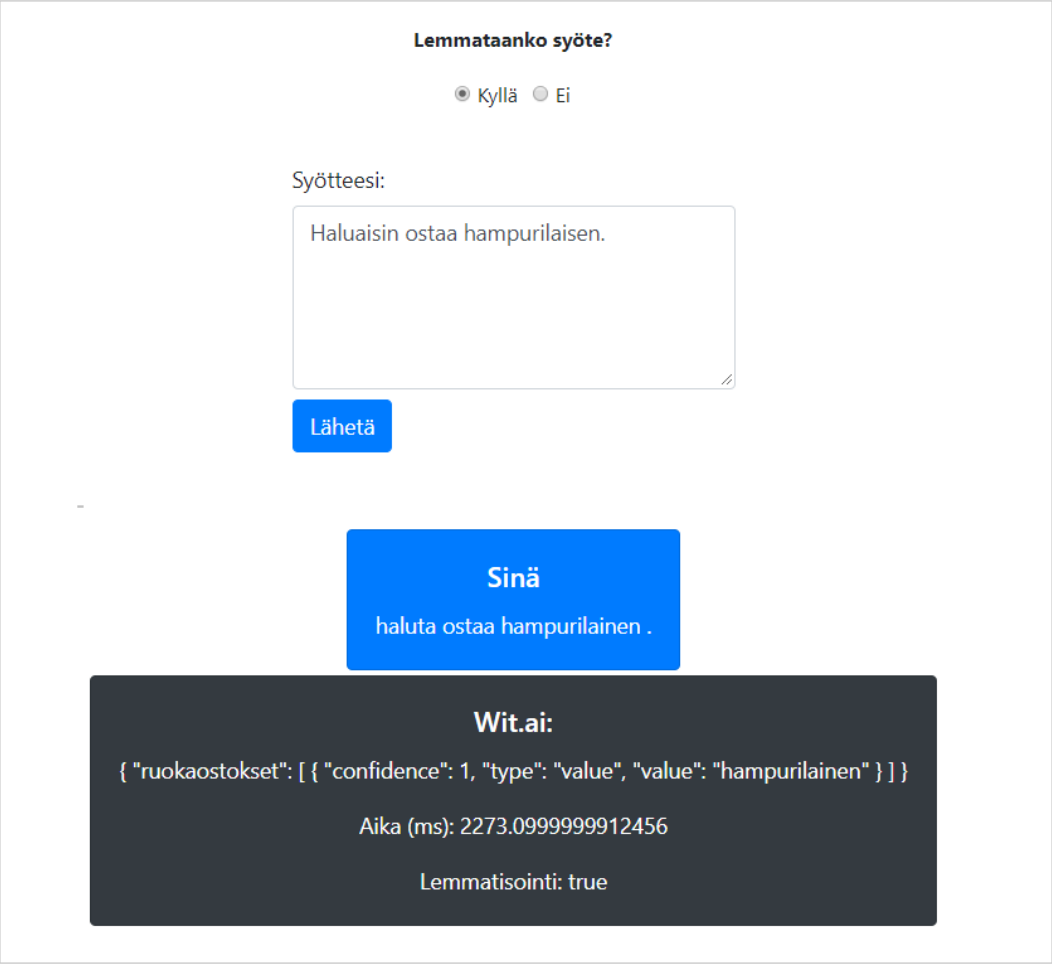
Testaustyökalun palvelinlogiikka toteutettiin Python Flask -ohjelmointikielellä. Ohjelma lähettää käyttäjän syötteen prosessikutsuna sille NLU-teknologialle, jonka käyttäjä on käyttöliittymässä valinnut. Ohjelma käyttää syötteiden lähettämiseen NLU-teknologioiden tarjoamia API-rajapintoja, joiden käyttöönottoa varten teknologiat tarjosivat ohjeet dokumentaatiosivuillaan[12,16]. Teknologioiden API-rajapintojen käyttö oli tehty helpoksi, sillä käyt-

töönotossa täytyi määritellä ainoastaan rajapinnan varmennetiedot. Kun varmennetiedot oli määritelty, teknologioille voitiin lähettää tekstimuotoisia syötteitä, joihin teknologia antoi JSON-muotoiset vastaukset. Liitteessä C olevassa koodissa on esitetty, miten prosessikutsut toteutettiin tutkimuksessa käytettyihin teknologioihin. Koodissa on näkyvillä myös varmennetiedot, joita käyttämällä kuka tahansa pystyy testaamaan tutkimuksessa käytettyjä teknologioita ja voi halutessaan toistaa tutkimuksessa toteutetut testit.

Testaustyökalun lemmausominaisuus toteutettiin NLP-työkalulla nimeltä Finnish-dep-parser[26] (FDP). FDP:stä oli saatavilla myös uudempi versio TNPP[2], mutta ainoastaan FDP oli käytettävissä Jyväskylän yliopiston palvelimella. FDP:n käyttö palvelimelta käsin mukaili oikeaa käyttötapausta, jossa vastaavaa sovellusta käytettäisiin todennäköisimmin palvelimen kautta. FDP:n rajapintayhteyden toteutus on esitetty liitteessä D.

Kuviossa 18 on esitetty esimerkkitilanne testaustyökalun käytöstä. Esimerkkitilanteessa syötteeksi on asetettu teksti “Haluaisin ostaa hampurilaisen.”, sekä tehty valinta syötteen lemmauksesta ennen Wit.ai:n käsittelyä. Kuviossa 18 näkyy, miten FDP on lemmannut syötteen muotoon “haluta ostaa hampurilainen .”. Samassa kuviossa näkyy myös Wit.ai:n palauttama JSON-muotoinen vastaus, josta käy ilmi, että Wit.ai löysi syötteestä entiteetin “hampurilainen” varmuudella yksi. Varmuuden arvo yksi tarkoittaa sitä, että teknologia on täysin varma siitä, että kyseinen entiteetti löytyy syötteestä. Jos arvo olisi jokin desimaaliluku nollan ja yhden väliltä, teknologia ei olisi varma löydetystä entiteetistä. Epävarmassa tapauksessa entiteetti olisi voinut löytyä esimerkiksi taivutusmuodossa olevasta sanasta, jossa entiteetti esiintyy sanan osamerkkijonona.

[26]Finnish-dep-parser GitHub. Lisätietoa: https://github.com/TurkuNLP/Finnish-dep-parser



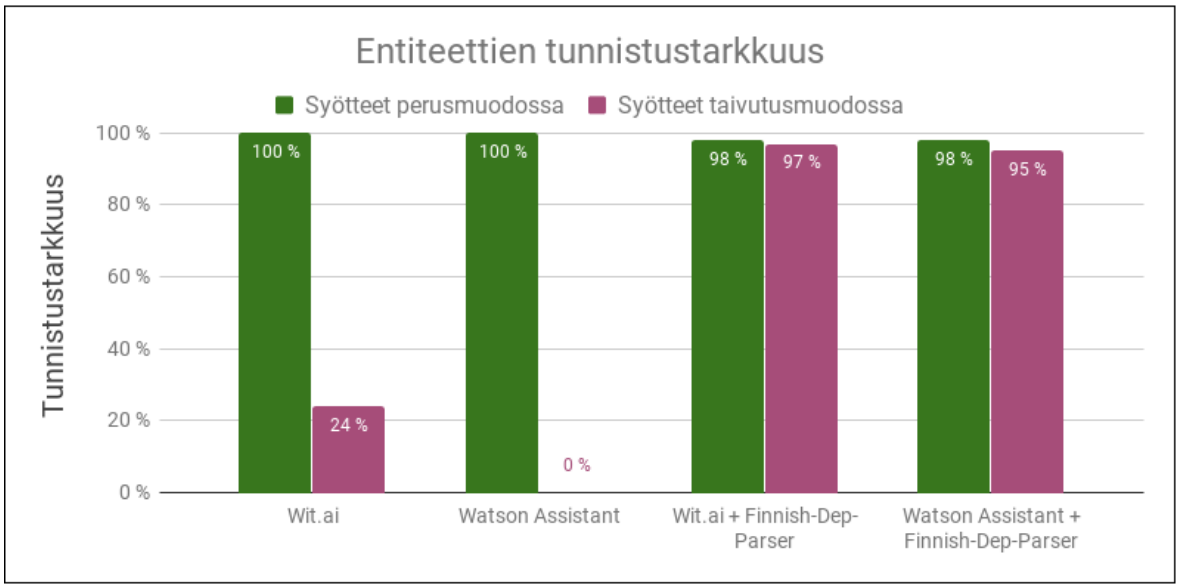
Kuvio 18. Testaustyökalun toiminta.

5.4 Entiteettien tunnistustarkkuus

Entiteettien tunnistustarkkuuden mittauksessa molemmat teknologioista löysivät perusmuodossa olevista syötteistä kaikki ennalta opetetut entiteetit eli yhteensä 59 entiteettiä. Syötteiden ollessa taivutusmuodossa Wit.ai löysi 14 entiteettiä eli 24 % kaikista entiteeteistä. Watson Assistant ei puolestaan löytänyt yhtään entiteettiä taivutusmuodossa olevista syötteistä.

Syötteiden lemmauksella oli huomattava vaikutus teknologioiden kykyyn tunnistaa syötteiden entiteettejä. Lemmauksen avulla Wit.ai tunnisti taivutusmuodossa olevista syötteistä 97 % entiteeteistä ja Watson Assistant 95 %. Lemmauksella oli vaikutusta myös taivutusmuodossa oleviin syötteisiin, sillä molemmilta teknologioilta jäi yksi entiteetti tunnistamatta, mi-

kä tarkoitti 98 %:n tunnistustarkkuutta. Yhden entiteetin tunnistamatta jääminen johtui siitä, että FDP lemmasi yhden syötteestä etsittävän entiteetin väärään muotoon. Teknologioiden tunnistustarkkuuden tuloksia on havainnollistettu kuviossa 19. Kuvioissa esiintyvät prosentit kuvaavat sitä, kuinka monta entiteettiä teknologia löysi niin perus- kuin taivutusmuodossa olevista syötteistä.



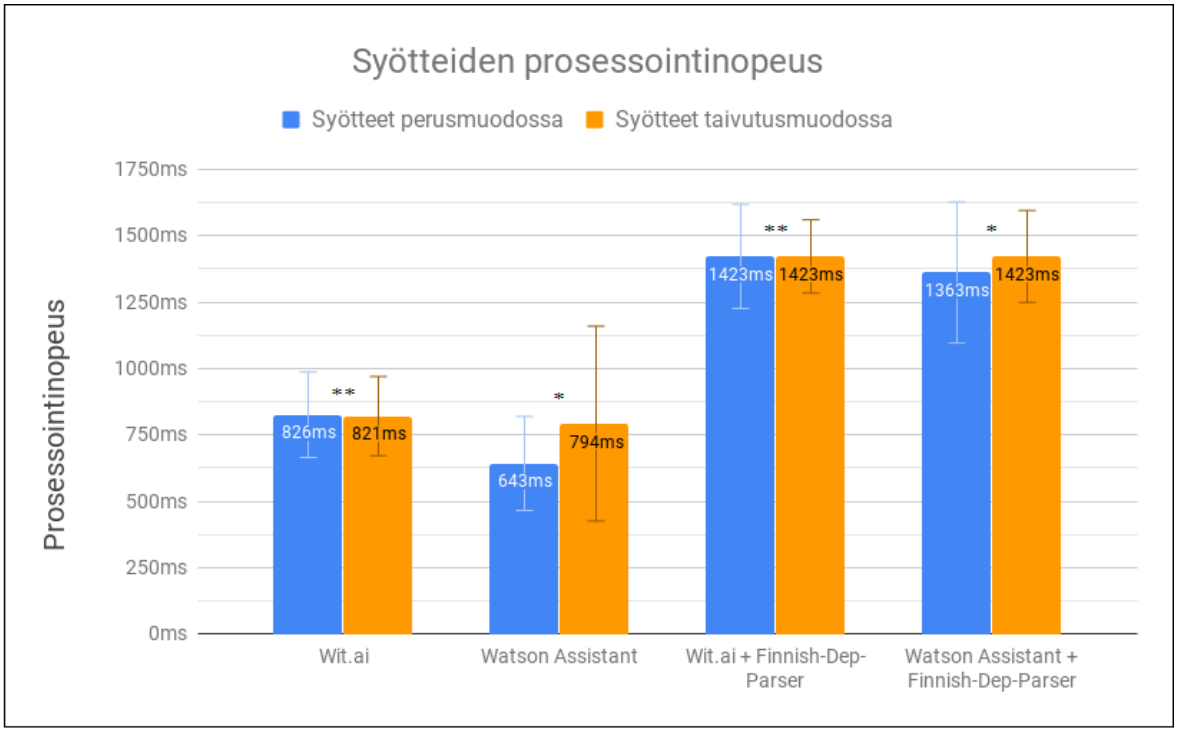
Kuvio 19. Entiteettien tunnistustarkkuus syötteistä.

5.5 Prosessointinopeus

Toisena mitattavana ominaisuutena oli prosessointinopeus eli kuinka kauan teknologialla kului aikaa syötteen käsittelyssä. Ajan mittaaminen alkoi syötteen lähetyksestä ja lopetettiin, kun teknologialta saatiin vaste takaisin. Syötteiden ollessa perusmuodossa Wit.ai:n prosessointinopeuden keskiarvo oli 826 ms ja Watson Assistantin 643 ms. Syötteiden ollessa taivutusmuodossa Wit.ai:n prosessointinopeuden keskiarvo oli 821 ms ja Watson Assistantin 794 ms.

Syötteiden lemmaus hidasti huomattavasti prosessointinopeutta, koska syötteet lähetettiin ensin Finnish-dep-parserille ja vasta sen jälkeen teknologioiden käsiteltäväksi. Lemmatuissa syötteissä Wit.ai:n perus- ja taivutusmuodossa olevien syötteiden prosessointinopeus oli keskiarvolta 1423 ms. Watson Assistantilla puolestaan kului aikaa perusmuodossa oleviin

syötteisiin keskiarvolta 1363 ms ja taivutusmuodossa oleviin syötteisiin keskiarvolta 1423 ms. Teknologioiden prosessointinopeuden tuloksia on havainnollistettu kuviossa 20.



Kuvio 20. Syötteiden prosessointinopeudet keskiarvoina ja -hajontoina. * $p < 0{,}05$ ja ** $p > 0{,}05$.

Kuvioissa 20 on esitetty prosessointinopeuksien lisäksi myös nopeuksien keskihajonta eli miten kaukana havainnot ovat esitetystä keskiarvosta. Keskihajonnan tulokset olivat teknologioittain samankaltaisia ja arvot sijoittuivat arvovälille 150-200. Ainoana poikkeuksena oli Watson Assistantin taivutusmuodossa olevien syötteiden käsittely, jossa keskihajonnan arvo oli 378. Keskihajonta laskettiin kaavalla

$$s = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \ ,$$

missä N on prosessointinopeuksien määrä, x_i on jokaisen saadun prosessointinopeuden arvo ja $\bar{x}$ on prosessointinopeuksien keskiarvo.

Prosessointinopeuksien keskiarvoille toteutettiin t-testi, jonka avulla haluttiin vertailla, on-

ko perus- ja taivutusmuodossa olevien syötteiden prosessointinopeuksien keskiarvojen erot systemaattisia vai satunnaisia. T-testille asetettiin riskiraja 0,05, jota pienemmät *p*-arvot kertoivat havaitun eron olevan systemaattista ja suuremmat *p*-arvot havaitun eron olevan satunnaista. Wit.ai:n kohdalla ilman syötteiden lemmausta *p*-arvo oli 0,68 ja lemmatuilla syötteillä 0,97 eli havaittu ero oli satunnaista. Watson Assistantin kohdalla *p*-arvo ilman lemmausta oli 0,000000001 ja lemmatuilla syötteillä 0,006 eli havaittu ero oli systemaattista.

5.6 Johtopäätökset

Tutkimusosuuden tuloksista voidaan todeta, että Wit.ai ja Watson Assistant pystyivät käsittelemään suomenkielisiä entiteettejä siinä muodossa kuin ne oltiin teknologioille opetettu. Wit.ai:n tuloksia analysoimalla huomattiin, että Wit.ai tunnisti sille opetetut entiteetit myös sellaisista syötteistä, joissa etsittävässä entiteetissä oli yhden kirjaimen ero. Tämä ominaisuus mahdollisti sen, että tunnistettavan sanan loppuun pystyttiin liittämään yhden kirjaimen pituinen taivutuspääte. Esimerkiksi sanasta “kahvia” Wit.ai pystyisi tunnistamaan vielä sille opetetun entiteetin “kahvi”. Tämä ominaisuus voidaan nähdä osana suomen kielen tukea, jonka Wit.ai ilmoitti sisältävänsä tutkimuksen aikana. Watson Assistant ei kertonut tukevansa suomen kieltä, joten lähtökohtana voitiin olettaa, ettei se suoriutuisi yhtä hyvin suomenkielisten syötteiden käsittelystä kuin Wi.ai. Watson Assistant onnistui kuitenkin syötteiden käsittelyssä yllättävän hyvin, joka vahvistaa Jyväskylän yliopiston tutkimusryhmän väitteen siitä, että Watson Assistant toimii osittain kieliriippumattomasti (Hänninen ym. 2018).

Testattaville NLU-teknologioille suurimmaksi ongelmaksi muodostui entiteettien tunnistaminen taivutusmuodoista. Taivutusmuodossa olevan entiteetin tunnistus vaatisi sen, että perusmuotojen lisäksi teknologioille opetettaisiin kaikki sanojen taivutusmuodot. Taivutusmuotojen opettaminen muuttuu kuitenkin todella työlääksi entiteettien määrän kasvaessa. Englannin kielessä tällaista ongelmaa ei ole, sillä sanojen taivutus tapahtuu suurimmaksi osaksi prepositioiden avulla. Yleisesti voidaan todeta, että teknologia saadaan tunnistamaan suomen kielisiä entiteettejä niin hyvin kuin kehittäjä jaksaa opettaa teknologialle tarvittavia taivutusmuotoja.

Syötteiden lemmaus Finnish-dep-parserin avulla mahdollisti sen, että teknologiat pystyivät

tunnistamaan syötteistä lähes kaikki opetetut entiteetit. Lemmatuissa syötteissä teknologioilta jäi vain kolme entiteettiä tunnistamatta. Sanan “tee” Finnish-dep-parser lemmasi muotoon “tehdä”, sanan “hampurilaisessa” muotoon “Hampuri” ja sanan “currya” FPD jätti lemmaamatta. FDP siis lemmasi kaksi sanaa tutkimuksen näkökulmasta väärin ja yhtä vierasperäistä sanaa FDP ei osannut lemmata ollenkaan. Väärin lemmatut sanat johtuivat todennäköisesti siitä, että FDP ei pystynyt tunnistamaan lauseesta sanojen kontekstia (luku 2.4).

Syötteiden prosessointinopeuden tuloksista nähtiin, että ilman lemmausta Watson Assistant oli hieman nopeampi syötteiden käsittelyssä kuin Wit.ai. Watson Assistantin tehokkaampi prosessointinopeus ei kuitenkaan tullut yllätyksenä, sillä kuten luvussa 5.1 todettiin, Watson Assistant on tarjonnut myös aiemmissa tutkimuksissa nopeampaa prosessointinopeutta Wit.ai:hin nähden. Teknologioiden välinen ero prosessointiajassa oli kuitenkin niin pieni, ettei sillä välttämättä ole käytännön tilanteessa suurta merkitystä. Syötteiden lemmaus keskimäärin tuplasi prosessointiin kuluvan ajan, koska syöte täytyi ensin lähettää prosessikutsuna Finnish-dep-parserille käsiteltäväksi. Teknologioiden välinen ero prosessointiajassa pieneni lemmauksen myötä.

Prosessointinopeuksien tuloksista myös huomattiin, että Watson Assistantilla kesti keskimäärin enemmän aikaa taivutusmuodossa olevien syötteiden käsittelyssä verrattuna perusmuodossa oleviin syötteisiin. Watson Assistantilla näytti siis kuluvan enemmän aikaa syötteiden käsittelyssä, josta se ei löytänyt yhtään entiteettiä. Wit.ai:n kohdalla tällaista tapausta ei todettu, vaan teknologia käsitteli syötteitä yhtä pitkään riippumatta siitä, löytyikö syötteistä entiteettejä vai ei.

6 Yhteenveto ja pohdinta

Tässä tutkielmassa haluttiin antaa lukijalle ymmärrys siitä, miten NLU-teknologiat soveltuvat suomen kielen käsittelyyn ja miten teknologioiden suomen kielen käsittelyä voidaan parantaa. Tutkituista NLU-teknologioista suomen kielelle tuen tarjosi ainoastaan Wit.ai ja Recast.ai, mutta näiden teknologioiden suomen kielen tuki ei vaikuttanut olevan merkittävästi parempi verrattuna muiden teknologioiden kykyyn käsitellä suomen kieltä. NLU-teknologioiden suurin ongelma suomen kielen käsittelyssä todettiin olevan entiteettien löytäminen taivutusmuodossa olevista sanoista. NLU-teknologiat pystyivät käsittelemään suomenkielisiä sanoja ainoastaan siinä muodossa, jossa sanat oltiin teknologioille opetettu. Teknologioille olisi pitänyt siis opettaa perusmuotojen lisäksi myös sanojen kaikki taivutusmuodot, jotta teknologiat olisivat kyenneet löytämään syötteiden entiteetit.

Taivutusmuodossa olevien sanojen käsittelyyn liittyvä ongelma onnistuttiin ratkaisemaan lemmauksella, jonka avulla syötteiden sanat voitiin muuttaa perusmuotoon ennen syötteiden lähetystä teknologioille. Syötteiden lemmauksen huono puoli oli se, että lemmaus kasvatti syötteiden käsittelyyn kuluvaa aikaa huomattavasti. Aikaa kului siihen, että syöte lähetettiin ensin Finnish-dep-parserin käsiteltäväksi ja vasta sen jälkeen teknologialle. Lemmaukseen kuluvaa aikaa voitaisiin todennäköisesti vähentää sillä, että lemmausominaisuus olisi valmiiksi sisäänrakennettu NLU-teknologioihin. Lemmausominaisuuden avulla syötteen käsittely onnistuisi yhdellä prosessikutsulla, jolloin myös prosessiin kuluva aika pienenisi huomattavasti.

Vaihtoehtoisena ratkaisuna lemmaukselle nähtiin kielenkääntäjät, joiden avulla suomenkieliset syötteet oltaisiin voitu kääntää englanninkielisiksi ennen syötteiden lähettämistä NLU-teknologioille. Kielenkääntäjien avulla NLU-teknologiat voitaisiin opettaa tunnistamaan englannin kielisiä entiteettejä eikä niille täten tarvitsisi opettaa kaikkia suomenkielisiä sanataivutuksia, sillä englannin kielessä sanat taipuvat prepositioiden ja postpositioiden kautta. Watson Assistant tarjosi palveluunsa yhteensopivaa Watson Language Translator -palvelua[27], joskin palvelu ei tukenut suomen kielen kääntämistä. Suomen kielen kääntämiseen oltaisiin

[27] Watson Language Translator. Lisätietoa: https://www.ibm.com/watson/services/language-translator

tarvittu siis jotain toista kielenkääntäjää, kuten Google Translation -palvelua[28].

Mahdollisissa jatkotutkimuksissa voitaisiin toteuttaa luvussa 5 esitettyä suorituskykyvertailua Watson Assistantin ja Wit.ai:n lisäksi myös muille luvussa 4 esitetyille NLU-teknologioille. Jatkotutkimuksissa voitaisiin myös vertailla, päästäänkö suomen kielen tunnistamisessa parempiin tuloksiin lemmauksen vai kielenkääntäjän avulla. NLU-teknologioiden toimivuutta voitaisiin käsitellä myös täysin uudesta näkökulmasta, kuten millä tasolla NLU-teknologioiden tarjoamat puheentunnistusominaisuudet ovat suomen kielen näkökulmasta.

[28]Google Translation API. Lisätietoa: https://cloud.google.com/translate

Lähteet

Alho, T., P. Neittaanmäki, P. Hänninen ja O. Tammilehto. 2018. *Humanoidirobotti Pepper - mahdollisuuksia ja haasteita.* Informaatioteknologian tiedekunnan julkaisuja No. 61/2018, Jyväskylän yliopisto.

Biemann, C. 2009. *Unsupervised Part-of-Speech Tagging in the Large.* Reseach on Language & Computation 7:101-135.

Bisk, Y., D. Yuret ja D. Marcu. 2016. *Natural Language Communication with Robots.* Association for Computational Linguistics.

Braun, D., A.H. Mendez ja F. Matthes. 2017. *Evaluating Natural Language Understanding Services for Conversational Question Answering Systems.* Association for Computational Linguistics.

Chomsky, N. 1957. *Syntactic Structures.* Mouton Co.

Goldberg, Y. 2016. *A Primer on Neural Network Models for Natural Language Processing.* AI Access Foundation, Journal of Artificial Intelligence Research 57.

Habert, B., G. Adda, M. Adda-Decker, P. Boula de Mareuil, S. Ferrari, O. Ferret, G. Illouz ja P. Paroubek. 1998. *Towards Tokenization Evaluation.* Proceedings First International Conference on Language Resources / Evaluation, volume I.

Haruechaiyasak, C., ja A. Kongthon. 2013. *LexToPlus: A Thai Lexeme Tokenization and Normalization Tool.* Asian Federation of Natural Language Processing.

Hassler, M., ja G. Fliedl. 2006. *Text Preparation through Extended Tokenization.* Data Mining VII: Data, Text / Web Mining / their Business Applications.

Hänninen, J., J. Juntti, P. Neittaanmäki, R. Kukkaniemi, E. Lehtomäki, R. Nyrhinen, T. Riipinen ja M. Savonen. 2018. *Tekoälypohjaisten teknologioiden testaus prototyyppisovelluksilla.* Informaatioteknologian tiedekunnan julkaisuja No. 68/2018, Jyväskylän Yliopisto.

Hopfield, J. 1982. *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Sciences of the United States of America vol. 79,8.

Hutchins, J. 2003. *Machine translation: half a century of research and use.* UNED summer school.

———. 2004. *Two precursors of machine translation: Artsrouni and Trojanskij.* International Journal of Translation 16(1).

Jiang, J., ja C. Zhai. 2007. *An Empirical Study of Tokenization Strategies for Biomedical Information Retrieval.* Information Retrieval 10(4-5).

Knuth, D. 2003. *Selected Papers on Computer Languages.* CSLI Lecture Notes, Stanford, California: Center for the Study of Language & Information.

Korenius, T., J. Laurikkala, K. Järvelin ja M. Juhola. 2004. *Stemming and Lemmatization in the Clustering of Finnish Text Documents.* Association for Computing Machinery.

Lee, J., J. Shin ja M. Realff. 2017. *Machine learning: Overview of the recent progresses and implications for the process systems engineering field.* Computers & Chemical Engineering Volume 114.

Lovins, J. 1968. *Development of a Stemming Algorithm.* Mechanical Translation / Computational Linguistics, vol.11, nos.1 / 2.

Lv, C., H. Liu ja Y. Dong. 2010. *An Efficient Corpus Based Part-of-Speech Tagging with GEP.* 2010 Sixth International Conference on Semantics, Knowledge / Grids.

Lyons, J. 1991. *Natural language and universal grammar.* Cambridge University Press.

Maind, S. B., ja P. Wankar. 2014. *Research Paper on Basic of Artificial Neural Network.* International Journal on Recent / Innovation Trends in Computing / Communication.

Mikolov, T., K. Chen, G. Corrado ja J. Dean. 2013. *Efficient Estimation of Word Representations in Vector Space.* Computing Reserach Repository - CORR.

Navigli, R. 2018. *Natural Language Understanding: Instructions for (Present and Future) Use.* International Joint Conferences on Artificial Intelligence Organization.

Nguyen, D. Q., D. Q. Nguyen, D. D. Pham ja S. B. Pham. 2016. *A Robust Transformation-Based Learning Approach Using Ripple Down Rules for Part-of-Speech Tagging.* Ai Communications 29(3).

Pepper Robot. 2020. Viitattu 17. maaliskuuta. https://pixabay.com/images/id-1695653/.

Perera, R., ja P. Nand. 2017. *Recent Advances in Natural Language Generation: A Survey and Classification of the Empirical Literature.* Computing / Informatics, Vol. 36.

Petrov, S., D. Das ja R. McDonald. 2012. *A Universal Part-of-Speech Tagset.* European Language Resources Association (ELRA).

Plisson, J., N. Lavrac ja D. Mladenic. 2004. *A Rule Based Approach to Word Lemmatization.* Proceedings of the 7th International multi-conference Information Society IS-2004, Ljubljana: Institut "Jožef Stefan".

Porter, M.F. 2006. *An algorithm for suffix stripping.* Program: electronic library / information systems, Vol. 40 No. 3, Emerald Group Publishing Limited.

Proisl, T., ja P. Uhrig. 2016. *SoMaJo: State-of-the-art tokenization for German web and social media texts.* Proceedings of the 10th Web as Corpus Workshop, Association for Computational Linguistics.

Rumelhart, D., G. Hinton ja R. Williams. 1986. *Learning representations by back-propagating errors.* Nature 323.

Rumelhart, D., G. Hintont ja R. Williams. 1986. *Learning from Delayed Rewards.* Viitattu 14. toukokuuta 2018. http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

Russell, S., ja P. Norvig. 2016. *Artificial intelligence: A modern approach (Global 3rd edition).* Pearson Education Limited.

Samuel, A. L. 1959. *Some Studies in Machine Learning Using the Game of Checkers.* IBM Journal of Research / Development, Vol. 3, No. 3.

Stenlund, S. 1990. *Language and Philosophical Problems.* Routledge, London & New York.

Sumathy, K., ja M. Chidambaram. 2013. *Text Mining: Concepts, Applications, Tools and Issues – An Overview*. International Journal of Computer Applications (0975 – 8887), Volume 80, No.4.

Toman, M., R. Tesar ja K. Jezek. 2006. *Influence of Word Normalization on Text Classification.* The 1st International Conference on Multidisciplinary Information Sciences & Technologies.

Turing, A. M. 2009. *Computing Machinery And Intelligence*. Parsing the Turing Test. Springer, Dordrecht.

Webster, J.J., ja C. Kit. 1992. *Tokenization As The Initial Phase In NLP*. 14th International Conference on Computational linguistics. Association for Computational Linguistics.

Liitteet

A Tutkimuksessa teknologioille opetetut entiteetit

maito	leipä	chili
mehu	jugurtti	curry
olut	muro	pitsa
energiajuoma	mysli	nugetti
vesi	hedelmä	lihapulla
limsa	kana	lihapiirakka
kahvi	kala	keitto
tee	sika	kalja
sipsi	nauta	hampurilainen
popkorni	ketsuppi	pesuaine
pähkinä	sinappi	saippua
keksi	suola	shampoo
karkki	pippuri	tiskiaine
jäätelö	öljy	deodorantti

B Tutkimuksessa käytetyt syötteet

Hampurilainen

Curry

Energiajuoma

Kahvi

Pitsa

Sipsi

Popkorni

Sinappi

Shampoo

Pesuaine

Tästä lauseesta tulisi löytyä sika.

Tästä lauseesta pitäisi puolestaan löytyä lihapulla.

Tässä onkin hieman pidempi syöte josta tulisi löytää sana maito. Jatketaan vielä hieman syötteen pituutta.

Tässä on toinen pitkä syöte, jonka ensimmäinen lause loppuu tähän. Tästä toisesta lauseesta tulisikin löytää sana ketsuppi.

Taas pidempi syöte, josta löytyy yhteensä kolme lausetta. Sijoitetaan etsittävä sana vasta viimeiseen lauseeseen. Ja nyt sitten se etsittävä sana on kala.

Ottaen huomioon oman tilanteensa, voi vain todeta, että kahvi noudattaa epäilemättä ideata-solla laajentuvia markkinoita ja uusia induktiivisia tuotantomenetelmiä.

Vaikka usein kuuleekin aivan järjettömiä mielipiteitä, on kuitenkin tosiasia, että öljy panostaa interaktiivisuuteen ajattelematta nykysukupolvia henkisesti rasittavaa elämäntyyliä.

Lisätään tähän tällainen täyte lause. Seuraava lause onkin generoitu puppugeneraattorilla. Vaikka osaltamme onkin saatettu syyllistyä laiminlyönteihin, voidaan taholtamme kuitenkin todeta, että keitto näyttelee keskeistä osaa pohdittaessa meihin kohdistuvaa tietotarvetta.

Taas pidempi syöte, josta löytyy yhteensä kolme lausetta. Sijoitetaan etsittävä sana vasta viimeiseen lauseeseen. Jotta emme unohtaisi yhteisön pyrkimyksiä, todettakoon, että nugetti pakottaa kohderyhmää huomioimaan liian vähäisiä tuotannollisia resursseja.

Olut, mehu, maito, limsa, kahvi, tee, sipsi.

Haluaisin ostaa seuraavia tuotteita: Pippuri, suola, kana, kala, mysli, muro ja jugurtti.

Hedelmä, leipä, tiskiaine

Pähkinä, tee, saippua

Pitsa ja hampurilainen

Lihapiirakka, lihapulla, öljy, tiskiaine

Energiajuoma, vesi, nauta

Kana, kala, sika, ketsuppi

Pesuaine, saippua, curry, chili

Keitto, jugurtti, leipä

Hampurilaisia

Currya

Energiajuomassa

Kahville

Pitsaksi

Sipsinä

Popkornilta

Sinapista

Shampoota

Pesuaineella

Tästä lauseesta tulisi löytää sikoja.

Tästä lauseesta pitäisi puolestaan löytää lihapullilta.

Tässä onkin hieman pidempi syöte josta tulisi löytää sana maidossa. Jatketaan vielä hieman syötteen pituutta.

Tässä on toinen pitkä syöte, jonka ensimmäinen lause loppuu tähän. Tästä toisesta lauseesta tulisikin löytää sana ketsupissa.

Taas pidempi syöte, josta löytyy yhteensä kolme lausetta. Sijoitetaan etsittävä sana vasta viimeiseen lauseeseen. Ja nyt sitten se etsittävä sana on kalaksi.

Ottaen huomioon oman tilanteensa, voi vain todeta, että kahvin noudattaa epäilemättä ideatasolla laajentuvia markkinoita ja uusia induktiivisia tuotantomenetelmiä.

Vaikka usein kuuleekin aivan järjettömiä mielipiteitä, on kuitenkin tosiasia, että öljyssä panostaa interaktiivisuuteen ajattelematta nykysukupolvia henkisesti rasittavaa elämäntyyliä.

Lisätään tähän tällainen täyte lause. Seuraava lause onkin generoitu puppugeneraattorilla. Vaikka osaltamme onkin saatettu syyllistyä laiminlyönteihin, voidaan taholtamme kuitenkin todeta, että keitotta näyttelee keskeistä osaa pohdittaessa meihin kohdistuvaa tietotarvetta.

Taas pidempi syöte, josta löytyy yhteensä kolme lausetta. Sijoitetaan etsittävä sana vasta viimeiseen lauseeseen. Jotta emme unohtaisi yhteisön pyrkimyksiä, todettakoon, että nugettina pakottaa kohderyhmää huomioimaan liian vähäisiä tuotannollisia resursseja.

Olutta, mehua, maitoa, limsaa, kahvia, teetä, sipsiä.

Haluaisin ostaa seuraavia tuotteita: Pippurin, suolan, kanan, kalan, myslin, muron ja jugurtin.

Hedelmiä, leipiä, tiskiaineita

Pähkinää, teetä, saippuaa

Pitsassa ja hampurilaisessa

Lihapiirakaksi, lihapullaksi, öljyksi, tiskiaineeksi

Energiajuomatta, vedettä, naudatta

Kanalle, kalalle, sialle, ketsupille

Pesuaineesta, saippuasta, currysta, chilistä

Keittoon, jugurttiin, leipään

C Rajapintatoteutus Wit.ai ja Watson Assistant -teknologioille

```
from wit import Wit
from watson_developer_cloud import AssistantV1

def wit_ai(text):
    client = Wit('O2ZBZCW5EQNL5SCQI5AC2RB73V72CRR5')
    response = client.message(text)
    return response['entities']



def watson_assistant(text):
    assistant = AssistantV1(
        version='2018-11-19',
        username='38c6d34b-4104-470d-ab52-5ef90b0ccc70',
        password='7LlB46HT2EGL',
        url='https://gateway.watsonplatform.net/assistant/api'
    )

    response = assistant.message(
        workspace_id='7369fe44-dba7-4043-8187-288fbafaf009',
        input={
            'text': text
        }
    ).get_result()
    return response['entities']
```

D Rajapintatoteutus Finnish-dep-parserille

```
def lemmatise(text):
    conn = http.client.HTTPConnection(FDP_SERVER)
    payload = text.encode("utf-8")
    conn.request("POST", "/", payload)

    res = conn.getresponse()
    data = res.read().decode("utf-8")
    res = [s.strip() for s in data.splitlines()]

    # Changes response from array into string
    lemma = []
    for idx, val in enumerate(res):
        row = val.split("\t")
        if (len(row) > 1):
            lemma.append(row[2])
    lemma_str = ' '.join(lemma)
    return lemma_str.replace("#", "")
```