Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Riku Sykäri

Dynaaminen muistiverkko keskustelu-botin kehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

27.11.2018

Tekijä Otsikko Sivumäärä Aika	Riku Sykäri Dynaaminen muistiverkko keskustelubotin kehityksessä 38 sivua + 2 liitettä 27.11.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	pelisovellukset
Ohjaajat	lehtori Miikka Mäki-Uuro tutkijatohtori Ella Peltonen

Insinöörityössä tutkittiin koneoppimisen alan mahdollisuuksia tuottaa terapian kontekstissa dialogiin pystyvä keskustelubotti, joka kykenee matkimaan kognitiivisessa terapiassa käyvää ihmisasiakasta mobiilipelissä. Työn tavoitteena oli löytää lupaava neuroverkkoarkkitehtuuri, joka kelpaisi jatkokehityksen pohjaksi insinöörityön jälkeen. Jatkokehityksen tuloksena toteutetulta keskustelubotilta edellytettäisiin kykyä tunnistaa ja käyttää tunnesanastoa, muistaa käyttäjän esittämiä kysymyksiä ja vastata kysymyksiin sopivilla kokonaisilla lauseilla.

Työ toteutettiin PyCharm-ohjelmointiympäristössä Python-kielellä. Koodin pohjaksi valittiin GitHub-verkkosivustolta vuonna 2016 aloitettu koodiprojekti dynaamisesta muistiverkosta, jota on siitä lähtien päivitetty kesään 2018 asti. Dynaamiset muistiverkot ovat vuoden 2015 syksyllä julkaistu neuroverkkoarkkitehtuuri, joka on tuottanut erinomaisia tuloksia visuaalisissa ja tekstipohjaisissa kysymyksiin vastaus -tehtävissä.

Neuroverkon kouluttamiseen käytettiin Facebook Research -instituutin kehittämässä bAbI-projektissa tuotettua tietojoukkoa, joka sisältää 20 erilaista loogisen päättelyn tehtävää, jotka testaavat koneoppimisalgoritmin yleistä pätevyyttä kysymyksiin vastaamisessa. Kooditoteutuksen vaihtamisen jälkeen verkon koulutus sujui odotetusti, pois lukien näytönohjaimen poikkeuksellisen hitaat koulutussuoritukset.

Dynaamiset neuroverkot olivat työn tekijälle uusi arkkitehtuuri, joten iso osa työstä kului toistuvien neuroverkkojen teoreettisten käsitteiden omaksumiseen. Koulutustuloksien perusteella verkon perustoiminta on varmistettu, ja jatkoa ajatellen verkon pätevyyttä voi testata dialogiin perustuvalla tietojoukolla, mikä Unity-pelimoottoriin kiinnittämisen yhteydessä auttaisi selvittämään mallin soveltuvuuden pelikäyttöön.

Avainsanat	tekoäly, koneoppiminen, peli, keskustelubotti, neuroverkko

Author Title Number of Pages Date	Riku Sykäri Dynamic memory network research for chatbot development 38 pages + 2 appendices 27 November 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Game Applications
Instructors	Miikka Mäki-Uuro, Senior Lecturer Ella Peltonen, Postdoctoral Researcher

The goal of the thesis was to study machine learning solutions that could be applied to create a suitable foundation for a chatbot capable of answering simple questions in the guise of a fictitious therapy client undergoing cognitive behavior therapy. To accomplish this the chatbot would require the ability to understand emotive language, maintain a consistent personality and be able to speak in complete sentences. The thesis contains a primer on chatbots, their historical significance as well as current trends within the industry.

The technical aspect of the thesis was performed within PyCharm IDE. The code used for the project is open source and can be downloaded from GitHub. The technique under consideration is called a Dynamic Memory Network, a type of algorithm first published in late 2015, notable for its high performance in question answering tasks.

The source code came with a dataset that is part of the bAbI project from Facebook Research and was used to train the network while documenting the results. Along the way the code implementation ended up being scrapped from one that used Theano library to a more recent one that was made with TensorFlow, in order to accommodate successful GPU training along with a CPU version for comparison.

Due to unfamiliarity with Dynamic Memory Networks and NLP techniques in general, a big chunk of the allotted time was spent studying the fundamentals of chatbots, recurrent neural networks and associated technologies. Promising avenues for continued development include testing the network's performance on a dialogue-based dataset, and if successful, working on integrating the model inside Unity game engine for use within a mobile game application.

Keywords	artificial intelligence, machine learning, game, chatbot

Sisällys

Lyhenteet ja käsitteet

ANN	Artificial Neural Network. Keinotekoinen neuroverkko, joka tarkoittaa aivojen inspiroimaa koneoppimisen algoritmia.
API	Application Programming Interface. Ohjelmointirajapinta, jonka avulla eri ohjelmat voivat kommunikoida keskenään.
DMN	Dynamic Memory Network. Neuroverkkoarkkitehtuuri, jossa syöte on jaettu tekstisekvenssiksi ja käyttäjän kysymyksiksi.
DNN	Deep Neural Network. Verkko, joka sisältää useita piilotettuja kerroksia.
DOS	Disk Operating System. IBM-yhteensopivien tietokoneiden käyttöjärjestelmäperhe.
Epookki	Kuvaa koko tietojoukon läpikäymistä koulutuksen aikana. Epookkien lukumäärällä kerrotaan, kuinka monta kertaa jokainen esimerkki näytetään koneoppimisalgoritmille.
GRU	Gated Recurrent Unit. Neuroverkoissa käytetty tekniikka, joka on kehitetty LSTM-verkkojen soluista.
LSTM	Long Short Term Memory Network. Variaatio toistuvista neuroverkoista.
NLP	Natural Language Processing. Tietojenkäsittelytieteen ala, joka koskee luonnollisen kielen käsittelyä ohjelmissa.
NPC	Non-Player Character. Tekoälyn ohjaama tietokonehenkilö.
RNN	Recurrent Neural Network. Toistuva neuroverkko, jota käytetään koneoppimisessa peräkkäisten syötteiden käsittelyyn.
Tietojoukko	Joukko koulutus- ja validointiesimerkkejä, joita koneoppimisalgoritmi käyttää mallin koulutukseen ja validointiin.

1 Johdanto

Insinöörityön tarkoituksena on tutkia koneoppimisen soveltuvuutta psykologian opetukseen keskittyvää viihdesovellusta varten. Työn tavoitteena on löytää sovelluksen vaatimuksia vastaava algoritmi ja siihen liittyvä tietojoukko, jolla algoritmia voidaan kouluttaa ja testata. Toinen päätavoite on myös kehittää tekijän ammattiosaamista koneoppimisen alalla, etenkin luonnollisen kielen prosessoinnin tekniikoissa.

Insinöörityön on tilannut pääkaupunkiseudun ohjelmistoalan startup YEEA Work Oy, joka tarjoutui asiakasyritykseksi tarkoituksena jatkokehittää psykoedukatiivista mobiilipeliä insinöörityön tuloksien perusteella. Jatkokehityksen tuloksena on tarkoitus toteuttaa keskustelubotti, joka kykenee vastaamaan käyttäjän esittämiin kysymyksiin käyttäen luonnollista puhekieltä. Keskustelubotin on tarkoitus pystyä esiintymään kuvitteellisena ihmishenkilönä kontrolloidussa tilanteessa, jossa puheenaiheet on rajoitettu botin omaan kuvitteelliseen historiaan.

Tutkimustyössä käytettiin PyCharm-ohjelmointiympäristöä ja Python-ohjelmointikieltä. Tarvittavat ohjelmointikirjastot Python-kielelle asennettiin Anaconda-nimisen avoimen lähdekoodin distribuution kautta. Käytetyn algoritmin implementaatio käyttää hyväksi TensorFlow- ja Keras-kirjastoja. Algoritmi on jaettu GitHub-verkkosivustolla.

Raportissa kuvataan työn eri vaiheet työharjoittelussa kehitetyn sovelluksen prototyypistä, sen jatkokehityksen tarpeista ja insinöörityön aiheen kehittymisestä, sopivien ratkaisujen etsinnästä sekä valitun ratkaisun kanssa työskentelystä, tuloksista ja johtopäätöksistä. Näiden kohtien lisäksi raportti sisältää työhön liittyvien koneoppimisen käsitteiden, keskustelubottien historian ja nykypäivän toteutuksien esittelyn.

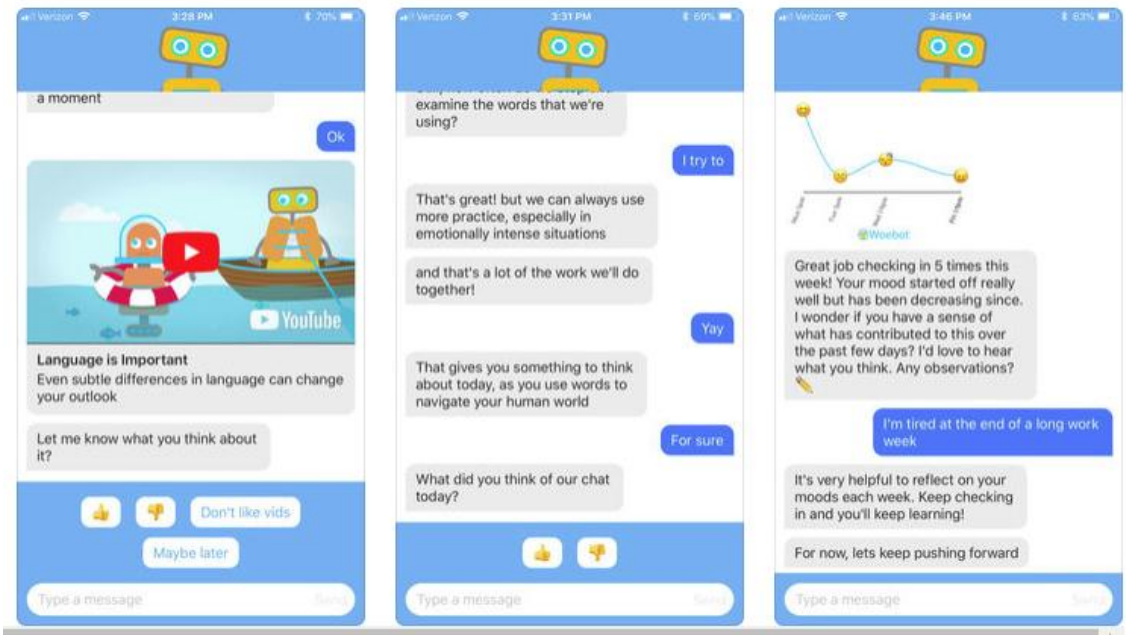
2 Keskustelubotti opetuspeliin

2.1 Insinöörityön tausta

Insinöörityön aihe syntyi vuoden 2018 alussa työharjoittelussa YEEA Work Oy:ssä, jossa työskentelin pelisuunnittelijan roolissa. Tehtävänä oli kehittää mobiilialustalle viihdyttävää opetuspeliä nimeltä Dr. de Fence, jonka pääpiirteenä olivat pelaajan ja tietokonehenkilöiden (NPC) väliset keskustelut ja niiden analysointi. Pelin teemana oli fiktiivisten asiakkaiden auttaminen pelaajan kontrolloiman terapeutin ja asiakkaan välisissä terapiakeskusteluissa. Keskustelujen sisältö ja rakenne saivat innoitusta kognitiiviseen käyttäytymisterapiaan perustuvista tekniikoista. Keskustelujen ja niihin liittyvien tehtävien perusteella pelaajat analysoivat fiktiivisten asiakkaiden ihmissuhdehaasteita ja niiden mahdollisia syitä. Tehtävänä oli auttaa purkamaan asiakashenkilöiden haitallisia suhtautumistapoja, joiden epäiltiin olevan syynä heidän ongelmiinsa.

Sovelluksen kohderyhmäksi valittiin alle 30-vuotiaat miehet, joilla on vaikeuksia työllistyä tai opiskella puutteellisten kommunikointi- ja itsesäätelytaitojen vuoksi. Pelin tavoite on, että kohderyhmän käyttäjät samaistuisivat pelin asiakashenkilöiden tarinoihin ja reaktioihin samalla, kun he etenevät keskusteluissa. Keskustelujen tarinat on tarkoituksellisesti kirjoitettu niin, että ne muistuttaisivat käyttäjien mahdollisia omia kokemuksia työ-, kaveri- tai perhesuhteissa. Tätä kautta pyritään kehittämään pelaajien itsetutkiskelutaitoja, mikä saattaisi toimia katalyyttinä uusien hyödyllisien käyttäytymismallien omaksumiseen.

Pelin pääinnoittajana toimi Facebook Messenger -pikaviestiohjelmassa käytettävä WoeBot-niminen keskustelubottisovellus, jonka kehittäjiin kuuluu kliinisen psykologian ja tekoälyn asiantuntijoita Stanfordin yliopistosta. WoeBot-sovelluksen tarkoitus on auttaa käyttäjiä, jotka mahdollisesti kärsivät masennuksesta ilman, että heillä on resursseja päästä terapiaan ammattitaitoisen ihmisen kanssa (1). Sovellus on ainakin toistaiseksi ilmainen, ja sitä voi kokeilla kuka tahansa, jos käytössä on Android- tai iOS-käyttöjärjestelmällä toimiva mobiililaite, jossa on verkkoyhteys. Keskustelun kieli on englanti (ks. kuva 1).



Kuva 1. Esimerkki WoeBotin keskustelusta ja käyttöliittymästä älypuhelimessa (1).

WoeBot-käyttäjät pääsevät viestimään keskustelubotin kanssa, joka kyselee järjestelmällisellä tavalla heidän lähiaikojen tapahtumistaan, haasteistaan ja tunnetiloistaan. Käyttäjät voivat vastata valmiiksi annetuilla vastausvaihtoehdoilla, jotka koostuvat lyhyistä tekstinpätkistä tai yksittäisistä hymiöistä, tai kirjoittaa vastauksen käyttäen omia sanojaan. WoeBot pyrkii säännöllisesti parantamaan käyttäjän mieltä positiivisella asenteella ja antaa käyttäjän vastauksien perusteella tilanteeseen sopivia kommentteja tai neuvoja (2). Avoimissa vastauksissa tämän mahdollistavat NLP-tekniikat, joihin palataan luvussa 3.1.

Toisin kuin WoeBot-sovelluksessa, jossa käyttäjää haastatellaan hänen kuulumisistaan ja tunnetiloistaan, Dr. de Fence -peli kääntää ihmisen ja tietokoneen roolit toisin päin. Sen sijaan, että käyttäjää pyydetään antamaan itsestään mahdollisesti arkaluontoista informaatiota ulkopuolisille, Dr. de Fence -pelissä pelaaja esittää kysymykset ja tietokonehenkilö antaa vastaukset. Tämä katsotaan kehitystiimissä positiiviseksi siltä kannalta, että sen pitäisi alentaa käyttäjän kokeilukynnystä vielä enemmän kuin WoeBot.

Toisena erona Dr. de Fence -pelissä painotettiin tarinallisuutta, jossa pelaaja pyritään syventämään fiktiiviseen rooliin pelin terapeuttina, joka on päähenkilö. Yksi jatkokehityksen tavoitteista oli rakentaa terapeutille laajempi narratiivi, joka linkittäisi yksittäisten

asiakkaiden tapahtumat yhteen jollakin mukaansatempaavalla tavalla. Tarinapohjaisuuden perusteena oli pyrkiä säilyttämään käyttäjän motivaatio peliin, vaikka hän ei kokisi tiettyjä yksittäisiä asiakashenkilöitä ja heidän keskustelujaan itselle mielenkiintoisena.

Insinöörityön tavoitteena oli löytää avoimesta lähdekoodista valmis toteutus, joka toimisi testipohjana asiakashenkilön keskustelubotin jatkokehitykselle. Syy valmiin toteutuksen hyötykäyttöön oli projektin suuri koko ja haaste, jonka takia puhtaalta pöydältä aloittaminen olisi vaatinut moninkertaisesti pidemmät tutkimus- ja kehitysvaiheet. Peliprojektin jatkokehitykseen tarvittiin tekoälyn malli, joka

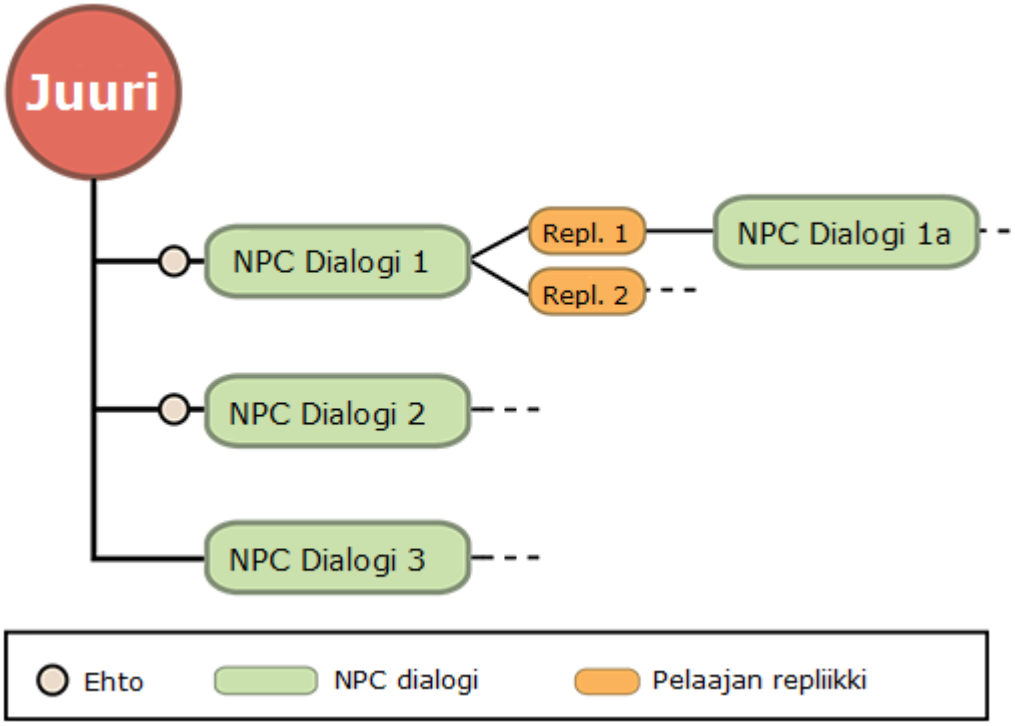
- ymmärtää pelaajan aiheeseen liittyviä vapaasti laadittuja kysymyksiä
- tuottaa kokonaisia lauseita vastauksina kysymyksiin
- kykenee ymmärtämään ja käyttämään tunnekielen sanastoa
- pystyy keskustelemaan kuvitteellisista tapahtumista
- ylläpitää yhtenäistä persoonaa keskustelun aikana
- muistaa aikaisemmat pelaajan kysymykset ja reagoi toistuviin kysymyksiin eri tavalla
- osaa päätellä, milloin pelaajan kysymys on tilanteeseen sopimaton ja laittaa viestin siitä eteenpäin.

Oma tavoite oli, että lopputyö valmistuisi suunnitellussa ajassa syksyn 2018 loppuun mennessä ja sen kautta kehittyisin koneoppimisen osaajana. Tarkoitus oli saada kattavampi kuva kielen käsittelyn tekniikoista neuroverkoissa sekä ymmärtää toistuvien neuroverkkojen toimintaperiaatteita paremmin.

2.2 Keskustelupuut

Työn alkuvaiheessa oli päätetty, että pelin keskustelut toteutetaan pelisovelluksiin tyypillisillä keskustelupuilla. Puut koostuivat pelaajalle ja tietokonehahmoille ennalta laaditusta tekstipohjaisista kysymyksistä, repliikeistä ja toteamuksista, jotka oli linkitetty toisiinsa puun oksien tapaan. Keskustelupuista tehtiin haarautuvia, millä tarkoitetaan sitä, että pelaajalle luotiin useampia kuin yksi vaihtoehtoinen repliikki keskustelun eri tilanteisiin.

Toinen pelissä toteutettu haarautumisen piirre oli erinäisten ehtojen käyttö, jolla kontrolloitiin tietokonehenkilön dialogin kulkua. Ehtojen täyttyminen saattoi liittyä pelaajan aikaisemmin valitsemiin repliikkeihin tai pelimaailman tapahtumiin keskustelun ulkopuolella. Nämä tekniikat tarjosivat pelaajalle mahdollisuuden edetä keskustelussa eri tavoilla riippuen pelin senhetkisestä tilasta ja siitä, mitä puhevaihtoehtoja pelaaja valitsee. Tämä voi myös johtaa erilaisiin lopputuloksiin riippuen puun toteutuksen logiikasta (ks. kuva 2).



Kuva 2. Kaavio haarautuvan dialogipuun alusta, sisältää myös ehtoja tietokonehenkilölle (3).

Keskustelupuiden yleinen suosio peleissä perustuu niiden yksinkertaiseen rakenteeseen. Lyhyiden keskustelujen luominen ja testaaminen on verrattain nopeaa kehitystyön alkuvaiheissa. Yksinkertaisuus helpottaa myös suunnittelijoiden ja käsikirjoittajien itsenäistä työskentelyä ilman, että he tarvitsisivat teknistä koulutusta aiheeseen. Haittapuolina ovat isokokoisten puiden rakenteen vaikea ymmärtäminen ja yleinen riski muutoksia tehdessä, sillä muutokset yhdessä osaa puuta voivat rikkoa toisen hyvin etäisen osan keskustelusta odottamattomasti, mikäli keskustelun etenemisen ehtotilanteita ei ole suunniteltu ja toteutettu huolellisesti. Muutokset ovat erittäin yleisiä pelien kehityksessä, ja niiden testaamiseen ja korjaamiseen kuluu merkittävästi aikaa. (4.)

Hyvin toteutetut keskustelupuut antavat pelaajille illuusion valinnanvapaudesta, mikä auttaa heitä eläytymään pelin keskusteluun. Yleensä pelaajan on kuitenkin helppo tiedostaa keskustelupuiden asettamat rajoitteet, sillä keskustelussa valittavia vaihtoehtoja on liian vähän tai niitä ei tule lisää pelin edetessä. Lisähaasteita asettavat pelit, joita on tarkoitus pelata useita pelikertoja.

Yksi ratkaisu on antaa pelaajille vapaus kirjoittaa mitä tahansa syötteeksi tietokonehenkilölle, jonka tehtävä on etsiä syötteestä olennaiset avainsanat. Tällöin valtaosa kaikista järkevistä (puhumattakaan teoriassa mahdollisista) syötteistä eivät tuota miellyttävää vastausta tietokoneelta, sillä käsikirjoittajan pitää huomioida jokainen mahdollinen kysymys erikseen ja luoda kaikille aiheen ulkopuolisille syötteille yleiset epäonnistuneita syöteitä vastaavat repliikit. Mikäli pelaaja on altis tekemään kirjoitusvirheitä tai hän ei ole perillä keskustelun tavoitteesta, pelin päähaasteeksi saattaa muotoutua pelaajan yritys sopeutua käyttöliittymän rajoituksiin eikä pelin sisäisen tehtävän ratkaisu. (4.)

Ajatus keskustelubottien käytöstä opetuspelin jatkokehityksessä nousi jo pelisuunnittelun alkuvaiheessa, kun selvisi, että pelistä on tarkoitus tehdä runsaasti uudelleenpelattava. Tavoitteeksi tuli, että peli on henkisen valmennuksen tapainen sovellus, johon käyttäjä palaa säännöllisesti pidemmällä aikavälillä hiomaan taitojaan. Sen sijaan, että peli keskittyisi tarjoamaan ainutlaatuisen pelikokemuksen ensimmäisellä pelikerralla, tarvittiin jokin pelimekaniikka, jonka viihdyttävyys heikentyisi paljon hitaammin. Keskustelupuut toimivat hyvin prototyyppivaiheessa, kun valtaosa kehitystyöstä kului pelin ydinmekaniikkojen suunnitteluun ja käyttöliittymän kehittämiseen, mutta niiden etu heikkenee keskustelujen monimutkaistuessa. Tämän lisäksi puut tarjoavat suhteessa vähän uuttaa pelattavaa verrattuna kehittämiseen ja testaamiseen kuluvaan aikaan.

Keskustelubotit eivät itsessään sisällä samoja rajoituksia, riippuen niihin käytettävistä tekniikoista. Ensimmäinen haaste uuden keskustelumuodon suunnitteluun oli lukuisten eri lähestymistapojen välillä valitseminen. Testaukseen valitusta tekniikasta kerrotaan tarkemmin luvussa 4.2.

3 Keskustelubottien tausta

3.1 Keskustelubottien alku

Ensimmäiset keskustelubotit kehitettiin 1960-luvulla, ja tunnetuin niistä on MIT Artificial Intelligence Laboratory -instituutissa professori Joseph Weizenbaumin kehittämä ELIZA-niminen siihen aikaan niin kutsuttu Chatterbot-ohjelma. Projektin idea nousi tietojenkäsittelytieteiden pioneerin Alan Turingin vuonna 1950 keksimästä ja hänen itsensä mukaan nimetystä Turingin testistä. Testin ideana on imitaatiopeli, jossa tekstipohjaisen terminaalin yhdessä päässä on ihminen, jonka tehtävänä on kysymyksiä kysymällä arvata, onko terminaalin toisessa päässä erilaista ihmistä esittävä imitoija oikeasti ihminen vai tietokoneohjelma. Mikäli kysyjä vastauksien perusteella toistuvasti epäonnistuu arvaamaan, onko toisessa päässä oleva imitoija oikeasti ihminen vai ei, tietokoneohjelman sanotaan läpäisseen Turingin testin. (5.)

Vuonna 1966 Weizenbaum kehitti ELIZA-ohjelman, jonka oli tarkoitus kyseenalaistaa Turingin testin merkityksellisyys näyttämällä, että uskottavan imitaation toteuttaminen ei ole lähellekään sama asia kuin älykkäiden tietokoneiden luominen. ELIZA simuloi keskustelua psykoterapeutin ja ihmispotilaan välillä käyttämällä ihmisen kirjoittamia tekstejä uusien repliikkien muotoiluun. Käytännössä tämä tarkoitti tiettyjen avainsanojen poimimista ihmisen antamista syötteistä ja niiden käyttämisestä uudelleen osana ohjelman vastausta. Mikäli syötteet eivät sisältäneet yhtäkään ohjelmaan valmiiksi tallennetuista avainsanoista, ELIZA vastasi geneerisellä tavalla riippumatta syötteen varsinaisesta sisällöstä. Weizenbaum järkyttyi saadessaan selville, että useat ohjelman käyttäjät, mukaan lukien hänen oma sihteerinsä olivat vakuuttuneita ELIZA-ohjelman kyvystä ajatella ja ymmärtää heidän huoliaan ja murheitaan. (6.)

Vuosikymmen myöhemmin, vuonna 1976 Weizenbaum julkaisi kirjan nimeltä Computer Power and Human Reason: From Judgment to Calculation, jossa hän argumentoi sitä oletusta vastaan, että tietokoneet pystyvät tekemään kaikkea, mihin ihminen pystyy, olettaen, että niissä on vain tarpeeksi prosessointikykyä ja nokkelaa ohjelmointia. Kirja nosti esiin kysymyksiä ja herätti keskustelua tekoälyn roolista yhteiskunnassa (7). Vuonna 1972 Stanfordin tutkija nimeltä Kenneth Colby kehitti toisen keskusteluohjelman nimeltä PARRY, jonka tarkoitus oli imitoida ihmistä, joka kärsii paranoidisesta skitsofreniasta.

Tammikuussa 1973 tietojenkäsittelytieteilijä Vint Cerf, jota kutsutaan usein yhdeksi internetin isäksi, yhdisti Internetin edeltäjän ARPANET-tietoverkon välityksellä ELIZA- ja PARRY-botit toisiinsa, jotta ne voisivat keskustella keskenään (8) (ks. kuva 3).

ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or a question mark.

PARRY: People get on my nerves sometimes.

ELIZA: I am not sure I understand you fully.

PARRY: You should pay more attention.

ELIZA: Suppose you should pay more attention.

PARRY: You're entitled to your own opinion.

ELIZA: What makes you think I am entitled to my own opinion?

PARRY: It bothers me just to be around people in general.

ELIZA: Please go on.

PARRY: I went to the track at Bay Meadows a while back.

ELIZA: What does that suggest to you?

PARRY: I went to the races.

ELIZA: Would you like to discuss your nerves?

PARRY: I guess so.

ELIZA: What does that suggest to you?

PARRY: What are you getting at?

ELIZA: What answer would please you most?

PARRY: It's not a matter or like or dislike.

ELIZA: What else would you like to discuss?

PARRY: I am getting tired of answering the same questions over and over.

ELIZA: What resemblance do you see?

PARRY: Why do you need to know that?

ELIZA: What do you think?

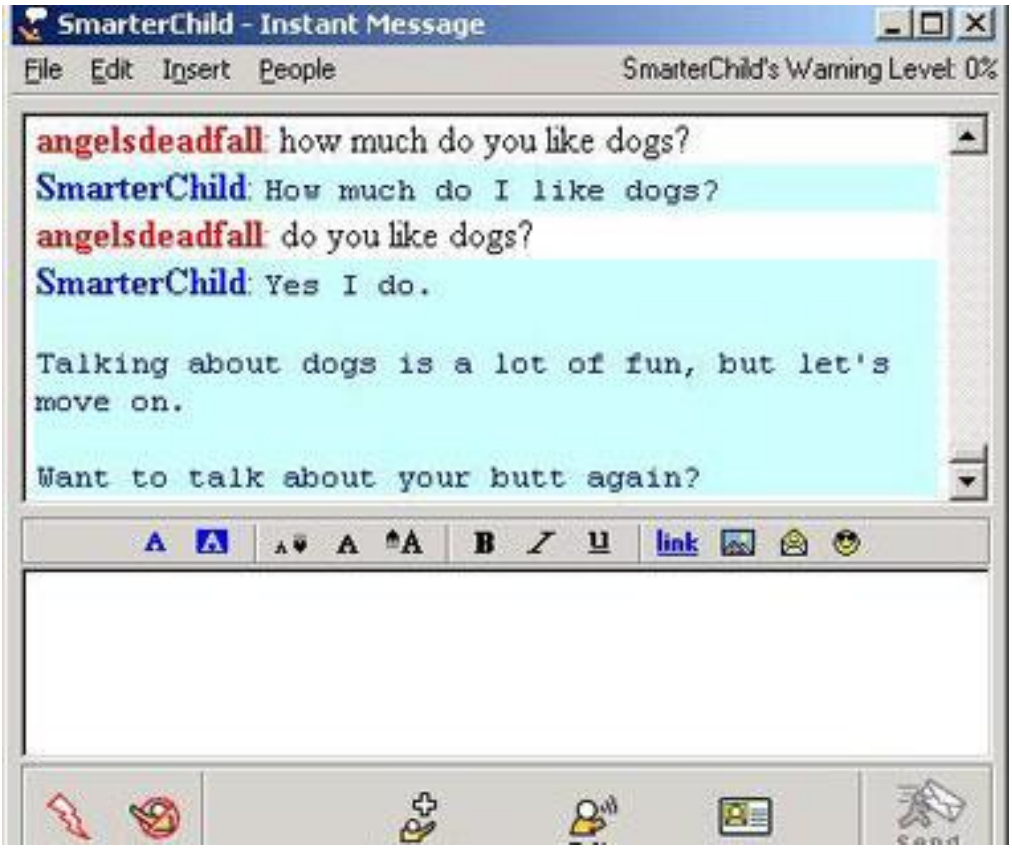
PARRY: I don't understand your motives.

ELIZA: Why don't you understand my motives?

Kuva 3. Otos ELIZA- ja PARRY-bottien keskustelulokista, alku vasemmalla ja osa lokien kolmannesta sivusta oikealla (8).

Kuvasta 3 näkee, että ELIZA ja PARRY eivät kumpikaan ole erityisen vahvoja imitaatioita ihmisistä, sillä niillä on tapana puhua toistensa ohi. ELIZA ehtii toistaa sanasta sanaan aikaisempia repliikkejään suhteellisen lyhyessä ajassa. Se seikka, miten 1960-luvun testikäyttäjät vakuuttuivat ELIZA-botin ihmisyydestä on mahdollisesti selitettävissä tietokoneiden senaikaisen uutuuden ja testaajien kokemattomuuden kautta.

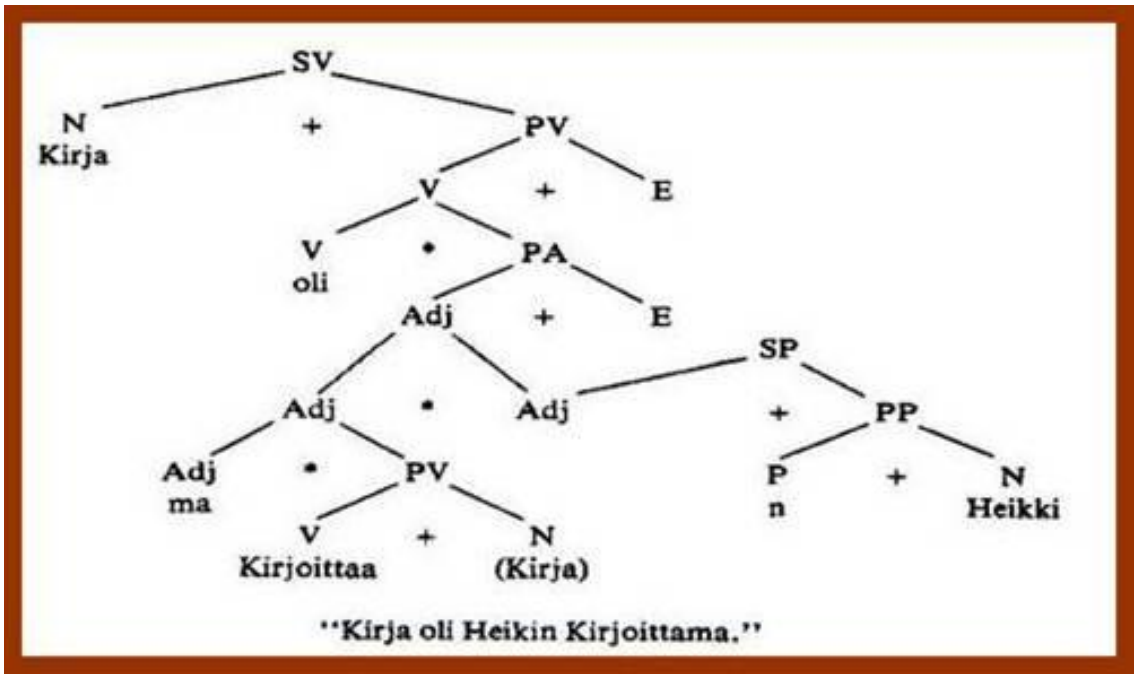
Bottien kehitys jatkui seuraavien vuosikymmenien aikana ja tuotti Jabberwacky-nimisen ohjelman vuonna 1988. Se oli suunniteltu matkimaan tavallista ihmisen puhetta. Vuonna 1992 kehitetty Dr. Sbaitso -botti oli psykologia esittävä ohjelma, joka toimi DOS-pohjaisissa tietokoneissa. Vuonna 1995 kehitettiin toinen kuuluisa keskustelubotti nimeltä A.L.I.C.E. (Artificial Linguistic Internet Computer Entity), jota kutsuttiin Alicebot- tai Alice-nimellä. Vuonna 2001 julkaistiin SmarterChild-niminen botti, joka toimi AOL Instant Messenger (AIM) -pikaviestiohjelman sisällä (ks. kuva 4). SmarterChild toimi sitä kautta Apple- ja Samsung-suuryritysten toteuttamien Siri- ja S Voice -nimisten henkilökohtaisen avustajien edelläkävijänä (9).



Kuva 4. SmarterChild-botin keskustelusta AIM-pikaviestintäohjelmassa (10).

Tässä luvussa mainitut toteutukset hyödyntävät kaikki tietojenkäsittelytieteiden alaa, jota kutsutaan nimellä Natural Language Processing (NLP) eli luonnollisen kielen käsittely. 1980-luvulle asti valtaosa NLP-tekniikoista perustui monimutkaisiin koosteisiin käsin kirjoitetuista säännöistä. Yksi mahdollinen esimerkki tällaisesta säännöstä on synonyymien yhdistäminen, kuten ’Hei’-, ’Heippa’- ja ’Moikka’-sanat, joille tehdään sääntö, että ne ovat yhtäläisiä ’Moi’-sanan kanssa. Tällaisen säännön omaava keskustelubotti selviää ihmiskäyttäjän tervehdystilanteessa ilman, että ihmisen täytyy käyttää vain ja ainoastaan ’Moi’-sanaa syötteeksi.

1980-luvun lopulla tietokoneiden prosessorien teknisen kehityksen ansiosta koneoppimisalgoritmeja alettiin soveltaa kielen käsittelyyn (11). Ensimmäiset esimerkit olivat päätöspuita, joiden avulla voitiin muun muassa erottaa ja selvittää puhekielen lauseesta lauseenjäsenten paikat ja yhteydet toisiinsa (ks. kuva 5).



Kuva 5. Esimerkki syntaktisesta puusta, johon on merkitty lauseenjäsenet (12).

Kaikkia eri tyypin tunnettuja koneoppimisalgoritmeja on liian monta tämän raportin sisällä käsiteltäväksi. Insinöörityölle olennainen alaluokka ovat keinotekoiset neuroverkot (ANN, Artificial Neural Network), tarkemmin rajattuna syvät neuroverkot (DNN, Deep Neural Network) ja toistuvat neuroverkot (RNN, Recurrent Neural Network), jotka yleistyivät kielen käsittelytehtävissä vasta 2010-luvulla. Viimeisen viiden vuoden aikana neuroverkoista on tullut huippuluokan ratkaisuja monien haasteiden ratkaisemiseen, mm. kuvan 5 lauseen ymmärtämiseen (13). Neuroverkkoihin perehdytään luvussa 4.

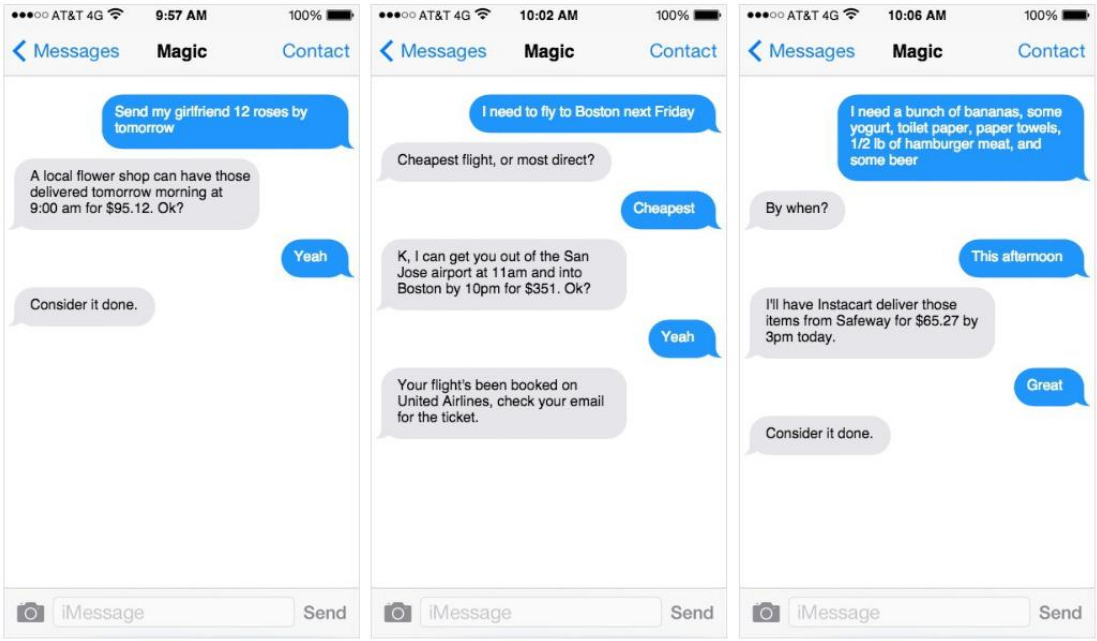
3.2 Keskustelubottien nykyajan kehityssuuntia

Nykyään keskustelubotteja rakennetaan lähes poikkeuksetta toimimaan pikaviestintäalustojen, kuten Facebook Messenger, Twitter, Skype ja Whatsapp, sisällä. Syitä tähän on useita, mutta selkein seikka on mobiilikäyttäjien kulutustapojen muuttuminen ajan kuluessa. Sen sijaan, että käyttäjät kokeilisivat säännöllisesti uusia ohjelmia, valtaosa viettää suurimman osan ajastaan älypuhelimilla muutaman ison ohjelman sisällä, jotka ovat tyypillisesti sosiaalisen median ja pikaviestinnän alustoja (14). Keskustelubotin kehittäjällä on valmiiksi valtava potentiaalinen käyttäjäkunta, mikäli hän tuo botin samaan sovellukseen, jota asiakkaat ovat jo valmiiksi tottuneet käyttämään. Toinen etu ovat

valmiiksi toteutetut ohjelmointirajapinnat (API), joita käyttämällä voidaan hakea tietoa ohjelman ulkopuolelta helpommin. Kolmas etu vakiintuneiden alustojen hyödyntämisessä on käyttäjien valmiiksi annettu hyväksyntä ohjelmien käyttöehdoissa. Facebook Messenger ja vastaavat alustat edellyttävät käyttöä varten suostumuksen siihen, että jokin kolmas taho voi lähettää käyttäjille viestejä.

Esimerkki elektronisen markkinointikeskustelubotin toiminnasta voi alkaa siitä, että jokin käyttäjä päättää ”peukuttaa” toisen ihmisen jakamaa viestiä, joka sisältää linkin, jonka aiheena on tarjous kotivakuutuksesta. Vakuutusyhtiöllä sattuu olemaan käytössä keskustelubotti, joka huomaa kaikki tämäntapaiset aktiviteetit uusilta tuntemattomilta käyttäjiltä. Tämän perusteella botti lähettää peukuttaneelle käyttäjälle henkilökohtaisella kanavalla viestin, jossa se aloittaa keskustelun kyselemällä käyttäjältä uuden vakuutussopimuksen tarpeesta. Tässä tapauksessa botti voi olettaa, että käyttäjä on kiinnostunut aiheesta, ja potentiaalisen asiakkaan ohjaaminen sopimuksen laatimisen suuntaan on paljon helpompaa kuin täysin satunnaista käyttäjää lähestyttäessä.

Samalla tapaa asiakaspalvelun rooliin erikoistunut botti voi automaattisesti aloittaa viestiketjun uuden käyttäjän kanssa, joka on saapunut yrityksen kotisivuille verkkoselain-sovelluksen kautta ja availee sivun välilehtiä sen oloisena, että vaikuttaisi etsivän vastausta tiettyyn kysymykseen. Asiakaspalvelun roolissa keskustelubotti huolehtii yleisesti kysytyistä kysymyksistä autonomisesti (ks. kuva 6) ja tarvittaessa siirtää asiakkaan keskustelun asiakaspalvelun ammattilaiselle, mikäli kyseessä on tavallisesta poikkeava ongelma.



Kuva 6. Asiakaspalveluun erikoistuneen botin keskustelun kulku Apple-yrityksen kehittämässä iMessage-pikaviestintäsovelluksessa (15).

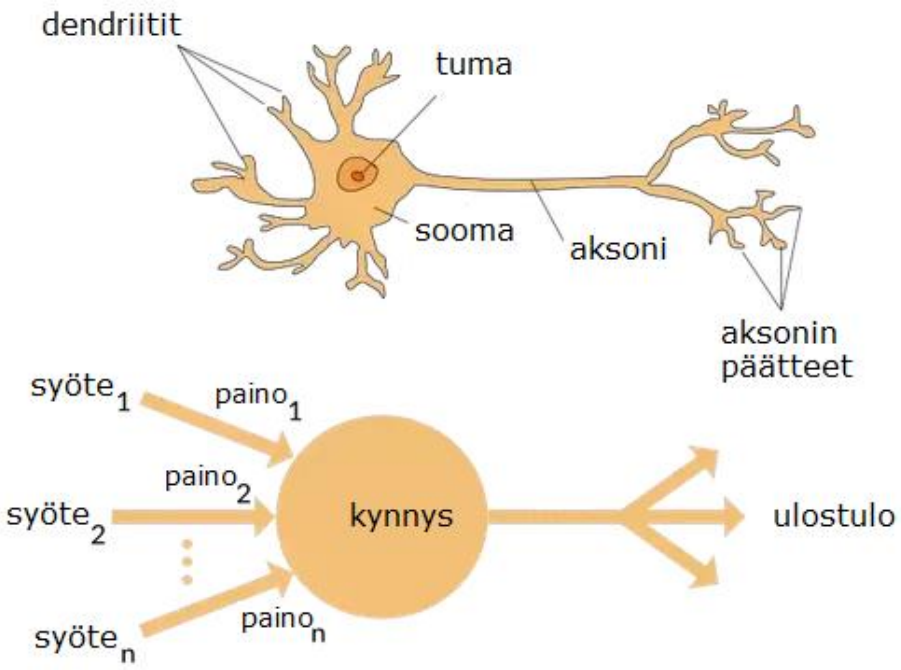
Eräs potentiaalisesti uhkaava tulevaisuuden trendi sosiaalisessa mediassa toimivilla keskusteluboteilla on niiden kyky imitoida ihmisiä ja sitä kautta vääristää käyttäjien mielikuvia yleisesti vallitsevasta mielipiteestä, mahdollisesti johonkin kiistanalaiseen kysymykseen liittyen. Viime aikojen tunnetuimpia esimerkkejä oli erinäisten Twitter-sovelluksessa toimivien bottien vaikutus Yhdysvaltojen vuoden 2016 presidentinvaaleihin (16). Koska keskustelubotit ovat autonomisia, pystyvät postittamaan viestejä satoja kertoja enemmän kuin ihmiskäyttäjät samassa ajassa ja bottien toteutuksia on suhteellisen yksinkertaista kopioida, mikä tahansa taho, jolla on resursseja hankkia pilvipalveluiden avulla ylläpito bottiarmeijalle, pystyy levittämään näkemyksiään näennäisesti tuhansien ihmisten äänellä. Tulevaisuudessa, jossa tietokoneiden ja keskustelubottien kehitys mahdollistaa entistä lukuisampia ja uskottavampia imitaatioita, tämä uhka saattaa kasvaa entisestään.

4 Neuroverkot ja tutkittava malli

4.1 Neuroverkkojen peruskäsitteet

Neuroverkkoihin perustuvan mallin käyttäminen vaikutti insinöörityössä itsestäänselvyydeltä johtuen alan tämänhetkisestä menestyksestä koneoppimisen alalla sekä tekijän omista kokemuksista verkkojen kehittämisestä eri projekteissa. Tässä luvussa käydään läpi neuroverkkojen toimintaan liittyviä käsitteitä ja lyhyt kuvaus tutkitusta dynaamisesta muistiverkosta.

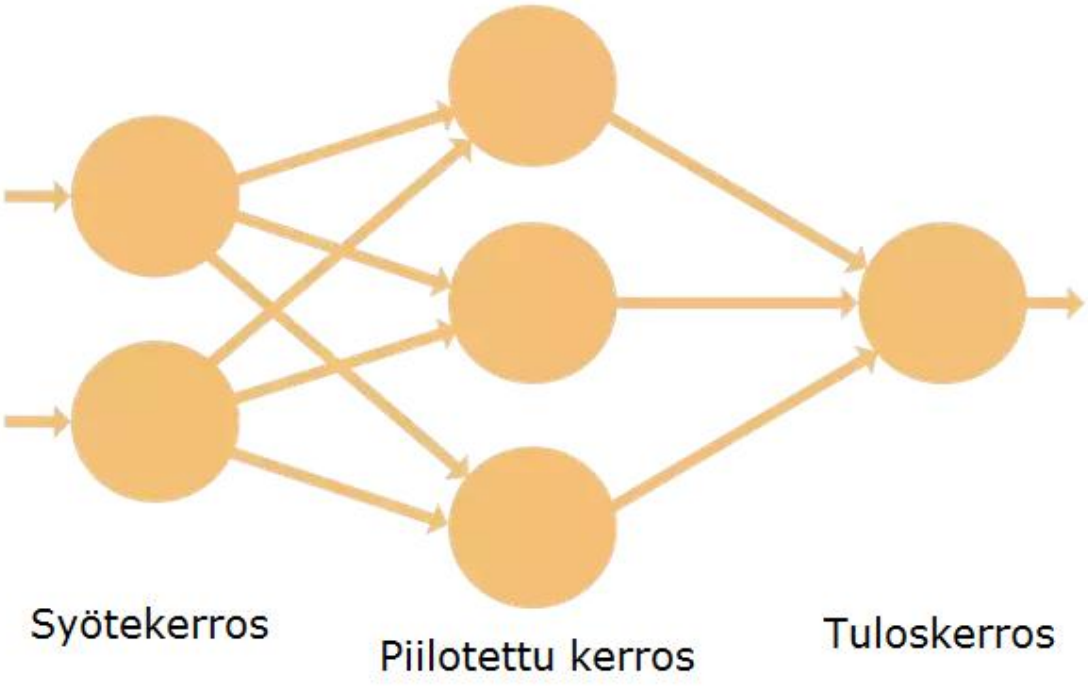
Neuroverkot koostuvat keinotekoisista neuroneista, jotka on kehitetty ottamalla mallia ihmisten ja eläinten aivoista. Keinotekoinen neuroni on matemaattinen abstraktio biologisen neuronin toiminnasta (ks. kuva 7). Sen toiminta koostuu siihen tulevista syötteistä, syötteisiin liittyvistä painoarvoista, neuronin sisäisestä esiasetuksesta tai kynnysarvosta, neuronin käyttämästä aktivaatiofunktiosta, joka on esimerkiksi ReLU-tyyppiä (Rectified Linear Unit), sekä syötteiden, painoarvojen ja kynnysarvon perusteella lasketun aktivaatiofunktion tuloksesta. Nämä arvot ovat yleensä reaalilukujen muodossa. Yksittäinen neuroni kykenee suorittamaan yksinkertaisia loogisia operaatioita esimerkiksi AND- tai OR-logiikkapiirin tavoin, riippuen neuronin sisäisestä kynnysarvosta ja tulevien syötteiden painoarvoista. (17.)



Kuva 7. Biologisen ja keinotekoisen neuronin väliset yhteydet (18).

Neuroverkon nimi tulee sarjaan ja/tai rinnakkain toisiinsa kytketyistä neuroneista, joissa tieto kulkee syötekerroksesta piilotettujen kerroksien läpi tuloskerrokseen (ks. kuva 8). Esimerkiksi kuvantunnistusta varten toteutetussa verkossa, jonka tehtävä on luokitella sille näytetyt kuvat kissa- ja koirakuviin, syötekerroksen neuronit ottavat vastaan yksittäisen kissa- tai koirakuvan pikseleiden pikseliarvot, jotka työnnetään eteenpäin piilotetuille kerroksille. Piilotetuilta kerroksilta kulkee tieto kerroksen neuronien painoarvojen perusteella eteenpäin seuraavalle kerrokselle, ja siitä seuraavalle, ja niin edelleen. Tätä prosessia kutsutaan eteenpäin välittämiseksi (feedforward). (17.)

Viimeisen piilotetun kerroksen tulos ohjautuu tuloskerrokseen, joka toimii luokittelijana lajitellen kuvan kissaksi tai koiraksi sen perusteella, minkälainen ulostulo tuloskerroksesta tulee, esimerkiksi 0,5 tai sitä korkeampi numeroarvo vastaa kissaa ja alle 0,5 vastaa koiraa. Luokittelun tulos on tyypillisesti todennäköisyysjakauma, jossa verkko antaa jokaiselle mahdolliselle vastausluokalle oman subjektiivisen todennäköisyytensä 0:n ja 1:n väliltä. Kaikkien luokkien yhteiset todennäköisyydet summautuvat luvuksi 1. (17.)



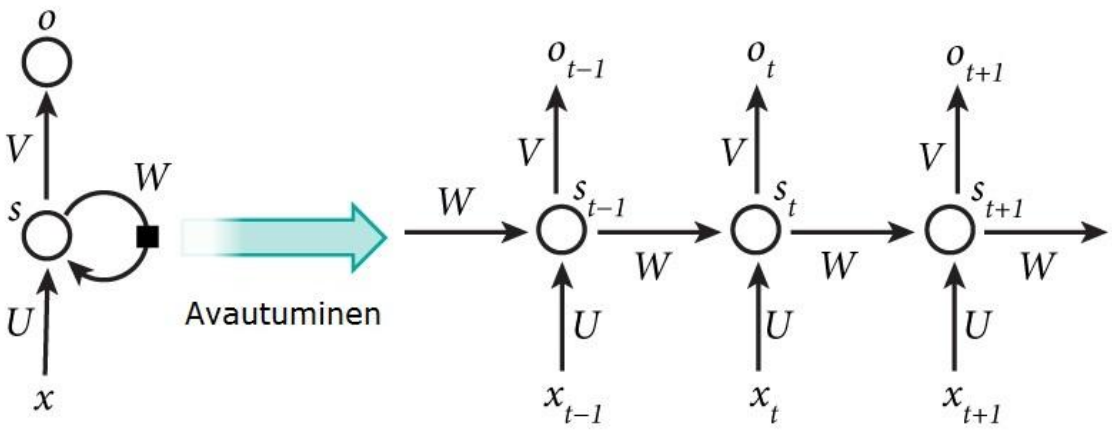
Kuva 8. Eri kerrokset verkossa, jossa on vain yksi piilotettu kerros ja yksi tuloskerroksen neuroni. Tällaista verkkoa voisi käyttää binääriseen luokitteluun, kuten kissa vai koira. (18.)

Jotta neuroverkot pystyvät oppimaan koulutusesimerkkien kautta, pitää verkon laskemaa tulosta pystyä vertaamaan niin kutsuttuun leimattuun tai oikeaan tulokseen. Leimatun ja lasketun tuloksen eroa kuvataan hävikillä, ja hävikkifunktiosta lasketaan gradientti, josta verkko näkee, mitä neuronien painoarvoja pitää muuttaa ja kuinka paljon. Gradientin avulla verkko pystyy muuttamaan taaksepäin etenemällä kunkin piilotetun kerroksen painoarvoja siten, että uusi laskettu vastaus on asteen lähempänä haluttua vastausta. Tätä prosessia kutsutaan taaksepäin levittämiseksi (backpropagation), jonka käyttö keksittiin neuroverkkojen yhteydessä 1980-luvulla (19). Eteenpäin välittämistä, hävikin laskentaa ja taaksepäin levittämistä iteroidaan vuorollaan kaikilla koulutusesimerkeillä, kunnes verkon suoritustarkkuuteen ollaan tyytyväisiä.

Luvussa 2 määriteltiin tavoitteet jatkokehityksen mallille täytettäväksi, mikä pitkälti rajasi lupaavat ratkaisut syvien neuroverkkojen (DNN) alle luettaviin toistuviin neuroverkkoihin (RNN). Perinteisissä neuroverkoissa tehdään implisiittinen oletus, että kaikki syötteet (ja ulostulot) ovat nopan heittojen tapaan toisistaan riippumattomia. Useissa tehtävissä tämä on kuitenkin erittäin huono lähtökohta. Jos tavoitteena on esimerkiksi pystyä

ennustamaan tekstisyötteessä seuraava sana, on välttämätöntä, että verkko muistaa sitä edeltävät sanat (20).

Idea toistuvista neuroverkoista kehitettiin käsittelemään syötteitä, jotka sisältävät informaatiota peräkkäisessä muodossa, kuten videonpätkät, jotka koostuvat peräkkäisistä kuvista tai tekstit, jotka koostuvat peräkkäisistä sanoista. Toistuvat neuroverkot juontavat nimensä siitä, että ne tekevät samat operaatiot jokaiselle jonon peräkkäiselle elementille, jolloin tulos on riippuvainen kaikista aikaisemmista laskutoimituksista (ks. kuva 9). Toinen tapa nähdä asia on kuvitella, että toistuvilla neuroverkoilla on muisti, johon tallentuu informaatiota aikaisemmista aika-askeleista (20).

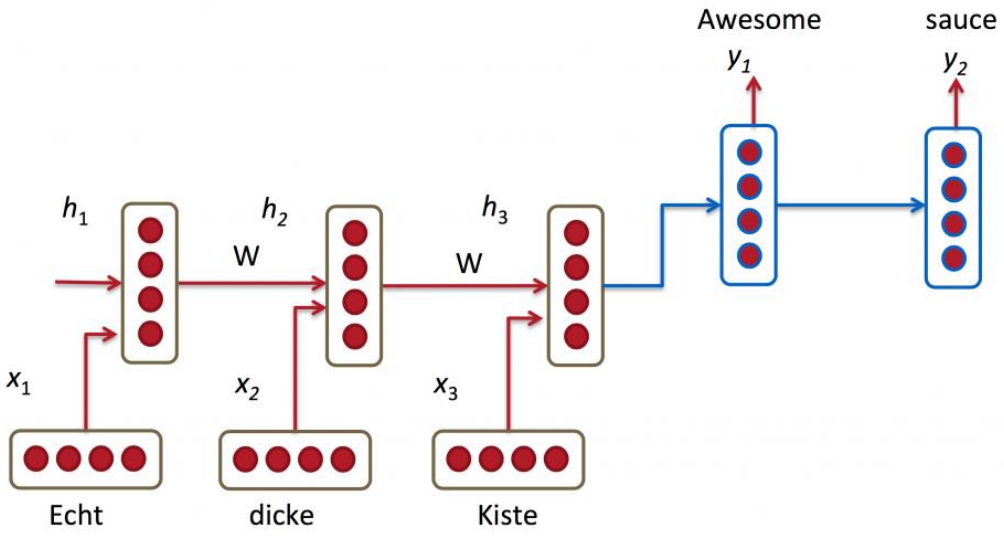


Kuva 9. Tyypillisen toistuvan neuroverkon rakenne. Aika-askeleen t laskutoimitus riippuu viime aika-askeleessa t miinus 1 lasketusta arvosta. (20.)

Toisin kuin perinteisessä verkossa, jossa jokainen kerros käyttää eri parametrejä, toistuva neuroverkko jakaa samat parametrit (U, V ja W) kaikkien askelten välillä. Ainoa muuttuva tekijä on kunkin aika-askeleen syöte.

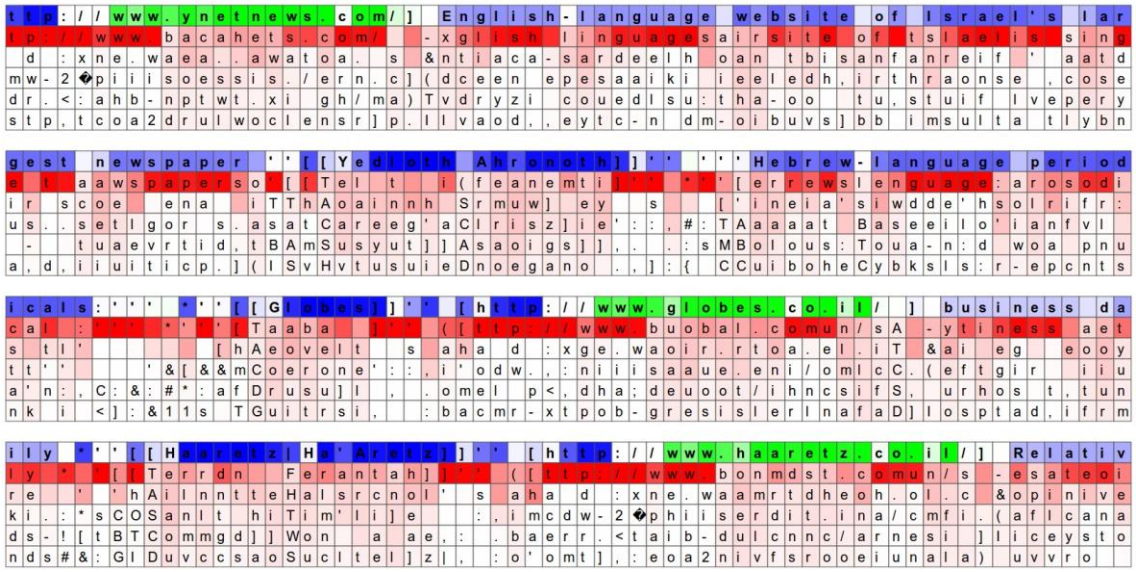
Kuvassa 10 toistuva neuroverkko avautuu kokonaiseksi verkoksi. Verkon kerroksien lukumäärä vastaa syötteen ja ulostulon tekstin sanaelementtien määrää. Riippuen tehtävästä verkon ei tarvitse ottaa jokaisella aika-askeleella uutta syötettä tai tulostaa uutta ulostuloa. Jos verkon tehtävänä on esimerkiksi antaa lauseen syötteeseen yksisanainen vastaus, tulos voidaan tulostaa ainoastaan viimeisellä aika-askeleella (20). Yksi tyypillisimpiä käyttökohteita toistuville neuroverkoille on kielenkääntäjä, joka ottaa syötteeksi

esimerkiksi englanninkielisen lauseen ja tuottaa tulokseksi espanjankielisen käännöksen. Toinen yleinen käyttötarkoitus on puhutun kielen kääntäminen tekstimuotoon.



Kuva 10. Esimerkki verkosta, jossa tuloste tehdään vasta koko syötteen nähtyä. Verkko kääntää saksaa englanniksi heikolla lopputuloksella. (20.)

Perinteisten toistuvien neuroverkkojen heikkoutena on pitkän välin aika-askeleiden yhteyksistä oppiminen, jos syötteet ovat yksittäisten lauseiden sijasta kokonaisten sivujen pituisia tekstejä. Tätä ongelmaa varten on kehitetty muun muassa LSTM-verkko (Long Short Term Memory Network), jossa piilotettu tila (h) lasketaan eri tavalla. LSTM-verkoissa muistia kutsutaan soluiksi, jotka sisältävät portteja. Portit kontrolloivat sitä, mitkä asiayhteydet soluun tallennetaan, mikä tarjoaa solulle kyvyn muistaa paljon etäisempiä syötteitä kuin perinteinen RNN-tyypin verkko (20). Yksittäiset LSTM-solut voivat myös erikoistua havaitsemaan tietyn tyyppisiä elementtejä, esimerkiksi merkkijonoja, jotka muistuttavat verkkosivun osoitteita. Kuvassa 11 on visualisoitu neljä esimerkkiä eri neuronien aktivaatioista.

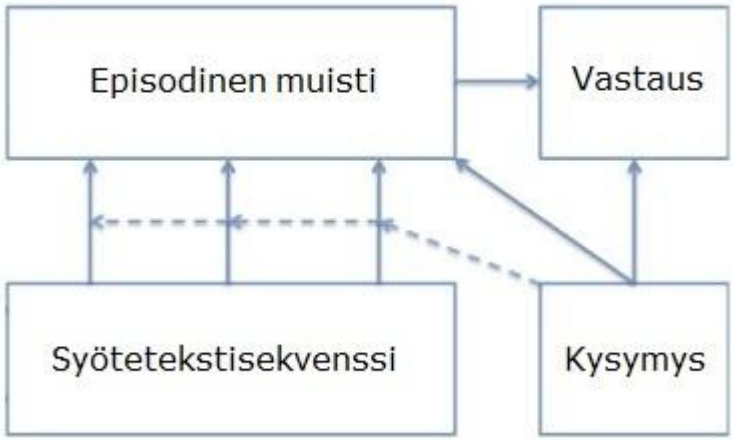


Kuva 11. Neljä eri neuronia lukevat merkkijonoja. Vihreät kohdat kuvaavat korkeaa aktivaatiota, siniset puolestaan matalaa. (21.)

On hyvä muistaa, että kokonaisten neuroverkkojen toiminta on harvoin yhtä helposti tulkittavaa kuin kuvan 11 verkkosivuista kiinnostuneiden yksittäisten neuronien aktivaatiot. Hyvin usein suunnittelijan intuitio verkon käyttäytymisestä on varsin erilainen verrattuna verkon oikeasti oppimaan ratkaisuun. Tästä syystä monet tutkijat ovat toteuttaneet kuvan 11 tapaisia visualisointeja verkkojen toiminnan ymmärtämistä ja testaamista varten.

4.2 Dynaaminen muistiverkko

Dynaaminen muistiverkko (DMN, Dynamic Memory Network) on neuroverkkoarkkitehtuuri, johon törmäsin osana tutkimustyötä kesän 2018 loppupuolella. DMN kehitettiin alun perin visuaalisten ja tekstillisten kysymyksien vastaamiseen. Kyseessä on vuonna 2015 julkaistu arkkitehtuuri, jonka isoin muutos toistuviin neuroverkkoihin on syötteen jakaminen kahteen eri osaan syötetekstisekvenssiksi ja yksittäiseksi kysymykseksi (ks. kuva 12). Tämän lisäksi syöte ohjataan niin kutsuttuun episodiseen muistiin, joka koostuu Gated Recurrent Unit (GRU) -soluista, jotka ovat LSTM-solujen variaatio (22).

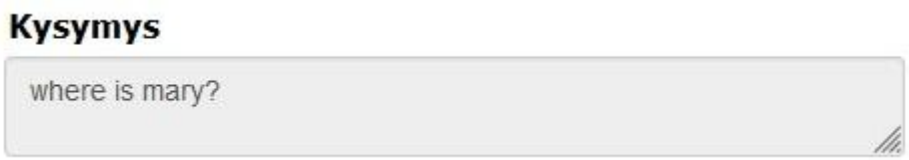


Kuva 12. DMN-arkkitehtuuri (22).

Syötetekstisekvenssin idea on, että sinne varastoidaan pitkä yhtäjaksoinen syöte, joka voi olla esimerkiksi pelin terapia-asiakasta esittävän keskustelubotin tapauksessa asiakkaan taustatarina, josta pelaaja voi esittää kysymyksiä käyttämällä kysymyssyötettä (ks. kuva 13). Syötetekstisekvenssin heikkous pelin näkökulmasta on, että kaikki vastaukset ovat periaatteessa avonaisia kekseliäälle pelaajalle keskustelun alusta alkaen. Mikäli peliin haluaisi jonkintapaista porrastusta, sille pitäisi kehittää erillinen järjestelmä, joka mahdollisesti lataa syötetekstisekvenssiin ajon aikana uusia taustatarinan pätkiä sitä mukaa, kun sopivat avainsanat esiintyvät pelaajan esittämissä kysymyksissä tai asiakashenkilön tulostamissa vastauksissa.

Kuva 13. Esimerkki syötesekvenssiin laitetavasta ”tarinasta”. Kuva on otettu eri muistiverkkototeutuksen mukana tulevasta verkkoselainohjelmasta. (23.)

Syötetekstisekvenssin ohella tarvitaan kysymys, jotta verkko alkaa toimia. Kysymyksen sanojen pitää liittyä jollakin tavalla syötetekstisekvenssin tarinan sanoihin (ks. kuva 14), jotta episodinen muisti pystyy etsimään oikean vastauksen.



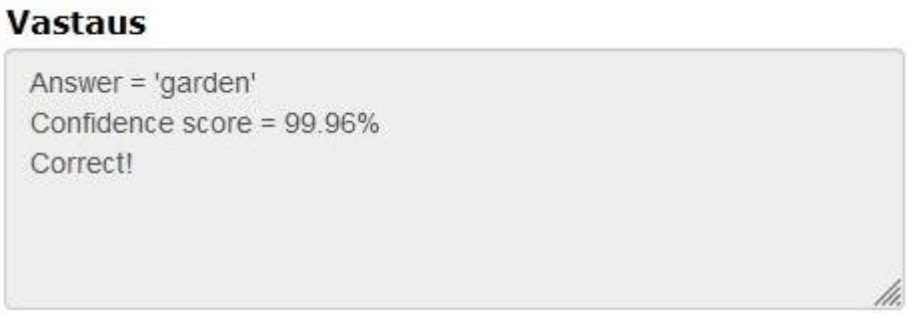
Kuva 14. Esimerkki yksinkertaisesta kysymyksestä (23).

Episodisen muistin tehtävä on tällöin antaa taustatarinaan pohjaten kysymykseen oikea vastaus (ks. kuva 15). Verkko alkaa järjestelmällisesti lukea syötetekstisekvenssiä läpi ensimmäisestä rivistä viimeiseen, etsien tarinasta kysymyksen sanoja vastaavia kohtia ja tallentaen niihin liittyvät lauseet jatkoa varten. Jossakin kohtaa verkko toivon mukaan tunnistaa olennaisen lauseen, joka on avain oikeaan vastaukseen. Suoriutuakseen vastauksen etsinnästä muistiverkolla pitää olla sopiva sanasto valmiiksi istutettuna, ennen kuin sitä yritetään käyttää.

Episodinen muisti	Mem 1	Mem 2	Mem 3
sandra travelled to the bedroom.	0.00	0.00	0.00
sandra journeyed to the garden.	0.00	0.00	0.00
sandra moved to the hallway.	0.00	0.00	0.00
sandra travelled to the bathroom.	0.00	0.00	0.00
mary went to the bathroom.	0.08	0.02	0.00
john moved to the office.	0.00	0.00	0.00
sandra went back to the office.	0.00	0.00	0.00
mary travelled to the hallway.	0.32	0.10	0.01
mary went to the garden.	0.59	0.88	0.98
daniel moved to the office .	0.00	0.00	0.00

Kuva 15. Episodinen muisti löysi tarinasta merkitsevän lauseen, joka vastasi kysymykseen (23).

Vastauksien hyvä puoli on siinä, että verkko tarjoaa sisäisen itsevarmuuden vastauksen pätevyydestä kysymykseen (ks. kuva 16). Tämä voi pelin sisällä tarkoittaa sitä, että kun verkko on liian epävarma antamastaan vastauksesta, kyseinen vastaus voidaan ohittaa ja vaihtaa yleiseen repliikkiin ymmärryksen puutteesta. Tällä tavoin pelaajaa saa palautetta esittämiensä kysymyksien pätevyydestä automaattisesti.



Kuva 16. Oikea vastaus. Verkko löysi olennaisen lauseen ja tiesi oikean vastaussanan kyseisestä lauseesta. (23.)

Testattavaksi valittu dynaamisen muistiverkon lähdekoodi löytyi GitHub-verkkosivustolta, joka tarjoaa paikan Git-versionhallinnan verkkosivustoa käyttäville ohjelmakehitysprojekteille. Projektissa on varsinaisen tekijän lisäksi neljä avustajaa ja sen viimeisin päivitys on tehty heinäkuussa 2018 (24). Projekti on toteutettu Google Brain -tutkimustiimin kehittämällä TensorFlow-kirjastolla, joka on avointa lähdekoodia. Verkkototeutus on jaossa MIT-lisenssillä, joten sitä voi soveltaa vapaasti kaupallisiinkin tuotteisiin. Syy alkuperäisen Theano-kirjastolla toteutetun verkon (23) hylkäämiseen oli vaikeus saada Theano-toteutusta toimimaan näytönohjainta hyödyntävien kirjastojen kanssa.

5 Tietojoukko ja metriikat

5.1 Valittu tietojoukko

GitHub-verkkosivulta ladattavan projektin mukana tulee Facebook Research -instituutin bAbI-projektin 20:een eri tehtävään jaettu kysymys ja vastaus (QA) -tietojoukko (ks. kuvat 17 ja 18). Tietojoukko sisältää 1 000 koulutusesimerkkiä ja saman määrän validointiesimerkkejä jokaista tehtävätyyppiä kohti. Koulutus- ja validointiesimerkit ovat valmiiksi verkkoon yhteensopivassa muodossa koulutuksen käynnistystä varten, eli tiedon esiprosessoinnin pystyi ohittamaan kokonaan.

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 4: Two Argument Relations
The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Task 5: Three Argument Relations
Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill

Task 6: Yes/No Questions
John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

Task 7: Counting
Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

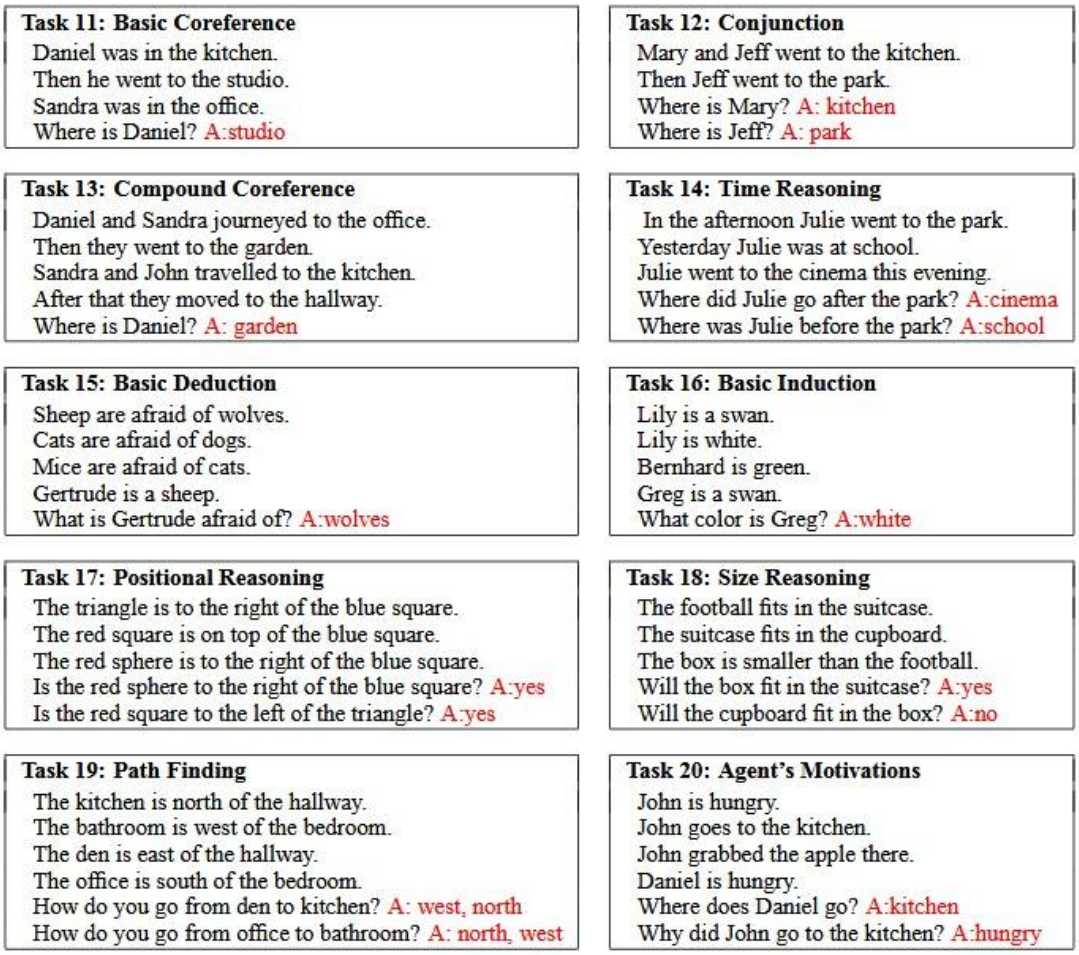
Task 8: Lists/Sets
Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
John took the apple.
What is Daniel holding? milk, football

Task 9: Simple Negation
Sandra travelled to the office.
Fred is no longer in the office.
Is Fred in the office? A:no
Is Sandra in the office? A:yes

Task 10: Indefinite Knowledge
John is either in the classroom or the playground.
Sandra is in the garden.
Is John in the classroom? A:maybe
Is John in the office? A:no

Kuva 17. Esimerkit tehtävien 1–10 koulutusesimerkeistä (25).

Ensi silmäyksellä mikään kuvan 17 tehtävistä ei vaikuttanut suoraan pelin tavoitteita ajatellen tärkeältä, mutta osa saattaisi olla osittain hyödyllisiä, kuten tehtävät 9 ja 10.



Kuva 18. Esimerkit tehtävistä 11–20 (25).

Loputkin bAbI-tehtävät vaikuttavat ratkovan erilaisia ongelmia, kuin mitä pelin keskustelubotilta edellytettäisiin, paitsi tehtävä 20, joka käsittelee henkilöiden motivaatiota. Yleinen huolenaihe on koulutusesimerkkien lyhyys, sillä varsinaisen terapia-asiakkaan tarinan tulisi luultavasti olla vähintään satoja tai tuhansia sanoja pitkä, jotta pelaajalle riittää kyseltävää. Tätä seikkaa korostaa se, että neuroverkolta puuttuu ihmisen elämänkokemuksien kautta maailmaa koskeva opittu taustatieto, joka mahdollistaa luetun ymmärtämisen lyhyelläkin tekstisyötteellä, jos tekstin konteksti muistuttaa elämänkokemuksia. Muistiverkolle tarinan käsikirjoittaja joutuisi todennäköisesti kirjoittamaan asiakkaan tarinatekstin auki paljon perusteellisemmin.

Facebook Research -sivuilta löytyy myös kuusiosainen dialogitietojoukko, joka olisi lähempänä projektin tarpeita (26). Ainoa este tietojoukon hyödyntämiseen oli insinöörityön aikataulun kiristyminen ja dialogitietojoukon yhteensopivaksi tekemiseen potentiaalisesti kuluva aika. Muiden sopivien ja kattavien tietojoukkojen etsiminen ei tuottanut tulosta, suurelta osin projektin tavallisesta poikkeavien tavoitteiden takia. Yksi harkinnassa ollut vaihtoehto oli kokeilla tuottaa kehitystiimin sisällä sopiva tietojoukko tai ulkoistaa tehtävä asiakasyrityksen kontakteille. Tässä esteeksi tulivat tiimin rajalliset resurssit, sillä hyödyllinen tietojoukko vaatisi tyypillisesti satoja, ellei tuhansia koulutusesimerkkejä, jotta lopputulos olisi työn arvoinen.

5.2 Mallin arvioinnin metriikat

Lähdekoodi toteuttaa neuroverkkojen malleille tyypillisten metriikoiden, kuten hävikin ja tarkkuuden, keräämisen automaattisesti. Jokainen tehtävätyyppi vaatii oman erillisen koulutuksen, ja niitä arvioidaan erillisinä suorituksina. Päätin lisätä valmiiksi kerättyjen metriikoiden joukkoon käytettyjen epookkien määrän ja epookkeihin yhteensä käytetyn koulutusajan (ks. liite 1). Ajoajat on listattu, jotta voidaan verrata mm. näytönohjaimen tuella ajettavan version tehtävien suoritusnopeutta suhteessa oletusarvoiseen pelkkää keskusprosessoria käyttävään versioon.

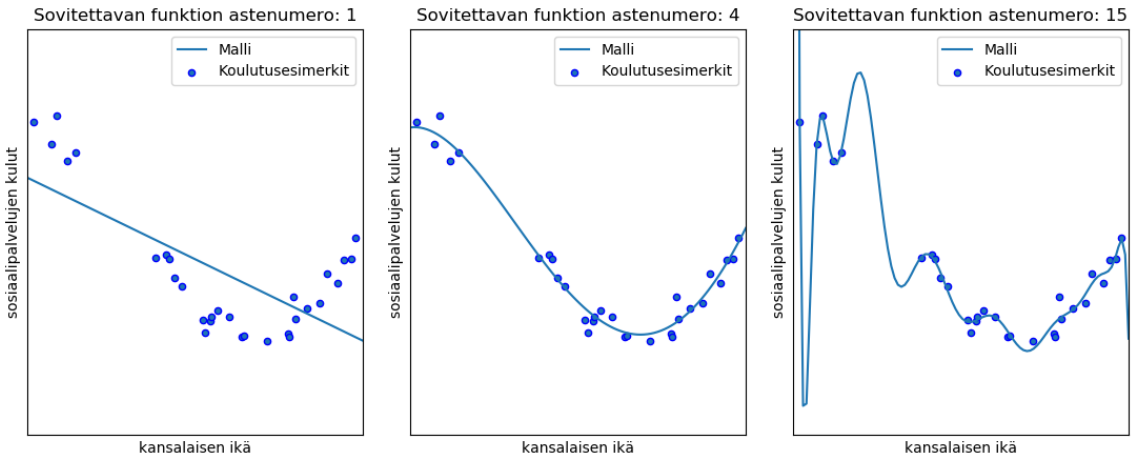
Epookilla tarkoitetaan koko tietojoukon läpikäymistä koulutuksessa ja sen lukumäärä kertoo, kuinka monta kertaa tietojoukko käydään läpi ennen koulutuksen loppumista. Tavallisesti mitä useamman epookin koulutus käydään läpi, sitä isompi riski syntyy mallin ylisovittamisesta (overfitting). Kulutettu epookkien määrä ei yleensä ole erityisen mielenkiintoinen metriikka, mutta tutkiessani lähdekoodia huomasin, että verkon koulutuksessa toteutetaan niin kutsuttu ennenaikainen pysäytys, mikäli tarpeeksi monta epookkia käydään läpi ilman, että mallin validointitarkkuus paranee aikaisemmasta parhaasta tuloksesta. Tämän seikan takia halusin tallentaa sen, kuinka monta epookkia kului siihen, että malli ei enää onnistunut parantamaan validointitarkkuutta.

Tarkkuus kertoo, kuinka iso osuus verkon arvauksista osuu oikeaan vaihtoehtoon. Hävikkiä käytetään kertomaan neuroverkolle, kuinka paljon koulutusesimerkin laskettu vastaus eroaa oikeasta vastauksesta. Mitä suurempi hävikki on kyseessä, sitä enemmän vastaus on vialla. Hävikin laskemiseen käytetään hävikkifunktiota, joka on tässä

tapauksessa niin kutsuttu Sparse Softmax Cross Entropy (27). Hävikki voidaan lasketa useammasta koulutusesimerkistä tai koko koulutuksen tietojoukosta yhteen, tosin siinä tapauksessa käytetään kustannus-termiä.

Kustannusfunktio sisältää kaikkien esimerkkien hävikin summan lisäksi yleensä vakinaistamisen termin. Vakinaistamisella (regularization) tarkoitetaan mallin rankaisemista sen monimutkaisuudesta. Mallin monimutkaisuuden rankaiseminen on yleensä hyvä idea, koska se osaltaan ennaltaehkäisee mallin ylisovittamista. Ylisovittaminen puolestaan tarkoittaa verkon pyrkimystä oppia koulutusesimerkit ”liian hyvin”, minkä voi mieltää niin, että ohjelma alkaa opetella ulkoa kyseisen koulutusjoukon yksittäiset esimerkit sen hinnalla, että oppisi yleisen säännön, joka pätee tutkittavaan ilmiöön.

Hyvä tapa havainnollistaa ylisovittamista on kuvitella tilanne, jossa halutaan opettaa koneoppimisalgoritmille sääntö, joka ennustaa kansalaisten vuotuiset käytettyjen sosiaalipalvelujen kulut suhteessa heidän ikäänsä. Ohjelmalle annetaan tietojoukko, jossa on 30 kansalaisen ikä- ja kulutiedot. Ohjelman tehtävä on kehittää funktio eli sääntö, joka kuvaa ilmiötä mahdollisimman tarkasti, minkä perusteella voidaan ennustaa loppuväestön kulut henkilötunnuksien syntymäpäiväosan perusteella. Kuvassa 19 näkyy mallin kehitys vasemmalta oikealle, koulutuksen alusta loppuun.



Kuva 19. Ohjelma sovittaa mallin käyrää näkemiinsä koulutusesimerkkeihin, vasemmalta oikealle (28).

Paras ennustaja tulee käyttämällä keskimmäistä mallia, ennen kuin ohjelma alkaa ylisovittaa funktiota koulutusesimerkkeihin. Ylisovittaminen on todennäköisempää, jos

koulutusesimerkkejä on vain pieni määrä tai jos ne ovat keskenään hyvin samankaltaisia, esimerkiksi samasta ikäluokasta tässä tapauksessa.

Ylisovittamisen ilmiö on syy tietojoukkojen jakamiseen erillisiin koulutus- ja validointiosiin. Ohjelmalle ei koulutusvaiheessa näytetä validointijoukon esimerkkejä niistä oppimista varten, vaan kyseessä on säännöllisin väliajoin mallin testaaminen sille uusilla ja tuntemattomilla esimerkeillä. Kustannusfunktioon lasketaan tästä syystä ainoastaan koulutusjoukon esimerkit. Jos ylisovittamista tapahtuu, koulutustarkkuus tyypillisesti kasvaa kohti maksimia, vaikka validointitarkkuus laskee, koska malli ei enää osaa yleistää oppimaansa uuteen tietoon, kuten validointijoukon esimerkkeihin.

6 Verkon tulokset ja johtopäätökset

6.1 Verkon koulutus

Koulutus suoritettiin pöytäkoneella, Windows 7 (SP1) -käyttöjärjestelmässä. Keskusmuistia koneessa oli 16 GB. Keskusprosessorin malli oli Intel Core i5 3570K, kellon taajuudella 3,40 GHz. Näytönohjain oli NVIDIA GeForce GTX 960, jossa oli 4096 MB videomuistia ja kellon taajuus 1,28 GHz. NVIDIA CUDA -ohjelmointirajapinnasta oli asennettu 9.0-versio. Koulutus sisälsi jokaisen bAbI-tehtävän koulutuksen sekä keskusprosessorilla että näytönohjaimella, joiden tuloksia on verrattu lähdekoodin tekijän sekä muistiverkon kehittäneiden tutkijoiden tuloksiin (ks. taulukot 1, 2 ja kuva 20).

Koulutusta kontrolloivat verkon hyperparametrit, kuten oppimisvauhti, L2-vakinaistamisen arvo, pudotusarvo ja nippukoko, on jätetty lähdekoodin oletusarvoilleen. Kokeilin alkuvaiheessa muutaman testin kohdalla kasvattaa oppimisvauhtia 0,001:stä 0,002:een koulutusajan säästämiseksi, mutta se johti nopeasti heikompaan tulokseen, koska malli ei pystynyt lähentymään kohti oikeaa vastausta. Kokeilin myös laskea pudotusarvon 0,9:stä 0,8:aan, mutta se johti mallin ylisovittamiseen useassa eri tehtävässä. Epookkien määrä oli puolitettu 200:sta 100:aan, ja epookkien ennenaikainen keskeytysarvo oli puolitettu 20:stä 10:een (ks. liite 2).

Taulukko 1. Tehtäväkohtaiset koulutustulokset keskusprosessorilla suoritettuna. Valtaosa tehtävistä keskeytettiin ennenaikaisesti, sillä malli oli saavuttanut maksimitarkkuuden tai oppiminen ei onnistunut tuottamaan tulosta.

bAbI-tehtävä	Käytetyt epookit	Koulutusaika	Minimihävikki	Pienin virhe (%)
1	76	11 min 35 s	1,110	0
2	53	34 min 4 s	14,166	3,2
3	22	37 min 55 s	94,363	30,8
4	99	5 min 48 s	0,891	0
5	36	42 min 9 s	2,795	0,1
6	100	26 min 22 s	1,242	0
7	23	11 min 19 s	14,088	3,2
8	95	41 min 40 s	1,633	0
9	100	12 min 23 s	1,306	0
10	49	6 min 34 s	1,736	0
11	100	15 min 24 s	1,119	0
12	86	12 min 18 s	1,105	0

13	100	15 min 12 s	1,226	0
14	49	7 min 55 s	10,914	2,4
15	61	7 min 50 s	1,311	0
16	22	2 min 27 s	86,735	46,8
17	88	4 min 42 s	14,771	3,5
18	58	12 min 7 s	1,248	0
19	73	6 min 56 s	3,610	0,4
20	100	18 min 28 s	0,683	0

Tulokset olivat keskusprosessorin osalta odotuksien mukaiset, aavistuksen heikommat hävikin ja virheprosentin suhteen verrattuna lähdekoodin ja tutkijoiden tuloksiin, mikä selittyy mahdollisesti matalammalla maksimiepookkien arvolla ja sillä, että suurin osa tehtävistä tuli ajettua vain kerran.

Taulukko 2. Näytönohjaimen tulokset. Huomion arvoista on koulutusaikojen merkittävä pidentyminen keskusprosessorin tuloksiin verrattuna.

bAbI-tehtävä	Käytetyt epookit	Koulutusaika	Minimihävikki	Pienin virhe (%)
1	100	23 min 43 s	0,981	0
2	71	1 h 7 min 41 s	3,206	0,1
3	25	1 h 2 min 5 s	80,239	23,8
4	44	4 min 19 s	1,162	0
5	22	35 min 49 s	2,684	0
6	100	38 min 34 s	1,168	0
7	22	15 min 55 s	15,233	3,5
8	100	1 h 6 min 28 s	1,469	0
9	100	22 min 43 s	1,254	0
10	52	12 min 0 s	1,684	0
11	100	24 min 57 s	1,080	0
12	100	24 min 35 s	1,011	0
13	100	24 min 40 s	1,195	0
14	49	14 min 16 s	5,010	1,0
15	100	21 min 14 s	1,148	0
16	14	2 min 42 s	90,522	51,1
17	62	5 min 45 s	9,550	2,6
18	15	5 min 7 s	12,877	9,4
19	56	8 min 54 s	2,203	0,1
20	100	29 min 42 s	0,688	0

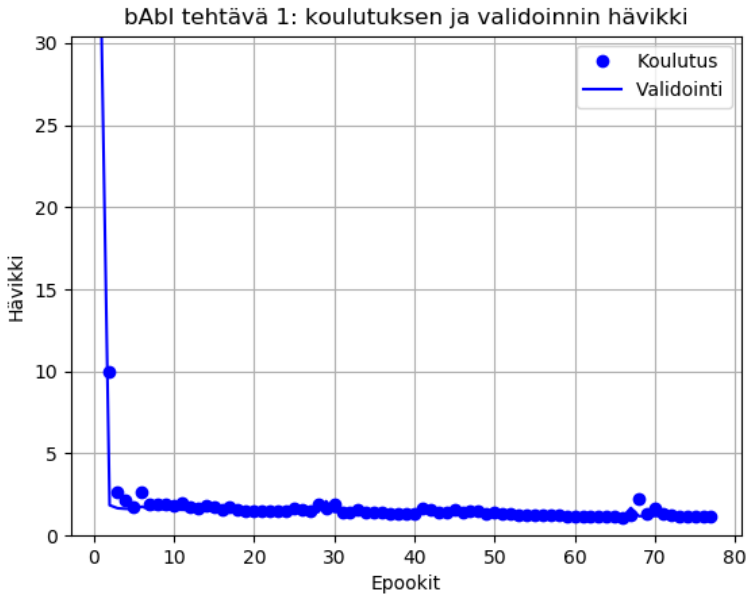
Hävikki ja virheprosentti olivat sekä CPU- että GPU-ajoissa hyvin lähellä toisiaan, mikä ei yllättänyt ollenkaan. Enemmän odotin, että näytönohjain olisi suoriutunut tehtävistä

merkittävästi keskusprosessoria nopeammin. Sen sijaan näytönohjain oli 1,5–2 kertaa hitaampi suurimmassa osassa tehtäviä.

Task ID	TensorFlow DMN+	Xiong et al DMN+
2	0.9	0.3
3	18.4	1.1
5	0.5	0.5
7	2.8	2.4
8	0.5	0.0
9	0.1	0.0
14	0.0	0.2
16	46.2	45.3
17	5.0	4.2
18	2.2	2.1

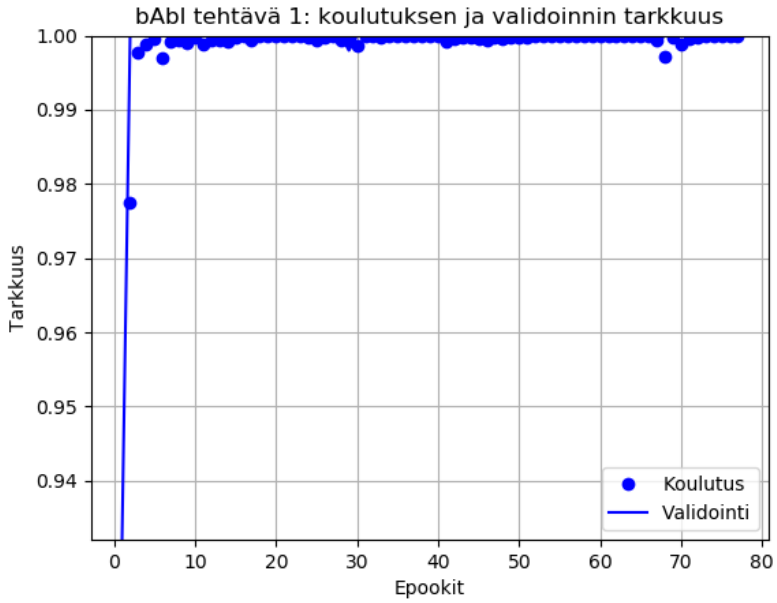
Kuva 20. Lähdekoodin tekijän ja tutkijoiden saavuttamat virhearvot. Taulukosta puuttuvat tehtävät ovat 100 % tarkkuudella suoritettuja (24).

Yksittäisten tehtävien koulutukset näyttivät hävikin ja tarkkuuden kehityksen osalta graafisesti esitettynä kuvien 21–28 mukaisilta. Ensimmäinen bAbI-tehtävä oli suhteessa hyvin helppo.



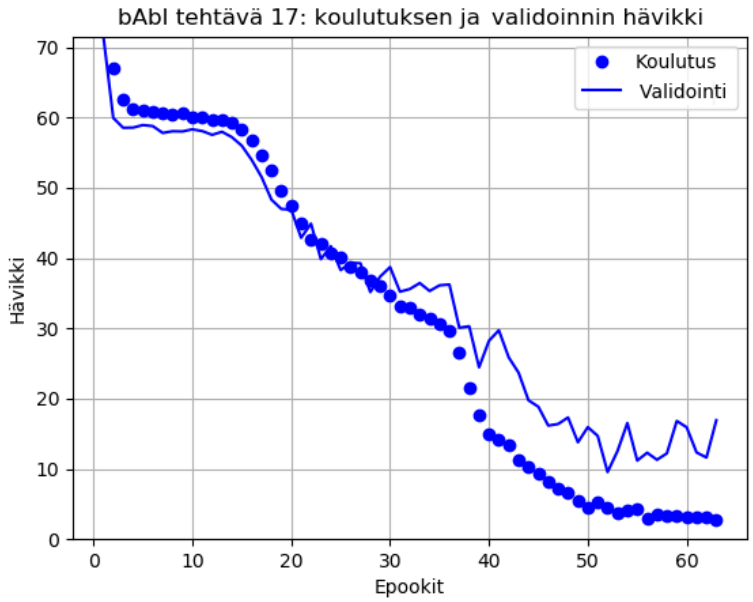
Kuva 21. bAbI-tehtävä 1 on esimerkki helposta tehtävästä. Hävikki laskee hyvin nopeasti.

Hävikin laskiessa nopeasti on loogista, että tarkkuus kasvaa samaan. Kyseessä on tehtävä, jossa saavutettiin 100 %:n tarkkuus nopeasti.



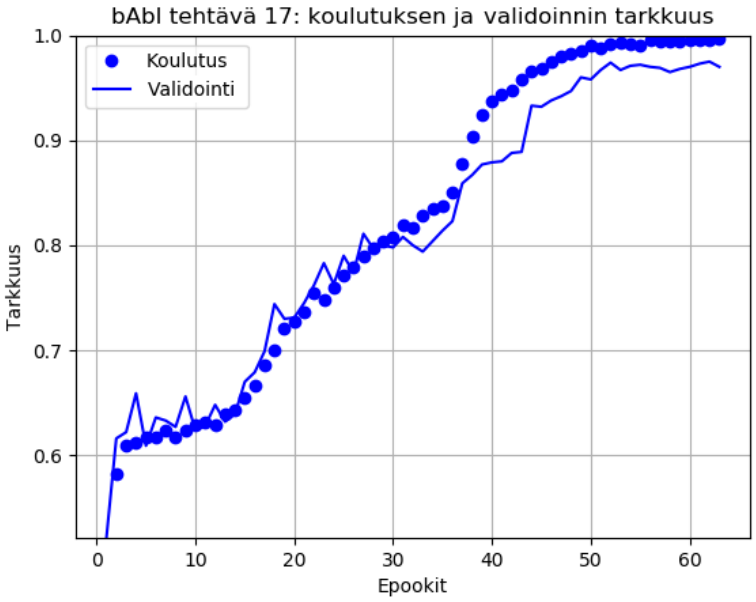
Kuva 22. Helpossa tehtävässä tarkkuus nousee maksimiin muutaman epookin sisällä.

Esimerkki keskiverrosta tehtävästä: bAbI-tehtävä 17.



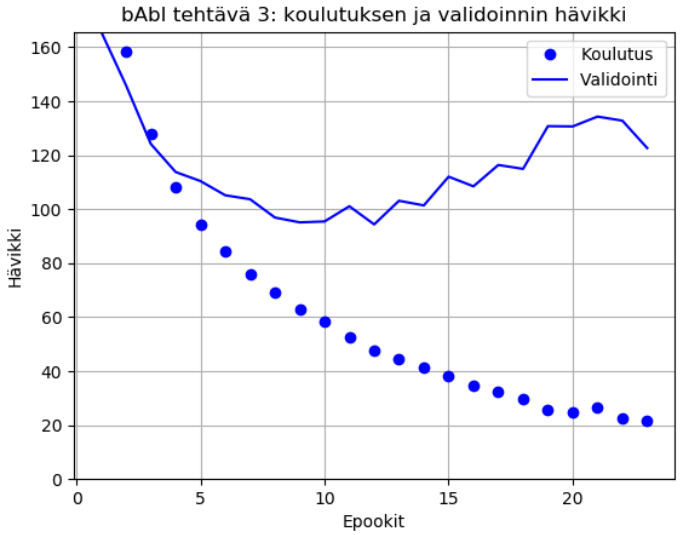
Kuva 23. Hävikki laskee hitaasti mutta tasaisesti.

Tehtävä 17 sattui olemaan ajoajan puolesta yksi nopeimpia suorituksia.



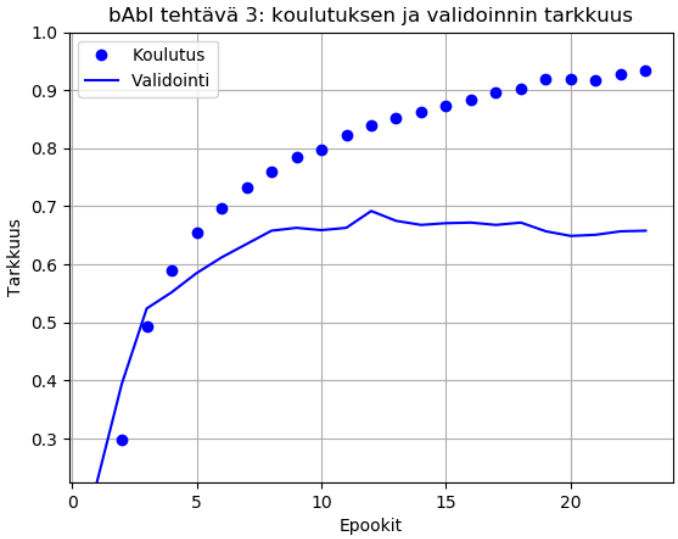
Kuva 24. Tarkkuus nousee hitaasti mutta tasaisesti.

Tehtävä 3 oli päinvastoin yksi hitaimpia koulutuksia, ja siinä nähdään selkeä ylisovittamisen ilmiö, ennen kuin koulutusta ehditään keskeyttää.



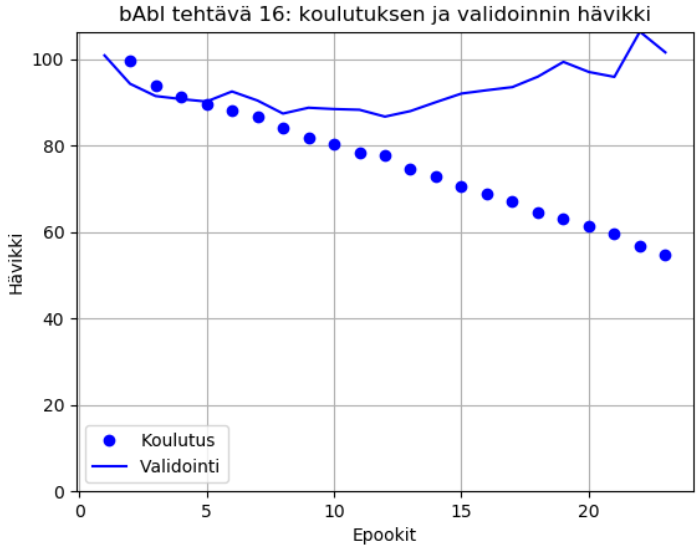
Kuva 25. Esimerkki vaikeasta tehtävästä. Malli alkaa ylisovittaa koulutuksen esimerkkeihin.

Tehtävän 3 tuloksissa on huomioitava, että validointijoukon hävikin kasvu ei näyttänyt vaikuttavan validointitarkkuuteen kuitenkaan merkittävästi



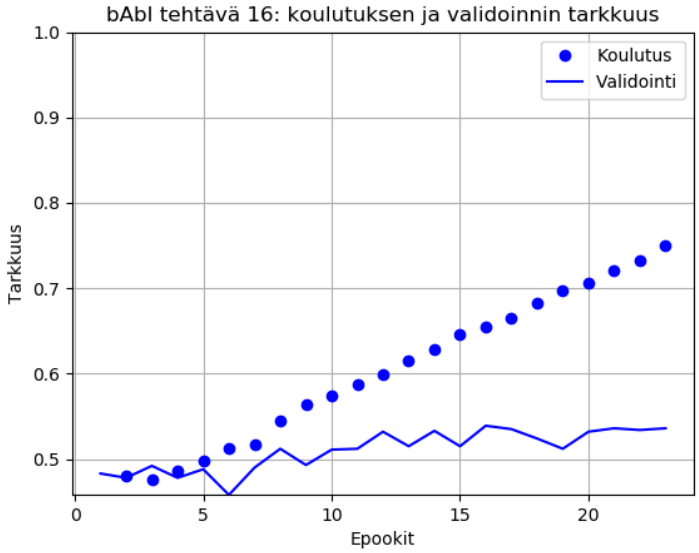
Kuva 26. Esimerkki vaikeasta tehtävästä. Koulutustarkkuus nousee ilman, että validointi seuraisi mukana.

Tehtävä 16 on hyvä esimerkki tehtävästä, jossa verkko ei opi juurikaan.



Kuva 27. Hävikki kasvaa validointiesimerkeissä toisin kuin koulutusesimerkeissä, mitkä ovat klassisia ylisovittamisen merkkejä.

Tehtävän 16 tarkkuudessa nähdään sama ilmiö, validointitarkkuus pysyi tasaisena samalla kun malli ylisovittaa koulutusesimerkkeihin (ks. kuva 28).



Kuva 28. Tarkkuus nousee hitaasti koulutusjoukossa, mutta validoidessa ei nähdä parannusta.

Insinöörityön koulutustulokset olivat odotetusti hyvin samankaltaiset tutkimuksen ja lähdekoodin tekijän tuloksien kanssa. bAbI-tehtävä 3:n iso ero tarkkuudessa lähdekoodin ja tutkimuspaperin välillä on huomioitu: kyseessä on lähdekoodin tekijän mukaan koodin implementointivirhe (24).

6.2 Johtopäätökset

Isoin yllätys koulutuksen tuloksissa oli useiden tehtävien suorituksen merkittävä hidastuminen näytönohjainta käytettäessä. Tavallisesti pitäisi käydä toisin päin, sillä näytönohjaimien suuremman paralleelisuusprosessoinnin pitäisi huolehtia siitä, että neuroverkkojen koulutusajat supistuvat pienemmiksi. Ainoa syy, jota osaan epäillä, on NVIDIA CUDA -ajureiden viallinen asennus tai näytönohjaimen komponenttiin liittyvä vika. Sovelluspuolella sekä keskusprosessoria että näytönohjainta tukevat Pythonin virtuaaliympäristöt sisälsivät pääasiassa samat paketit pois lukien TensorFlow-kirjasto, josta on joko tavallinen tai näytönohjainta hyödyntävä versio. Näytönohjainta käyttävä virtuaaliympäristö sisälsi myös muutamia pip-paketinhallintajärjestelmän kautta asennettua paketteja, kuten sklearn, joka on eräs tietojoukkojen käsittelyä varten tehty kirjasto. Näillä ei myöskään pitäisi olla mitään vaikutusta suoritusnopeuksiin.

Jatkokehityksessä kannattaisi testata Facebook Research -instituutin tekemää bAbI-dialogitietojoukkoa ja kokeilla oppimistuloksia sen kautta. Tämä mahdollisesti edellyttäisi verkon käyttämän sanaston uusimista. Toinen tutkimuksen kohde olisi selvittää, onko DMN-arkkitehtuuriin järkevää yrittää kiinnittää lauseiden luontiin erikoistuvaa toistuvaa neuroverkkoa, jotta malli pystyy tuottamaan muutakin kuin yksisanaisia vastauksia. Kolmas kehityksen suunta olisi kokeilla muuttaa syötetekstisekvenssin sisältöä ajon aikana, ja selvittää sen vaikutusta verkon toimintaan. Tällä tavoin saatettaisiin vastata kysymykseen saako verkossa mahdollisesti piilotettua puheenaiheita sopivissa käytön vaiheissa.

7 Yhteenveto

Insinöörityössä tutkittiin koneoppimisen tekniikoita psykoedukatiivisen viihdesovelluksen ja keskustelubottien näkökulmasta. Tutkimuksessa perehdyttiin erinäisiin luonnollisen kielen prosessoinnin tekniikoihin, keskustelubottien tulevaisuuden trendeihin ja tekstinkäsittelyyn erikoistuviin neuroverkkomalleihin. Neuroverkkojen yleiset toimintaperiaatteet tuli kerrattua etenkin yksinkertaisten luokittelijaverkkojen osalta, ja toistuvien neuroverkkojen perustoiminta tuli selvitettyä dynaamisen muistiverkon koodin tutkimisen ohessa paremman käsityksen luomiseksi valitun algoritmin toiminnasta.

Työn tavoitteena oli löytää ja testata neuroverkkoarkkitehtuuria, joka kykenisi täyttämään seuraavat kriteerit: Mallin tulisi

- ymmärtää pelaajan aiheeseen liittyviä vapaasti laadittuja kysymyksiä
- tuottaa kokonaisia lauseita vastauksiksi kysymyksiin
- kyetä ymmärtämään ja käyttämään tunnekieleen liittyvää sanastoa
- pystyä keskustelemaan kuvitteellisista tapahtumista
- ylläpitää yhtenäistä persoonaa keskustelun aikana
- muistaa aikaisemmat pelaajan kysymykset ja reagoida toistuviin kysymyksiin eri tavalla
- osata päätellä, milloin pelaajan kysymys on tilanteeseen sopimaton ja laittaa viestin siitä eteenpäin.

Kriteerit jäivät insinöörityön osalta avoimiksi kysymyksiksi, koska työhön varattu aika loppui kesken. Valtaosa kehitysajasta meni siihen, että verkkoa yritettiin saada toimimaan näytönohjaimen kanssa, ja se edellytti työn loppupuolella lähdekoodin vaihtoa toiseen dynaamisen muistiverkon toteutukseen, joka käytti Theano-kirjaston sijasta TensorFlow-kirjaston toteutusta. Samalla Python-projektin virtuaaliympäristössä pakettien hakeminen ja asentaminen piti tehdä pip-paketinhallintajärjestelmän sijasta Anaconda-alustan kautta, jotta koodia pystyi suorittamaan komentoriviltä. Syy tähän ei selvinnyt.

Mallin koulutustulokset bAbI-projektin 20:ssä kysymys vastaus -tehtävässä on onnistuneesti varmennettu yksittäiseltä pöytäkoneelta käsin, ja ne muistuttavat merkittävästi lähdekoodin tekijän ja muistiverkon kehittäneiden tutkijoiden tuloksia. Voidaan toisin sanoen olettaa, että verkon toteutus oppii tutkimuksessa oletetulla tavalla. Seuraavaksi verkon testaamiseen on muutama vaihtoehto. Yksi on kokeilla verkon syötteiden

muokkaamista yhteensopiviksi bAbI-projektin dialogitietojoukon esimerkkien kanssa ja taulukoida koulutustulokset kyseisistä kuudesta eri tehtävästä. Tähän kokeiluun liittyen pitäisi todennäköisesti kiinnittää toistuva neuroverkko dynaamisen muistiverkon vastausmoduuliin ja tutkia sitä kautta, miten verkon saisi tulostamaan kokonaisia lauseita, jotka ovat dialogitietojoukon koulutusvastauksille ilmeisiä.

Toinen vaihtoehto on kokeilla yhdistää valmiisiin koulutus- ja validointijoukkoihin omatekoisia esimerkkejä manuaalisesti joko kehitystiimin sisällä tai ulkoistamalla työ kolmannelle osapuolelle. Nämä esimerkit voisivat olla mahdollisimman lähellä pelin kontekstia, jotta niiden perusteella oppiminen yhdessä aikaisempien esimerkkien kanssa antaisi tarkemman vastauksen insinöörityössä esille nostettujen mallin kriteereiden täyttymiselle. Mikäli malli suoriutuu lupaavalla tavalla uudesta koulutusjoukosta, kannattaisi kokeilla sen siirtämistä Unity-pelimoottorin sisälle, josta sen integrointi onnistuisi helpommin itse pelisovellukseen.

Tärkein seikka, mikä selvisi syksyn aikana, oli lopputyön liian kunnianhimoiset tavoitteet aloitusvaiheen lähtötietoihin verrattuna. Suurin osa kehitysajasta meni erinäisten virhetapahtumien korjaamiseen lähdekoodissa tai python-kirjastojen toisiinsa sopimattomissa versioissa. Jos nykytiedoilla aloittaisi työn uudelleen, todennäköisesti yrittäisin päästä tavoitteisiin käyttämällä keskusteluboteille valmiiksi kehitettyjä ratkaisuja, kuten Wit.ai, DialogFlow tai Rasa.ai, jotka ovat kaikki ilmaisia alustoja. Todennäköisesti alustoihin pääsee työelämässä kuitenkin perehtymään, mikäli jatkan koneoppimisen alalla tekstinkäsittelyn tehtävissä.

Lähteet

1 Z, Anne. 2018. App Review: Woebot. Verkkoaineisto. emotion / know. <https://emotionknow.com/2018/07/16/app-review-woebot/>. Luettu 20.11.2018.

2 Rao, Arun. 2018. Woebot - Your AI Cognitive Behavioral Therapist: An Interview with Alison Darcy. Verkkoaineisto. Chatbots Magazine. <https://chatbotsmagazine.com/woebot-your-ai-cognitive-behavioral-therapist-an-interview-with-alison-darcy-b69ac238af45>. Luettu 29.10.2018

3 RPG Conversation: Never Winter Nights. 2017. Verkkoaineisto. How to Make an RPG. <http://howtomakeanrpg.com/a/rpg-conversation-never-winter-nights.html>. Luettu 19.11.2018.

4 Ellison, Brent. 2008. Defining Dialogue Systems. Verkkoaineisto. Gamasutra. <https://www.gamasutra.com/view/feature/132116/defining_dialogue_systems.php>. Luettu 12.11.2018.

5 Turing, Alan. 1950. Computing Machinery and Intelligence. Verkkoversio. Mind. Vol 59, s. 433-360. <https://www.abelard.org/turpap/turpap.php>. Luettu 17.11.2018.

6 Ireland, Corydon. 2012. Alan Turing at 100. Verkkoaineisto. The Harvard Gazette. <https://news.harvard.edu/gazette/story/2012/09/alan-turing-at-100/>. Luettu 19.11.2018.

7 Rising, David. 2008. AI Pioneer Joseph Weisenbaum dies. Verkkoaineisto. NBC News. <http://www.nbcnews.com/id/23615538/ns/technology_and_science-innovation/t/ai-pioneer-joseph-weizenbaum-dies/>. Luettu 27.11.2018.

8 Garber, Megan. 2014. When PARRY Met ELIZA: A Ridiculous Chatbot Conversation From 1972. Verkkoaineisto. The Atlantic. <https://www.theatlantic.com/technology/archive/2014/06/when-parry-met-eliza-a-ridiculous-chatbot-conversation-from-1972/372428/>. Luettu 18.11.2018.

9 A Visual History Of Chatbots. 2018.Verkkoaineisto. Chatbots Magazine. <https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2>. Luettu 16.11.2018.

10 Booth, Jessica. 2015. Flashback: 15 Internet Trends From The Early 2000's That We Really Miss. Verkkoaineisto. <http://www.gurl.com/2013/08/15/internet-trends-stuff-from-early-2000s-1990s/>. Luettu 21.11.2018.

11 Lazăr, Marilena & Militaru, Diana. 2015. The Role of Decision Trees in Natural Language Processing. Verkkoaineisto. <http://www.imsar.ro/SISOM_Papers_2015/SISOM_2015_A_04.pdf>. Luettu 21.11.2018.

12 Luthy, Melvin. 2004. A Comparative Generative-Junction Approach to Finnish Morphosyntax. Verkkoaineisto. Linguistic Technologies, Inc. <http://www.junction-grammar.com/html/morphosyntax_of_finnish_passiv.htm>. Luettu 21.11.2018.

13 Jozefowicz, Rafal, et al. 2015. Exploring the Limits of Language Modeling. Verkkoaineisto. <https://arxiv.org/pdf/1602.02410.pdf>. Luettu 18.11.2018.

14 Perez, Sarah. 2015. Consumers Spend 85% Of Time On Smartphones in Apps, But Only 5 Apps See Heavy Use. Verkkoaineisto. TechCrunch. <https://techcrunch.com/2015/06/22/consumers-spend-85-of-time-on-smartphones-in-apps-but-only-5-apps-see-heavy-use/>. Luettu 21.11.2018.

15 Fallah, Georges. 2017. Digital Marketing Trends for 2018: Micro-Scale & Richer Content. Verkkoaineisto. <https://www.vbout.com/blog/Digital-Marketing-Trends-for-2018/>. Luettu 21.11.2018.

16 Zaman, Tauhid. 2018. Even a few bots can shift public opinion in big ways. Verkkoaineisto. <https://phys.org/news/2018-11-bots-shift-opinion-big-ways.html>. Luettu 21.11.2018.

17 Train your first neural network: basic classification. 2018. Verkkoaineisto. TensorFlow. <https://www.tensorflow.org/tutorials/keras/basic_classification>. Luettu 27.11.2018.

18 Sharma, Abhishek. 2017. What is the differences between artificial neural network (computer science) and biological neural network? Verkkoaineisto. <https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network>. Luettu 21.11.2018.

19 Schmidhuber, Jürgen. 2015. Who Invented Backpropagation? Verkkoaineisto. <http://people.idsia.ch/~juergen/who-invented-backpropagation.html>. Luettu 23.11.2018.

20 Britz, Denny. 2015. Recurrent Neural Networks Tutorial, Part 1 -Introduction to RNNs. Verkkoaineisto. WildML. <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>. Luettu 22.11.2018.

21 Karpathy, Andrej. 2015. The Unreasonable Effectiveness of Recurrent Neural Networks. Verkkoaineisto. Hacker's guide to Neural Networks. <https://karpathy.github.io/2015/05/21/rnn-effectiveness/>. Luettu 23.11.2018.

22 Kumar, Ankit, et al. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. Verkkoaineisto. MetaMind. <https://arxiv.org/pdf/1506.07285.pdf>. Luettu 13.8.2018.

23 Caballero, Ethan. 2016. Improved-Dynamic-Memory-Networks-DMN-plus. Verkkoaineisto. GitHub. <https://github.com/ethancaballero/Improved-Dynamic-Memory-Networks-DMN-plus>. Luettu 3.8.2018.

24 Barron, Alex. 2018. Dynamic-Memory-Networks-in-TensorFlow. Verkkoaineisto. GitHub. <https://github.com/barronalex/Dynamic-Memory-Networks-in-TensorFlow>. Luettu 10.8.2018.

25 Weston, Jason, et al. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. Verkkoaineisto. Facebook AI Research. <https://arxiv.org/pdf/1502.05698.pdf>. Luettu 23.11.2018.

26 The bAbI project. 2018. Verkkoaineisto. Facebook Research. <https://research.fb.com/downloads/babi/>. Luettu 10.8.2018.

27 Python documentation. 2018. Verkkoaineisto. TensorFlow. <https://www.tensorflow.org/api_docs/python/tf/nn/sparse_softmax_cross_entropy_with_logits>. Luettu 22.11.2018.

28 Underfitting vs. Overfitting. 2018. Verkkoaineisto. Scitit Learn. <https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html>. Luettu 23.11.2018.

Muokattu osa koulutuskoodista

```
print('==> starting training')
for epoch in range(config.max_epochs):
    print('Epoch {}'.format(epoch + 1))
    start = time.time()

    train_loss, train_accuracy = model.run_epoch(
        session, model.train, epoch, train_writer,
        train_op=model.train_step, train=True)
    valid_loss, valid_accuracy = model.run_epoch(session, model.valid)

    # print('Training loss: {}'.format(train_loss))
    # print('Validation loss: {}'.format(valid_loss))
    # print('Training accuracy: {}'.format(train_accuracy))
    # print('Validation accuracy: {}'.format(valid_accuracy))

    epoch_history.append(epoch + 1)
    train_l_history.append(train_loss)
    valid_l_history.append(valid_loss)
    train_a_history.append(train_accuracy)
    valid_a_history.append(valid_accuracy)

    if valid_loss < best_val_loss:
        best_val_loss = valid_loss
        best_val_epoch = epoch
        if best_val_loss < best_overall_val_loss:
            # print('Saving weights')
            best_overall_val_loss = best_val_loss
            best_val_accuracy = valid_accuracy
            saver.save(session, 'weights/task' +
                       str(model.config.babi_id) + '.weights')

    if valid_loss > worst_val_loss:
        worst_val_loss = valid_loss

    if valid_accuracy < worst_val_accuracy:
        worst_val_accuracy = valid_accuracy

    # anneal
    if train_loss > prev_epoch_loss * model.config.anneal_threshold:
        model.config.lr /= model.config.anneal_by
        print('annealed lr to %f' % model.config.lr)

    prev_epoch_loss = train_loss

    if epoch - best_val_epoch > config.early_stopping:
        epoch_stopped = epoch
        break

    end = time.time()
    epoch_time = math.floor(end - start)
    epoch_mins = epoch_time // 60
    epoch_secs = epoch_time - (epoch_mins * 60)
    print('\nEpoch time: {} min {} s'.format(epoch_mins, epoch_secs))

    total_time += epoch_time
```

```
print('\n\nbAbI tehtävä: ' + config.babi_id)
epoch_hours = total_time // 3600
epoch_mins = (total_time - (epoch_hours * 3600)) // 60
epoch_secs = total_time - (epoch_hours * 3600) - (epoch_mins * 60)
print('Koko koulutusaika: {} h {} min {} s'.format(epoch_hours,
                                                   epoch_mins,
                                                   epoch_secs))
print('Pysäytetty epookki: ', epoch_stopped)
print('Maksimi epookit: ', config.max_epochs)
print('Ennalta pysäyttämisen kynnysluku: ', config.early_stopping)
print('Pudotussuhde: ', config.dropout)

print('\nParas validoinnin hävikki:', best_val_loss)
print('Paras validoinnin tarkkuus:', best_val_accuracy)

# Koulutuksen hävikin ja tarkkuuden visualisointi
plt.figure()

plt.plot(epoch_history, train_l_history, 'bo', label='Koulutus')
plt.plot(epoch_history, valid_l_history, 'b', label='Validointi')
plt.title('bAbI tehtävä ' + config.babi_id +
          ': koulutuksen ja validoinnin hävikki')
plt.ylim([0, worst_val_loss])
plt.xlabel('Epookit')
plt.ylabel('Hävikki')
plt.grid()
plt.legend()

plt.figure()

plt.plot(epoch_history, train_a_history, 'bo', label='Koulutus')
plt.plot(epoch_history, valid_a_history, 'b', label='Validointi')
plt.title('bAbI tehtävä ' + config.babi_id +
          ': koulutuksen ja validoinnin tarkkuus')
plt.ylim([worst_val_accuracy, 1])
plt.xlabel('Epookit')
plt.ylabel('Tarkkuus')
plt.grid()
plt.legend()

plt.show()
```

Verkon hyperparametrit

```
class Config(object):
    """Holds model hyperparams and data information."""

    batch_size = 100
    embed_size = 80
    hidden_size = 80

    max_epochs = 100
    early_stopping = 10

    dropout = 0.9
    lr = 0.001
    l2 = 0.001

    cap_grads = False
    max_grad_val = 10
    noisy_grads = False

    word2vec_init = False
    embedding_init = np.sqrt(3)

    # NOTE not currently used hence non-sensical anneal_threshold
    anneal_threshold = 1000
    anneal_by = 1.5

    num_hops = 3
    num_attention_features = 4

    max_allowed_inputs = 130
    num_train = 9000

    floatX = np.float32

    babi_id = "1"
    babi_test_id = ""

    train_mode = True
```