TALLINNA TEHNIKAÜLIKOOL
Tarkvarateaduse instituut

Heidi Korp 164418IAPB

ANDMESTIKU KOOSTAMINE EESTIKEELSE KÕNEABIPROGRAMMI SÜNTESAATORI JAOKS

bakalaureusetöö

Juhendajad: Martin Rebane
MSc
Kairit Sirts
PhD

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Heidi Korp

26.05.2019

Annotatsioon

Bakalaureuse lõputöö eesmärgiks on luua eestikeelne andmestik, mida oleks võimalik kasutada augmentatiivse ja alternatiivse kõneabiprogrammi loomiseks. See tähendab, et andmestik peaks koosnema sõna algvormidest (lemmadest) koos morfoloogiliste märgenditega, et algset sõna hiljem tagasi sünteesida. Andmestiku loomise aluseks võeti eestikeelsetelt veebisaitidelt kogutud lausetest koosnev ETenTen korpus [1] ning Eesti Keele Instituudi koostatud eesti keele põhisõnavara sõnastik [2]. Töö teises pooles tehti andmestikuga esimesi eksperimente. Morfoloogiliste märgendite abil algse sõna sünteesimine on võimalik juhul, kui lemmadele suudetakse vastavusse panna korrektne morfoloogiline märgend. Märgendite ennustamiseks treeniti neurovõrk. Töös uuritakse erinevaid sõnavektoreid, mida kasutatakse mudeli treenimisel. Sõnavektorite eesmärgiks on luua seoseid andmestiku sõnade vahel.

Töö tulemuseks on lemmatiseeritud ehk sõna algvormidest koosnev eestikeelne keelekorpus. Lisaks on võimalik eksperimentidest järeldada, milliseid sõnavektoreid peaks kõneabiprogrammi loomisel kasutama, kui seda treenida eestikeelse lemmatiseeritud korpuse peal.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 5 peatükki, 11 joonist, 5 tabelit.

Abstract

Dataset for the Text Synthesizer Component of the Estonian Speech Aid Software

The aim of the thesis at hand is to create a dataset that could be used is an Alternative and Augmentative Communication (AAC) program. In an icon-based AAC system, a user selects icons that best describe his thoughts, and so a sentence is formed, each word represented by an icon. All these words appear in their dictionary forms (lemma) and thus, it is necessary to create a dataset consisting of lemmatized sentences. The basis for creating the dataset is the ETenTen text corpus [3] that consists of sentences scraped from Estonian websites and the Basic Estonian Dictionary [4].

The second part of the thesis describes the first experiments done with the previously created lemmatized corpus. Morphological part-of-speech was added to each lemma in the corpus to be able to synthesize the original word from the lemma. Morphological part-of-speech contains information about the word's tense, case, verb form etc. The synthesizing can only be successful in case each lemma is paired up with its correct morphological part-of-speech. For this purpose, a neural network was trained that could be initialized with pretrained word embeddings. The purpose for using pretrained embeddings is to better classify words in the dataset.

As a result, an Estonian lemmatized text corpus was created. The analysis of the experiments done with the dataset provide an overview, what kind of embeddings perform the best on a lemmatized dataset and what could be done to improve the results.

The thesis is in Estonian and contains 34 pages of text, 5 chapters, 11 figures, 5 tables.

Lühendite ja mõistete sõnastik

AAC	*Alternative and Augmentative* Communication, augmentatiivne ja alternatiivne kommunikatsioon
Baassõnavara	Sõnad, mille jaoks eksisteerib ikoon
CBOW	*Continuous bag of words*, vektoresituse meetod, kus vaadeldava sõna ümbruses olevate sõnade järjestus pole oluline
Edasilevi	*Forward propagation,* igal närvivõrgu kihil saadud tulemuste kandumine järgmisesse kihti
Epohh	*Epoch, ajavahemik,* mille käigus läbib kogu treeningandmestik tehisnärvivõrku ühe korra
Gradiendi hajumine	*Gradient vanishing,* gradiendi lähenemine nullile
Gradientlaskumine	*Gradient descent, algoritm,* mida kasutatakse mudeli parameetrite uuendamiseks
Ikoon	*pilt,* mis kirjeldab ühte sõna või ühte tegevust
Jadamärgendmaise mudel	*tagger model*, neurovõrgu arhitektuur, mis võimaldab sõnadele ennustada morfoloogilisi märgendeid
Kooder	*Encoder,* loob sõnadele nende tähtede järjekorra ning konteksti põhjal vektoresituse
Korpus	*Text corpus*, suur tekstist koosnev andmestik
Kujutis	*dictionary*, Pythoni andmetüüp, kus igale järjestamata võtmeelemendile on vastavusse seatud teatud väärtus
Lemma	Sõna algvorm
LSTM	*Long-Short Term memory*, mäluomadusega arhitektuur, mida kasutatakse rekurrentsetes närvivõrkudes
Morfoloogiline märgend	*Morphological part-of-speech,* lisainformatsioon sõna kohta, mis võimaldab lemmast sünteesida sõna algvormi. Eesti keeles sõnavorm (kääne, pööre) ja mitmus-ainsus
Neuromudel	*Neural network model*, närvivõrgu arhitektuuriga mudel, millel on kindel ülesanne
NLP	*Natural Language Processing,* loomuliku keele töötlus
PECS	*Picture Exchange Communication System*, ikoonide valimisel põhinev suhtlusmetoodika

Plokk	*Batch,* hulk treeningandmeid, mida treenitakse korraga
RNN	*Recurrential Neural Network,* rekurrentne närvivõrk
Skip-gram	Vektoresituse meetod, kus vaadeldava sõna lähiümbruses olevad sõnad omavad suuremat tähtsust kui kaugemal olevad
Sõnavektor	*Word embedding,* sõnade ja fraaside esitamine reaalarvulistest numbritest koosneva vektorina
Tagasilevi	*Back-propagation*, närvivõrgu kaalude uuendamine eelmistes kihtides
Täide	*Padding,* vektoritele täidissümboli lisamine, et kõik vektorid maatriksis oleksid ühepikkused
Täpsus	*Accuracy,* protsent, millisel määral suudab närvivõrgu mudel teha õigeid ennustusi
Ülesobitumine	*Overfitting,* närvivõrgu liiga pikal treenimisel tekkiv olukord, kus mudel ei suuda teha üldistusi ega täpseid ennustusi
Varajane lõpetamine	*Early stopping*, regulatsioonimeetod, mis tuvastab olukorra, kus mudel on saavutanud oma parima õppimistulemuse

Sisukord

Jooniste loetelu

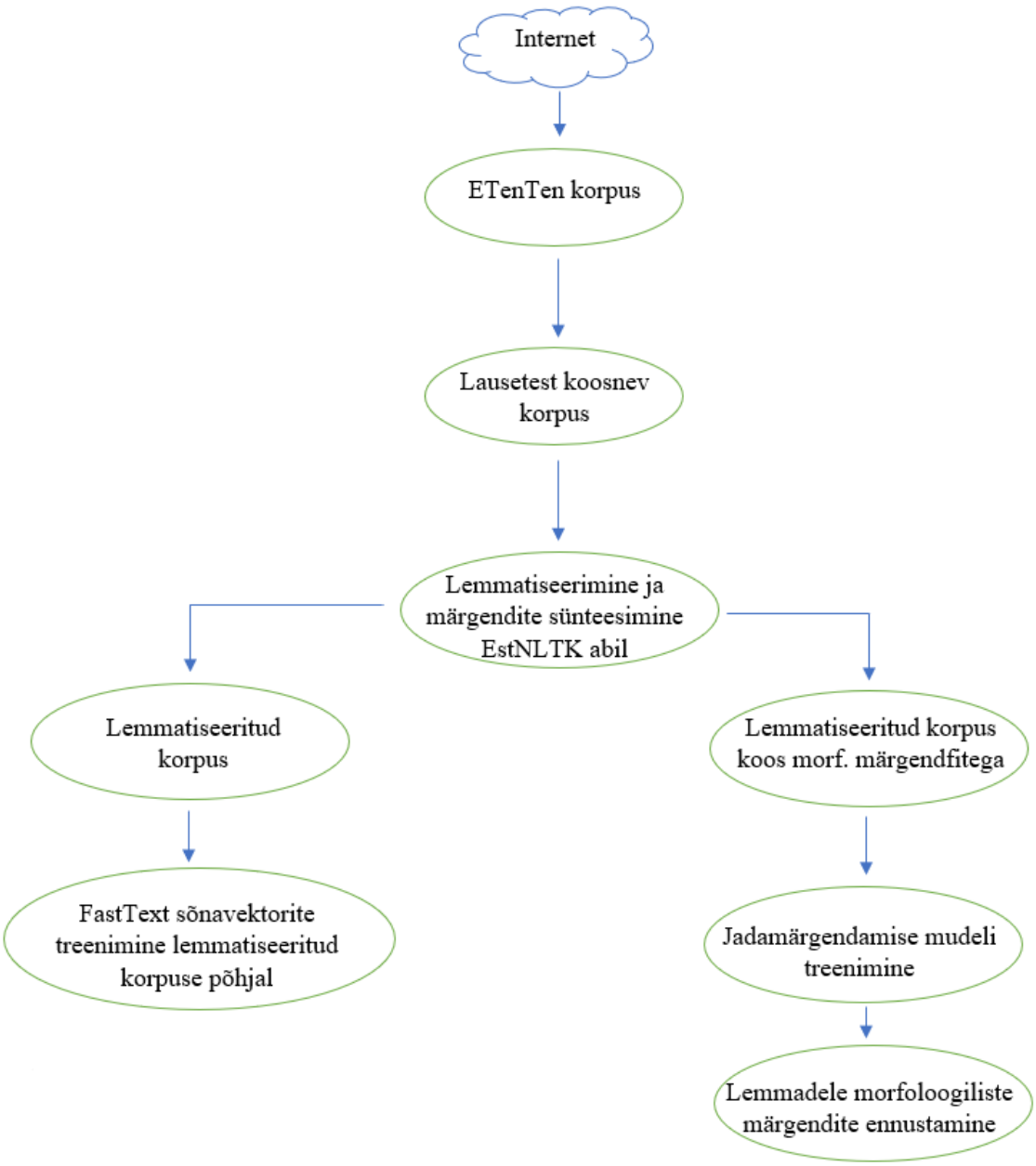
Tabelite loetelu

1 Sissejuhatus

Augmentative and Alternative Communication (AAC) viitab erinevatele suhtlust abistavatele meetoditele, mida kasutatakse lisaks tavalisele kõnele [3]. Tänu tehnoloogia ning meditsiini arengule on kasvanud inimeste hulk, kellel võiks AAC meetodite kasutamisest igapäevases suhtluses kasu olla, näiteks Downi sündroomiga lastel [4]. Samuti on loodud erinevaid programme, mis on suunatud konkreetsele inimrühmale, näiteks lasteaia- ning eelkooliealistele lastele.

AAC programmide loomisel masinõppe meetodeid kasutades on tähtis sobiva andmestiku olemasolu. Eestis ei ole teadaolevalt konkreetselt AAC programmi jaoks loodud keelekorpust. Antud töös on võetud eesmärgiks luua sõna algvormidest koosnev keelekorpus, mida saaks kergesti kasutada näiteks *Picture Exchange Communication System* (PECS) ehk ikoonide valimisel põhineva suhtlusprogrammi loomiseks. Aluseks võetakse ETenTen keelekorpus [1], mis koosneb eestikeelsetest veebisaitidelt kogutud lausetest ning genereeritakse igale sõnale tema algvorm (lemmatiseeritakse). Lemmatiseeritud keelekorpuse loomiseks koostatakse arvutiprogramm Pythoni programmeerimiskeeles.

Lõputöö teise osa fookuses on masinõppel põhinevate eksperimentide läbi viimine eelnevalt loodud lemmatiseeritud keelekorpusega. Kuna lemmadest peab olema võimalik sünteesida algset sõna, lisatakse igale lemmale lisainformatsiooni ehk tema morfoloogiline märgend. Lemma ja korrektse märgendi vastavusse viimiseks treenitakse jadamärgendamise neuromudelit. Mudeli põhjal on võimalik teha ennustusi, kus sisendiks on lemmatiseeritud lause ning tulemuseks vastavad morfoloogilised märgendid igale lemmale. Järgnev Joonis 1 kirjeldab antud lõputöö põhilist struktuuri.



Joonis 1. Lõputöö põhiliste tegevuste struktuurne kirjeldus.

Neuromudelit on võimalik initsialiseerida eeltreenitud sõnavektoritega. Tekstipõhiste andmete treenimisel esitatakse sõnu ja fraase reaalarvulistest numbritest koosnevate vektoritena. Vektori elementides väljendub sõna seos teiste korpuses leiduvate sõnadega ning mida suuremal andmestikul on vektoreid treenitud, seda täpsemad on seosed. Kuna lemmatiseeritud andmestik on väiksem kui see andmestik, mida on kasutatud eeltreenitud sõnavektorite treenimiseks, sisaldavad eeltreenitud vektorid sõnade kohta rohkem informatsiooni, mida saab mudeli treenimisel kasutada.

Eksperimentide käigus võrreldakse kolme vektoresitust, et saada teada, milline neist annab kõige täpsema tulemuse lemmadele morfoloogilise märgendi ennustamiseks. Uuritavad vektoresitused on järgmised:

- juhuslikult initsialiseeritud sõnavektorid;
- FastText eeltreenitud sõnavektorid [5];
- FastText lemmade põhjal treenitud sõnavektorid

Töö jaguneb neljaks peatükiks: sisuline taust, tehniline taust, andmestiku koostamine ning eksperimendid andmestikuga. Sisulises taustas antakse ülevaade sarnastest töödest ning AAC süsteemides kasutatava andmestiku omadustest, tehnilises taustas tutvustatakse töös kasutatud tehnilisi lahendusi, andmestiku koostamises kirjeldatakse korpuse loomise protsessi ja eksperimentides andmestikuga tutvustatakse kolme vektoresitust ning nendega saadud tulemusi neuromudeli treenimisel.

Tulemustest selgus, et kolme vektoresituste tulemused erinesid väga vähe. 13 835 lausest koosneva andmestiku treenimisel saavutati täpsus ligikaudu 63%, kusjuures lemmade põhjal treenitud mudeli täpsus oli madalam, 58%. Pärast andmestiku suurendamist 252 343 lauseni jäid vektoresituste tulemused ligikaudu samaks, kuid üldine täpsus paranes, jõudes 70%-ni.

2 Sisuline taust

Antud töö eesmärgiks on koostada andmestik, mida oleks võimalik kasutada kõneabiprogrammides. Kuna sellised kõneabiprogrammid töötavad sõnu kirjeldavate ikoonide valimise põhimõttel, tuleb andmestiku loomisel sellega arvestada. Antud töö autori käsutuses ei ole täpset informatsiooni, kui paljudele eestikeelsetele sõnadele leidub vastav ikoon. Selle tõttu on tehtud eeldus, et eesti keele põhisõnavara sõnastik [2] on suurim kogu eestikeelseid sõnu, millele on juba olemas või millele tõenäoliselt luuakse tulevikus ikoon.

Andmestiku koostamisel tasub samuti tähelepanu pöörata sellele, et selles sisalduksid igapäevakõnes enim kasutatud sõnad. Järgnevad alapeatükid selgitavad baassõnade tähtsust ning kuidas on varem kõneabiprogrammide jaoks andmestikku loodud.

2.1 Baassõnade tähtsus

Teadaolevalt on paljudel kõnehäiretega lastel raskuseks uute sõnade õppimine [6]. Seepärast on oluline, et loodav AAC tarkvara või muu strateegia sisaldaks enimlevinud sõnu, mis on hädavajalikud uute, keerulisemate sõnade õppimiseks. Bracken ja Crawford [7] on töötanud välja 300 ingliskeelset baassõna, mis peaksid sisalduma igas eelkooliealistele lastele mõeldud AAC programmis. Need sõnad võib jagada järgmistesse rühmadesse: värvid, tähed, numbrid ja loendamine, suurused ja võrdlus, kujundid, suunda ning asukohta kirjeldavad sõnad, tundeid ja inimest kirjeldavad omadussõnad, tekstuuri ning materjali kirjeldavad omadussõnad, hulga määratlemine ning aeg [7].

McCarthy, Schwarzi ja Ashworth [6] uurisid baassõnade olemasolu ning kättesaadavust Ameerika Ühendriikides laialdaselt kasutusel olevates AAC programmides. Baassõnade esinemist arvestati igas kategoorias eraldi ning tulemustest selgub, et kõik uuritud programmid sisaldavad keskmiselt 60% baassõnadest. Kõige vähem leidus programmides võrdlusi (sarnane, samasugune, erinev jne) ning suurusi kirjeldavaid sõnu.

Baassõnade kättesaadavust hinnati klikkide arvuga, mida näiteks elektroonsetes tahvelarvutites peab sõna üles leidmiseks tegema. Tulemused näitasid, et keskmiselt on vaja mõne baassõna leidmiseks teha 2-3 klikki ning tihti asuvad need sõnad mõnes suuremas kategoorias. Näiteks numbrite kasutamiseks peab liikuma avavaatest „matemaatika“ teema alla. Liigne klikkide arv nõuab lastelt head mälu ning sõnade seostamist teema pealkirjaga, mis võib sagedase kasutuse korral muutuda ebamugavaks. Uurimistöö autorid soovitavad luua paindlikud programmid, kus oleks võimalik luua avalehele otseteid vastavalt hetkel käsil olevale teemale [6].

2.2 Sarnased tööd

Antud bakalaureusetöö koostamisel on aluseks võetud kaks teaduslikku uurimistööd, mis käsitlevad sarnase probleemi lahendamist.

2.2.1 Ikoonandmestiku koostamine

Lõputöö koostamisel on aluseks võetud Dudy ja Bedricki [8] artikkel, mis kirjeldab Symbolstix ikoonsõnastikule treeningandmestiku loomist, mudelite treenimist ja nende sobivust kasutamaks neid AAC programmides. Ikoonsõnastiku näidiselt [8, Joonis 2] on näha, et kujutatud pilti on võimalik mitmeti mõista. Eksperimendis kasutatud ikoonsõnastik sisaldab palju meta-informatsiooni, kuid ikoonidest lause genereerimiseks pole loodud sobivat tekstikorpust. Sõnavektorite initsialiseerimiseks võtsid Dudy ja Bedrick aluseks tekstikorpuse ja püüdsid selle sõnu panna vastavusse ikoonidega. Nii saavutasid nad korpuse, mille kõik sõnad on võimalik esitada vektorkujul ehk asendada ikoonidega.

Joonis 2. Näidisikoonid: kartma ja jalga hammustama [8].

Treeningkorpuse aluseks võeti SubtlexUS korpus, mis koosneb filmide ja telesaadete subtiitritest, kuna need laused imiteerivad suurel määral igapäevaelu kõnekasutust. Samal põhimõttel on ka antud bakalaureusetöös valitud andmestiku koostamise aluseks tekstikorpus, mis sisaldab veebilehtedelt kogutud lauseid, muu hulgas blogipostitusi ja kommentaare.

Autorid katsetasid erinevaid võimalusi andmestiku loomiseks:

- kahte erinevat vektoresitust: ikoonide põhjal treenitud vektoreid ning eeltreenitud sõnavektoreid. Mõlema puhul kasutati eeltreenitud vektoreid, aga ühel juhul pidid need ikoonide sõnadega kattuma;
- kahte treeningandmestikku, kus üks koosnes ainult ikoonidele vastavatest sõnadest ning teine sisaldas ka muid sõnu;
- teise tekstikorpuse katsetamist, mitte Subtlex-i

Mõlema treeningandmestiku ettevalmistamiseks eemaldati treeningkorpustest kirjavahemärgid. Eksperimentide tulemustest selgus, et kõikide vektoresituste puhul olid tulemused üsna sarnased. Vektoresituse valimisel on aga oluline, et kõiki ikoone oleks võimalik esitada vektoritena kui ka see, et kõikidel treeningkorpuse sõnadel leiduks vektoresitus.

Treeningkorpuse valimiseks võrreldi omavahel korpust, mis koosnes täielikult ikoonesitusega sõnadest ning korpusest, kus oli lisaks muid sõnu. Tulemusena selgus, et mudeli treenimiseks sobib paremini korpus, mis sisaldab eeltreenitud vektoreid kui ka <unk> märgendeid, mille kohta vektor puudub. Põhjuseks võib olla see, et ikoonkorpus koosneb liiga lühikestest lausetest, mille tõttu mudeli õppimisvõime langeb.

Lisaks subtiitritele katsetati antud uurimuses ka SMS sõnumitest koosnevat korpust kombineeritult väikese, umbes 6000 lausest koosneva korpusega. Tulemustest selgus, et viimasena loodud korpuse efektiivsus oli madal, tõenäoliselt väikese mahu ning ülesobitumise (*overfitting*) tõttu.

Erinevalt Dudy ja Bedricki tööst ei eemaldatud käesolevas töös treenimiseks kasutatavast andmestikust kirjavahemärke ning eksperimente tehti vaid korpusega, mille kõik sõnad

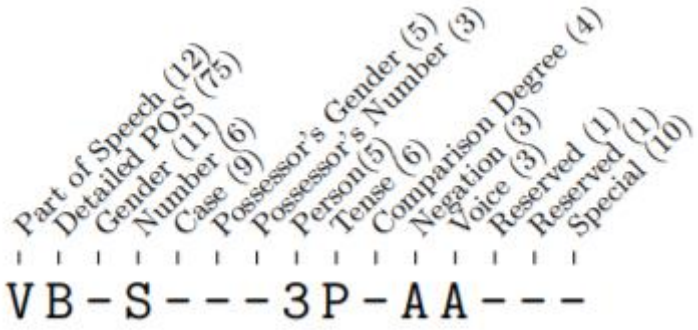
leidusid baassõnavaras. Käesoleva bakalaureusetöö puhul on tehtud eeldus, et kõik baassõnavaras sisalduvaid sõnu on võimalik esitada ikoonidena.

Antud töö tulemustest selgus sarnaselt eelpool kirjeldatud eksperimentidele, et erinevad vektoresitused ei anna märkimisväärselt teistsuguseid tulemusi. Kuna mudel saavutas parima tulemuse juba esimestel kordadel, näiteks ühel juhul neljandal, teisel juhul teisel epohhil (*epoch*) ehk treeningsammul, siis võib järeldada, et sarnaselt Dudy ja Bedricki [8] töö tulemustele on treeningandmestik liiga väike.

2.2.2 Lemmade ja morfoloogiliste märgendite korraga ennustamine

Teine töö, mis on antud lõputöö koostamise aluseks, on D. Kondratyuki, T. Gavenčiaki, M. Straka ja J. Hajiči artikkel „LemmaTag: Jointly Tagging and Lemmatizing for Morphologically Rich Languages with BRNNs“ [9]. Töö fookuses on neurovõrgu arhitektuur LemmaTag, mis on võimeline sünteesima sõnadele nende morfoloogilisi märgendeid ning lemmasid korraga. Lemmade ja märgendite sünteesimiseks kasutatakse kahesuunalist rekurrentset närvivõrku.

Ülesande lahendamiseks vaadatakse morfoloogiliselt keerulisi keeli, nagu saksa keel, tšehhi keel ning araabia keel. Erinevalt näiteks inglise keelest sisaldavad eelnimetatud keelte sõnad grammatilist informatsiooni nagu sugu, käändeid ja ajavorme. Sarnaselt eesti keelele ning antud lõputöös loodud morfoloogilistele märgenditele koosnevad ka Kondratyuki uurimuses loodud märgendid mitmest osast. Märgendi erinevaid osasid selgitab järgnev Joonis 3.



Joonis 3. Tšehhi keelele omased morfoloogiliste märgendite tüübid koos neis esinevate erinevate märgendite arvuga. Kokku on märgendeid umbes 1500. [9].

LemmaTag mudel koosneb kolmest osast. Esimene neist on ühine kooder, mis loob iga sõna jaoks sõnavektori (*word embedding*) tähtede järjekorra ning konteksti põhjal. Teine osa neurovõrgust on *tagger* dekooder, mis saab sisendiks kõikvõimalikud morfoloogilised märgendid ning ennustab nende põhjal igale sõnale tema morfoloogilise märgendi. Saadud märgend ning algne sõna on sisendiks närvivõrgu kolmandale osale, lemmatiseerijale, mis võtab arvesse nii algset sõna kui morfoloogilist märgendit ning sünteesib vastavalt kontekstile õige lemma.

Artikli autorid peavad LemmaTag-i eelisteks võrreldes teiste mudelitega järgmisi omadusi:

- Mudel ei nõua eelnevat ega treenimisjärgset tekstitöötlust;
- Mudel jagab sõnavektoreid, sümbolipõhiseid vektoreid ning RNN kaalusid nii jadamärgendamise funktsionaalsuse kui lemmatiseerijaga. Sel viisil paraneb sünteesimise täpsus ja samal ajal väheneb operatsioonideks vajaminevate parameetrite hulk.
- Mudel on võimeline ennustama morfoloogiliste märgendite alammärgendeid (ennustab eraldi sugu, ajavormi jne) ning saadud märgend on üheks lemmatiseerija sisendiks.

Eksperimentidest LemmaTag mudeliga selgus, et lemmade ja morfoloogiliste märgendite koos sünteesimine annab morfoloogiliselt keeruliste keelte, näiteks tšehhi, saksa ja araabia keele, puhul paremaid tulemusi kui mõlema operatsiooni eraldi teostamine. Teisalt halvenes sellise meetodi rakendamisel morfoloogiliselt lihtsate keelte, näiteks inglise ja lihtsama saksa keele puhul sõnadele lemmade ja märgendite sünteesimise täpsus. Autorite hinnangul võib põhjus olla selles, et märgendite ennustamisel on esmatähtis süntaktiline struktuur ehk sõnade järjekord, mitte konkreetne sõna ise. Paremate tulemuste saavutamiseks soovitavad autorid andmestikku suurendada ning teha samu eksperimente eeltreenitud sõnavektoritega.

Käesolevas lõputöös sünteesitakse sõnadele lemmat ning morfoloogilisi märgendeid eraldi, kasutades EstNLTK teeke [10]. Ühel sõnal võib olla mitu võimalikku lemmat ning nendest antud konteksti arvestades õige valimine on lemmatiseerija kasutaja ülesanne. Kuna ka eesti keel on morfoloogiliselt rikas keel, tasub sarnaselt eelnevalt kirjeldatud

LemmaTag mudelile teha tulevikus eksperimente ka sellise mudeliga, mis arvestab lemmade sünteesimisel sõna konteksti.

Sarnaselt eelpool kirjeldatud eksperimentidele, kasutatakse ka antud töös mitmest komponendist koosnevat morfoloogilist märgendamist. Erinevus seisneb selles, et antud lõputöös ennustatakse morfoloogilist märgendit ühtse tervikuna ehk puudub võimalus alammärgendite eraldi ennustamiseks. Tulevikus võiks eestikeelsetele sõnadele morfoloogiliste märgendite ennustamisel proovida ennustada igat alammärgendit eraldi või katsetada võimalusel rohkemate alammärgendite lisamist.

3 Tehniline taust

Antud lõputöö koostamise aluseks on avalikult kättesaadav ETenTen internetikorpus [1]. Andmete töötlemiseks loodi töö käigus programm Pythoni programmeerimiskeeles. Lemmatiseerimiseks (sõna algvormide genereerimiseks) kasutati EstNLTK teekide kogumikku [10], mis võimaldab eestikeelset teksti töödelda ning analüüsida.

Lõputöö teise osa eesmärgiks on saada teada, kuidas luua uuritavatest võimalustest parim eeldus lemmadest grammatiliselt korrektse eestikeelse sõna genereerimiseks. Selleks treeniti jadamärgendamise mudel (*tagger model*), mis suudaks ennustada lemmavormis sõnadele morfoloogilisi märgendeid, mille alusel saaks hiljem sünteesida õiged sõnavormid. Kasutatava mudeli aluseks on Allen Institute for Artificial Intelligence [11] poolt loodud loomuliku keele töötluse (NLP) teek, täpsemalt AllenNLP Tagger moodul.[1] Mudelis kasutatakse kolme sõnavektoresituse viisi ning võrreldakse neid omavahel.

3.1 EstNLTK teekide kogumik

EstNLTK (NLTK ehk Natural Language ToolKit) on peamiselt Pythonis kirjutatud kogumik teeke eestikeelsete tekstide töötluseks [10]. Lõputöös on kasutatud EstNLTK Text moodulit,[2] mida on tarvis järgmiste tegevuste jaoks: lause sõnadele algvormide leidmine (lemmatiseerimine) ning igale sõnale tema sõnaliigi leidmine (*form* ja *pos*).

3.2 Tehisnärvivõrgud

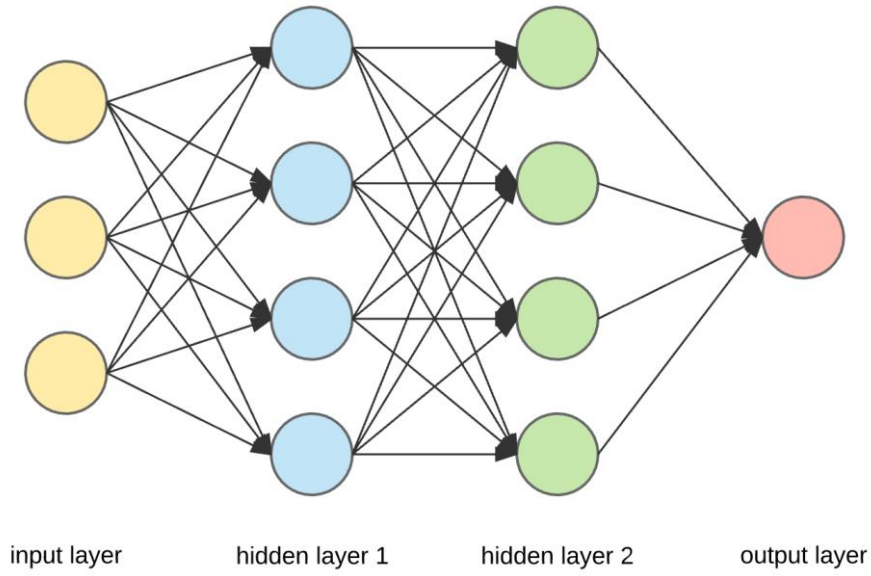
Tehisnärvivõrk (edaspidi närvivõrk) on üldistatud matemaatiline mudel bioloogilisest närvivõrgust [12]. Tehisnärvivõrk koosneb samuti neuronitest, mis moodustavad kihilise sisemise arhitektuuri. Saades sisendiks andmed, läbivad need sisendkihi, kus toimub andmete kategoriseerimine ja seoste leidmine, ning jõuavad väljundkihti.

---

[1] https://github.com/allenai/allennlp/tree/master/tutorials/tagger

[2] https://estnltk.github.io/estnltk/1.3/api/text.html

Närvivõrgud koosnevad kihtidest, millest esimeseks on sisendkiht, järgnevad peidetud kihid ning lõpus asub väljundkiht [13]. Sisendkihte on mitu, millest iga eelmine on järgmise sisendkiht. Iga kiht koosneb neuronitest, millest igaühes toimuvad arvutused eelmise kihi neuronite väljundi põhjal ja tulemus kandub järgmise kihi neuroniteni. Tulemus esitatakse numbrina ja seda nimetatakse kaaluks. Need läbivad aktiveerimisfunktsiooni ning selle väljundi põhjal tehakse otsus, kas antud neuroni tulemus kandub närvivõrgus edasi või mitte. Õpitud kaalud jõuavad väljundkihti ning nende põhjal toimub hiljem ennustamine. Järgnev Joonis [15, Joonis 4] illustreerib närvivõrkude sisemist arhitektuuri.



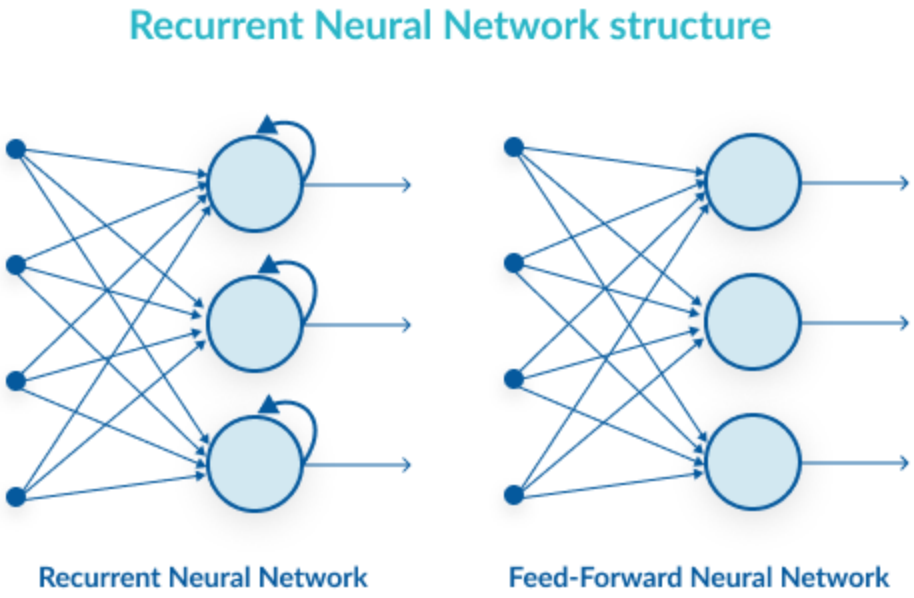
Joonis 4. Tehisnärvivõrgu arhitektuur [14].

Tehisnärvivõrgu üheks tähtsamaks ülesandeks on etteantud funktsioonide rakendamisel ja oodatud tulemuste vaatlemisel kaalude leidmine, et teha sisendandmete jaoks ennustusi.

Igal treeningsammul korrutub sisendinfo kaaluga ning tulemuseks on hinnang antud sisendinfole. Hinnangu ja tegeliku väljundväärtuse vahe on viga, mis korrutub omakorda kaaluga, et anda kaaludele uus täpsem väärtus.

3.3 Rekurrentsed tehisnärvivõrgud

Kui edasilevi (*forward propagation*) tehisnärvivõrgu puhul kandub parandatud tulemus eelmisest kihist järgmisesse, siis rekurrentsete tehisnärvivõrkudel on olemas sisemine mälu, mis võtab arvesse eelnevalt õpitud teadmiseid ning praeguseid sisendeid ja jätab jada andmeid paremini meelde [15]. Põhimõte seisneb selles, et teadmine neuroni kaalu parandusest kandub lisaks järgmisele kihile ka eelmistesse kihtidesse. Vea arvutus ja kaalude korrigeerimine toimub gradientlaskumise leidmise (*gradient descent*) teel. Rekurrentsed närvivõrgud leiavad kasutust eelkõige ülesannetes, kus sisendandmed sõltuvad kontekstist, näiteks tekst, kõne, video jt. Järgnev Joonis [3, Joonis 5] kirjeldab erinevust rekurrentse tehisnärvivõrgu ning edasileviga närvivõrgu vahel.



Joonis 5. Rekurrentse ja edasileviga tehisnärvivõrgu võrdlus [16].

3.4 Long-Short Term Memory

Rekurrentsete närvivõrkude puhul võib kaalude uuendamisel tekkida gradiendi hajumise probleem (*gradient vanishing*) [17]. See tähendab, et kui veafunktsiooni gradient levib tagasi (*back-propagatio*), korrutatakse gradienti teatud kordajaga. Seetõttu muutub gradiendi väärtus ajapikku eksponentsiaalselt kas hästi suureks või väga väikeseks. Mida suurem on neuroni kaal, seda tähtsam on ta närvivõrgus. Gradiendi suur suurus annab

järgmisena uuendatavate neuronite suurema kaalu, muutes nad liialt tähtsalt. Teisalt, kui gradiendi suurus on väga väike, kaotavad järgmisel neuronis oma tähtsuse närvivõrgus.

Üheks võimaluseks gradiendi probleemi lahendamiseks on teha seda LSTM (Long Short-Term Memory) abil [17]. Selle arhitektuur on sarnane arvuti mäluga, sest LSTM suudab uusi andmeid kirjutada, lugeda ning kustutada. Iga uue neuroni kaalu arvutamise järel otsustab LSTM, kas antud informatsioon on piisavalt oluline, ehk kas see teadmine kandub järgmistesse kihtidesse või on kasulikum seda mitte arvestada. LSTM aitab hoida gradiendi väärtust piisavalt kõrgel, et mudeli treenimine oleks võimalikult lühike ning täpsus võimalikult suur. Tänu LSTM-ile saab vältida gradiendi väärtuse hajumise probleemi.

3.5 Sõnavektorid

Tekstitöötluses ja neuromudeleid treenides on tarvis esitada sõnu numbritena. Variante on selleks mitu, ning valik sõltub konkreetse närvivõrgu eesmärgist ja arhitektuurist ning sisendandmete mahust [18].

Üks kõige populaarsem moodus väikese andmehulgaga teksti esitamiseks on sõnavektorite kasutamine [18]. Nii on võimalik kirjeldada igat dokumendis leiduvat sõna semantikat, süntaktilisi sarnasusi teiste ümbritsevate sõnadega ja palju muud. Üks enim kasutatud vektoresituse meetod on Word2Vec, mis on sarnane antud töös kasutatavale FastText vektoresitusele. Word2Vec moodustab suurest tekstikorpusest teatud dimensiooniga vektorruumi. Selles ruumis vastab igale unikaalsele sõnale üks vektor ning need on paigutatud nii, et sarnase kontekstiga sõnad paiknevad üksteise lähedal. Selle tõttu erinevad sarnased sõnad ka vektorestituses üksteisest vähe. Word2Vec [18] võimaldab kasutada kahte vektoresituse viisi: CBOW (*continuous bag of words*) ja *continuous skip-gram*.CBOW puhul antakse sõnadele, mis on sisendsõnaga samas kontekstis, sama kaal ehk tähtsus. Konteksti sõnade arv on igal epohhil sama. *Skip-gram* meetodi puhul saavad sisendsõnale lähemal asuvad sõnad suurema kaalu kui kaugemal olevad. Tulemusena saab vektorestituses sõnaga „kuningas“ sarnase kaalu ka sõna „mees“ ja vastavalt saab sõnale „kuninganna“ sarnase kaalu sõna „naine“.

3.5.1 FastText

FastText on teek, mis võimaldab kasutajatel treenida sõnavektoreid ning sõna klassifikaatoreid [5]. See on kirjutatud C++ programmeerimiskeeles ning selle põhjal on loodud eeltreenitud sõnavektorid 157 keelde. Vektorid on saadaval kahes failis: .bin laiendiga, mis sisaldab endas mudeli kaale, ning .vec laiendiga, kus igal real asub sõna ja sellele vastavad vektorid peidetud kihtides. Vektorite treenimiseks saab kasutada *Skip-gram* meetodid, kus treenitav sõna jagatakse teatud suurusega plokkideks. Vektorite treenimine toimub mitmes lõimes, kuid andmete lugemiseks on üks lõim.

3.6 Masinõppel põhineva mudeli parameetrid

Vastavalt riistvara võimekusele, sõnavektorite dimensioonile ning andmehulga suurusele on tarvis muuta mudeli parameetreid. Järgnevalt on kirjeldatud olulisemaid parameetreid, mida tuli ka antud lõputöös muuta.

3.6.1.1 Epohh ja plokid

Suure hulga sisendandmete korral ei ole võimalik kõiki andmeid korraga mudelile ette anda ning seepärast on tarvis andmehulk mitmeks väiksemaks osaks jagada ning iga korra järel andmete kaalusid uuendada [19].

Epohh on tsükkel, mille vältel jõuab kogu sisendandmestik sisendkihist tehisnärvivõrgu väljundkihti ning tagasi. Kuna arvuti ei suuda kogu andmestikku korraga mällu lugeda, tuleb sisendandmestik omakorda jagada võrdsete suurustega plokkideks (batch).

Kui mudeli treenimiseks kasutada ainult ühte epohhi, on tulemuseks alasobitusega õppimiskõver. Seega, et saada optimaalset tulemust, tuleb treenida mudelit mitme epohhi vältel. Tsüklite arv sõltub aga andmete suurusest ning hinnatavate parameetrite erinevusest.

Suure hulga sisendandmete korral ei ole võimalik kõiki andmeid korraga mudelile ette anda ning seepärast on tarvis andmehulk mitmeks väiksemaks osaks jagada ning iga korra järel andmete kaalusid uuendada [19].

Epohh on tsükkel, mille vältel jõuab kogu sisendandmestik sisendkihist tehisnärvivõrgu väljundkihti ning tagasi. Kuna arvuti ei suuda kogu andmestikku korraga mällu lugeda, tuleb sisendandmestik omakorda jagada võrdsete suurustega plokkideks (batch).

Kui mudeli treenimiseks kasutada ainult ühte epohhi, on tulemuseks alasobitusega õppimiskõver (*underfitting learning curve*). Seega, et saada optimaalset tulemust, tuleb treenida mudelit mitme epohhi vältel. Tsüklite arv sõltub aga andmete suurusest ning hinnatavate parameetrite erinevusest.

3.3.2.2. Varajane lõpetamine

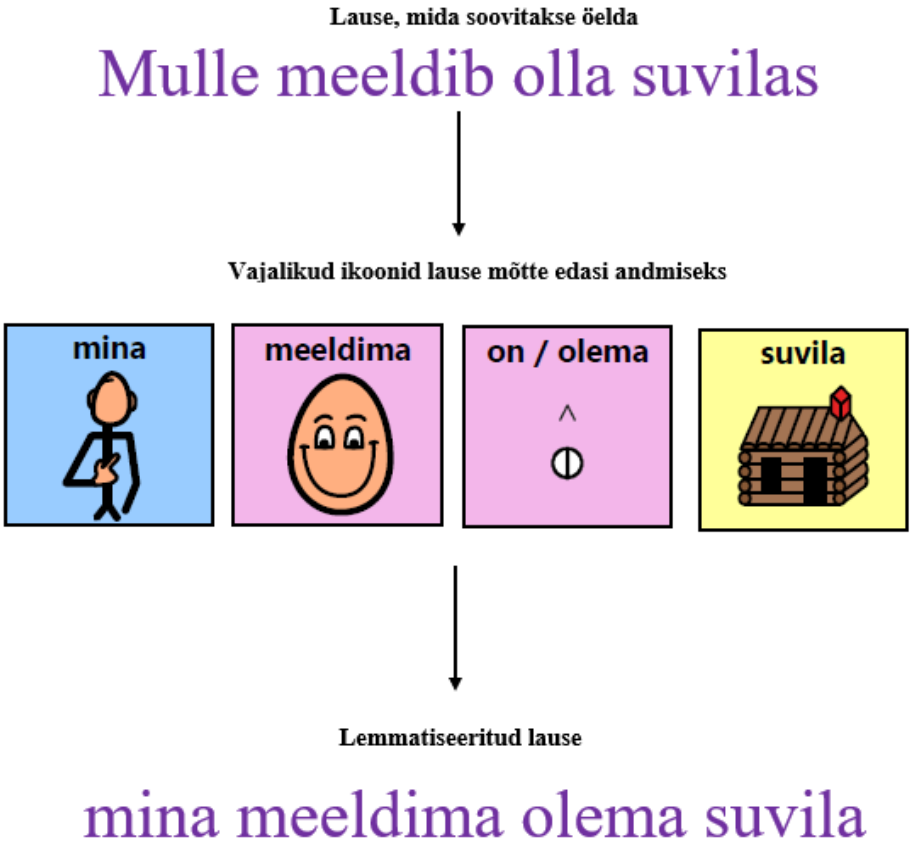
Epohhide õige arvu määramine mõjutab oluliselt mudeli täpsust. Liiga väikese arvu valimise korral on tulemuseks alasobitumine ning mudel ei võta treenimisel kõiki sisendandmeid piisavalt arvesse. Liiga suure arvu korral tekib ülesobitumine, kus mudel on õppinud treeningandmete järgi liiga täpselt. Ülesobitumise vältimiseks kasutatakse varajase lõpetamise (*early stopping*) metoodikat [19], mis eeldab, et lisaks treeningandmetele antakse mudeli sisendiks ka valideerimisandmed. Varajane lõpetamine tähendab seda, et iga epohhi järel võrreldakse tulemust valideerimisandmetega ning kui mudeli täpsus ei ole teatud aja jooksul muutunud, lõpetatakse treenimine.

3 Andmestiku koostamine

Ikoonide valimisel põhineva suhtlusprogrammi loomisel eeldatakse, et igale ikoonile vastab vähemalt üks eestikeelne sõna algvormis. Antud töös on tehtud eeldus, et ikoonidega on võimalik esitada iga sõna, mis esineb Eesti Keele Instituudi koostatud eesti keele põhisõnavara sõnastikus (edaspidi baassõnavara). Töö autoril pole täpset informatsiooni, kui paljudele eestikeelsetele sõnadele on loodud vastav ikoon. Kuna osadele sõnadele 5000 sõnast koosnevast põhisõnavara sõnastikust on juba loodud ikoon [2, Joonis 6], siis eeldatakse, et ka tulevikus loodavatele ikoonidele vastavad sõnad esinevad põhisõnavara sõnastikus.

Joonis 6. Ikoon sõnale „hunt“ [2].

Eelpool kirjeldatud suhtlusprogrammi arendamisel on oluline andmestikku, mis koosneb ainult lemmadest. Samuti on oluline, et kõik sõnad, mis on treeningkorpuses, oleksid olemas ikoonidena. Järgnev skeem [20, Joonis 7] illustreerib korpuse loomise sisulisi nõudeid.



Joonis 7. Mõtte väljendamise protsess ikoonide valimise meetodil [20].

Eelneva joonise põhjal on andmestiku loomise nõudeks, et sõnad „mina“, „meeldima“, „olema“ ja „suvila“ esineksid kõik baassõnavaras ehk oleksid esitatavad ikoonidena. Sellest lähtuvalt on koostatud ainult lemmadest koosnev andmestik ning arvutiprogramm filtreerimaks välja lauseid, mille iga sõna esineb baassõnavaras.

3.1 Eesti keele põhisõnavara sõnastik

Eesti keele põhisõnavara sõnastik [2] on Eesti Keele Instituudi koostatud sõnastik, milles on 5000 eesti keele olulisemat sõna. Sõnastik koostati aastatel 2010-2013 ning on ID-kaardiga sisenedes internetist alla laaditav. Antud lõputöös on eeldatud, et igale sõnastikus leiduvale sõnale on olemas ikoon ja sellest lähtuvalt on seatud eesmärgiks leida parim keelekorpus, mis oleks vastavuses põhisõnavara sõnastikuga (baassõnavara).

3.2 Andmestiku koostamine

Teadaolevalt pole loodud eestikeelse ikoonidel põhineva suhtlusmeetodi jaoks treeningandmestikku. Antud töös ei käsitleta ikoone ning nende valimise protsessi. Küll aga on eeldatud, et igale ikoonile vastab üks eestikeelne sõna algvormis ning et programmi edasi arendades saaks kasutaja ise ikoonile lisada mitmust ning ajavormi. Sellest tulenevalt on seatud eesmärgiks koostada lemmadest ehk sõna algvormidest koosnev korpus, mille abil oleks võimalik keelemudelit treenida ning hinnata.

3.2.1 Andmestiku koostamine

ETenTen keelekorpus [1] koosneb eestikeelsetelt veebilehtedelt kogutud lausetest. Korpus kuulub TenTen korpuste perekonda, mis on kättesaadav üle 30 keeles ning sisaldab kokku üle 10 miljardi sõna. Eestikeelse korpuse loomine toimus 2013. aasta jaanuaris ning koosneb 260 miljonist sõnast. Veebilehtede sisu on salvestatud .ela laiendiga faili ning need on omakorda grupeeritud 99 kausta. ETenTen korpus on antud lõputöö andmestiku loomise aluseks.

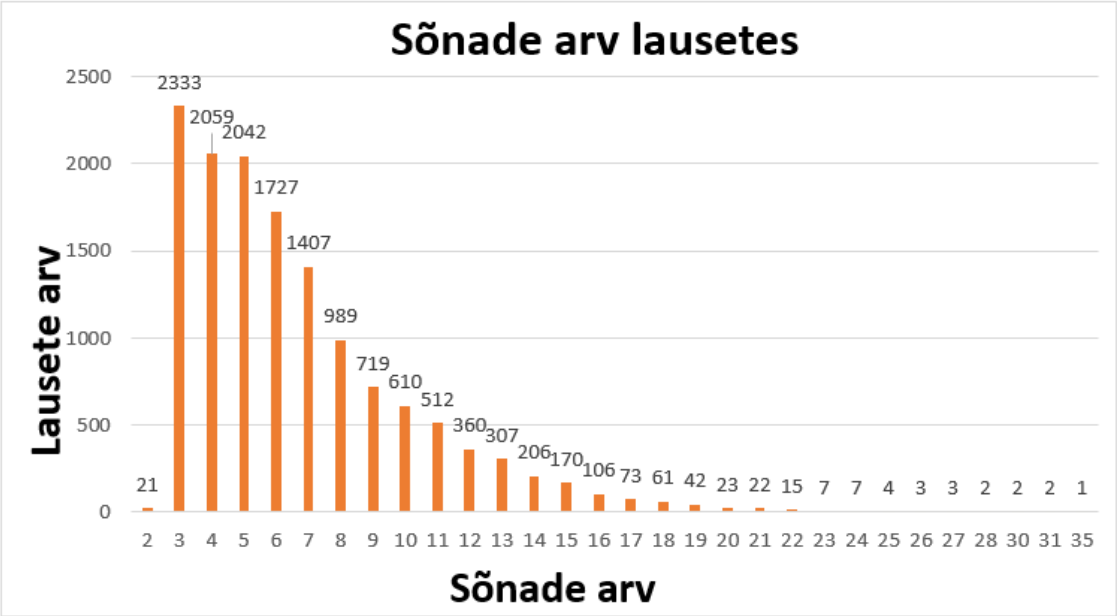
3.2.2 Lemmatiseeritud keelekorpuse loomine

Lemmadest koosneva andmestiku loomiseks võeti aluseks ETenTen keelekorpus ning kirjutati Pythoni arvutiprogramm. Lausete eraldamiseks Ela laiendiga failidest loodi vastav funktsioon, milles loodi EstNLTK lauseobjekt ning lemmatiseerija teegi abil genereeriti igast lause sõnast lemma. Lemmatiseerija võimaldab valida ühe sõna puhul mitme algvormi vahel. Näiteks sõna „lood“ lemma võib olla nii „lood“, mis tähendab vee sügavuse mõõtmise riista, kui ka „lugu“, mille puhul viitab sõna „lood“ algvormi mitmusele. Antud töös valiti alati viimane võimalik variant. Nii algsed laused kui lemmatiseeritud laused kirjutati mõlemad eraldi tekstifaili, iga lause eraldi real. Laused sisaldavad kirjavahemärke.

Eksperimendi alguses otsustati kasutada esmalt umbes 10% kogu olemasolevast lausetekogust. See andmestik sisaldab endas 466 893 lauset.

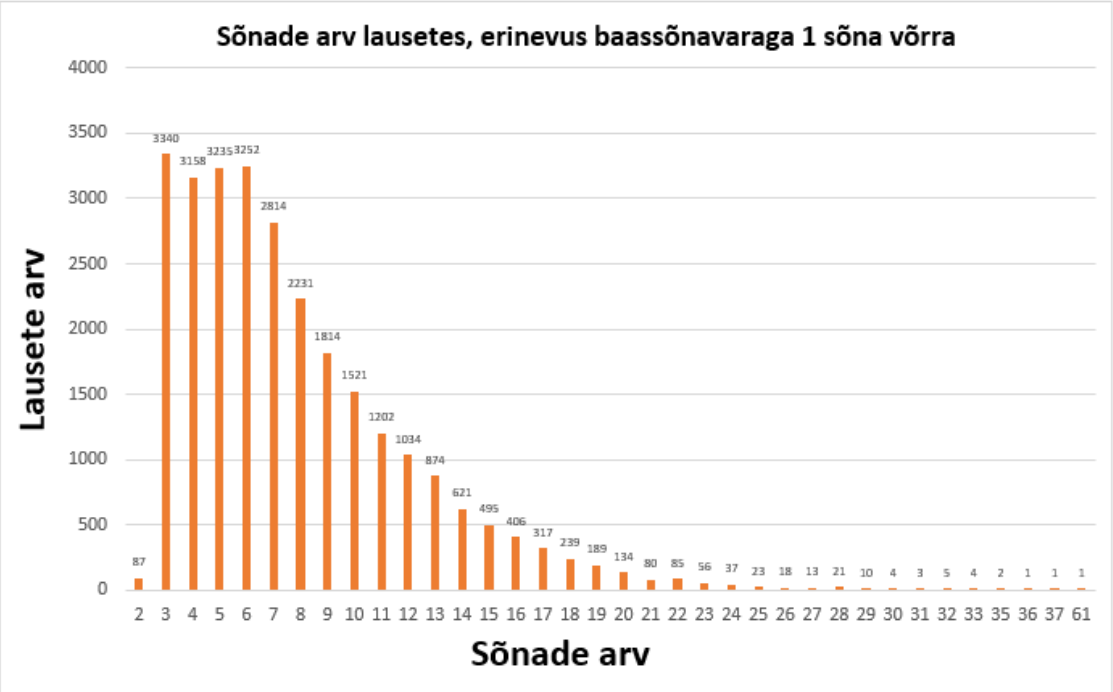
10% lausetekogust lemmatiseeriti ning otsiti välja välja sellised laused, mille kõik sõnad sisalduvad baassõnavaras ning mida saaks seetõttu AAC programmi loomisel kasutada. Selgus, et tingimustele vastavaid lauseid on ligi 3% ehk 13 835 ning need kirjutati uude faili. Arvesse võeti vaid lauseid, mis koosnevad vähemalt kahest sõnast. Enamik lauseid

koosnes kolmest või neljast sõnast ning kõige pikem lause koosneb kolmekümne viiest sõnast. Sõnade arvu jaotust lausetes kirjeldab järgnev Joonis 8.



Joonis 8. Ikoonidega asendatavate lausete arv ja nende pikkus sõnades.

Lisaks vaadati ka lauseid, milles on ainult üks sõna, mis puudub baassõnavaras. Eesti keele põhisõnavara sõnastik ei sisalda numbreid ega paljusid tihti esinevaid sõnu, näiteks „parim“, „pingutamine“ ja „veebileht“. Lisaks esinevad lemmatiseeritud andmestikus nimed, mis puuduvad samuti baassõnavaras. Edaspidisel arendusel tasuks uurida, kas andmestik oleks paremini kasutatav, kui nimed asendada teatud märgenditega või kui uuendada baassõnavara vastavalt inimeste keelekasutusele. Järgnev Joonis 9 kirjeldab lausete arvu, milles sisaldas ühte sõna, mis ei kuulunud baassõnastikku ning selliste lausete pikkust sõnades.



Joonis 9. Lausete, mille üks sõna pole asendatav ikooniga, arv ja nende pikkus sõnades.

3.2.3 Andmestiku täiendamine morfoloogiliste märgenditega

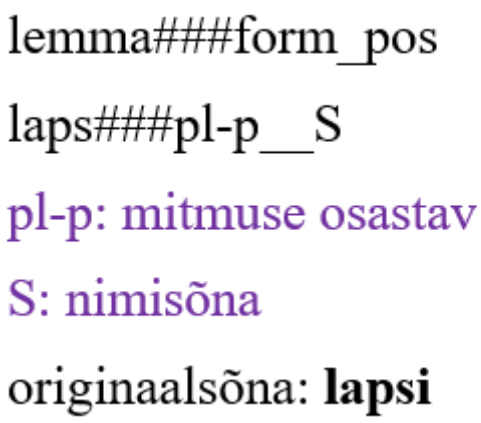
Pärast ikoonide valimist (lemmadest koosneva lause moodustamist) on tarvis neist genereerida grammatiliselt korrektne eestikeelne lause. Selleks on võimalik lisaks lemmale määrata algse sõna ajavormi, käänet, pööret jm täpsustusi. Nendest andmetest moodustub igale lemmale lisainformatsioon, mida nimetatakse töös edaspidi morfoloogiliseks märgendiks . Andmestikuga tehtavate eksperimentide eesmärgiks seati uurida, kui suure täpsusega (*accuracy*) on võimalik ennustada lemmadele märgendeid.

Eestikeelsete sõnade töötlemiseks loodud teekide kogumik EstNLTK pakub võimalust leida sõnale tema lemma, *form* (käänded, pöörded, kõneviisid, eitused jne) ja *pos* (sõnatüübid). Kuna mõlemad andmed on olulised, et hiljem sünteesida lemmadest grammatiliselt korrektse sõna, seati eesmärgiks ennustada mõlemat liidet ühe elemendina. Nendest koosnebki töös kasutatav morfoloogiline märgend. Järgnev Tabel 1 [21, Table 1] illustreerib lause „Usjas kaslane ründas künklikul maastikul tünjat Tallinnfilmi režissööri“ sõnadele genereeritud *form*-i ja *pos*-i märgendeid ja nende tähendusi.

Tabel 1. Lause „Usjas kaslane ründas künklikul maastikul tünjat Tallinnfilmi rešisööri“ sõnadele genereeritavad morfoloogilised märgendid *form* ja *posttag*. [21]

	word_texts	**forms**	**form description**	**postags**	**postag description**
0	Usjas	sg n	ainsus nimetav (nominatiiv)	A	omadussõna algvõrre
1	kaslane	sg n	ainsus nimetav (nominatiiv)	S	nimisõna
2	ründas	s	kindel kõneviis lihtminevik 3. isik ainsus aktiiv jaatav kõne	V	tegusõna
3	künklikul	sg ad	ainsus alalütlev (adessiiv)	A	omadussõna algvõrre
4	maastikul	sg ad	ainsus alaütlev (adessiiv)	S	nimisõna
5	tünjat	sg p	ainsus osastav (partitiiv)	A	omadussõna algvõrre
6	Tallinnfilmi	sg g	ainsus omastav (genitiiv)	H	pärisnimi
7	režisööri	sg p	ainsus osastav (partitiiv)	S	nimisõna

Lähtuvalt vajadusest teada iga algvormis sõna kohta nii tema lemmavormis sõna kui ka morfoloogilist märgendit, töödeldi andmestikku selliselt, et igale lemmale oleks lisatud vastav märgend eraldustähise „###“ abil. Uue andmestiku loomiseks koostati programm, kus form-is asendati tühikud sidekriipsuga ning liideti see pos-liitega. Tulemuseks saadi andmestik, kus iga lause on eraldi real, iga lemma lauses on oma liidetega teistest sõnadest eraldatud. Järgnev Joonis 10 illustreerib seda, kuidas lemmad ja neile liidetud morfoloogilised märgendid esinevad treeningkorpuses.



Joonis 10. Lemmade ja morfoloogiliste märgendite esituse arhitektuur andmestikus.

3.2.4 Treeningandmestik, valideerimisandmestik ja testandmestik

Enne mudeli treenimist on raske oletada, kui kaua ja milliste parameetritega peab mudelit treenima, et hiljem oleksid ennustused võimalikult täpsed. Üks võimalus on pärast mudeli treenimist võrrelda ennustustulemusi tegelikkusega ning teha selle põhjal järeldused [22]. Selline lähenemine, mis teeb ennustusi otsingutabeli põhimõttel, ennustab treeningu jooksul nähtud andmeid 100% õigesti, kuid ei suuda klassifitseerida uusi sisendandmeid.

Paremaks ennustustäpsuse mõõtmise lahenduseks on jagada treeningandmestik kahte osasse, suhtega 20% ja 80%, treeningandmestikuks ning valideerimisandmestikuks. Nii on võimalik treenimise käigus arvutada mudeli täpsust nii treeningandmestikus leiduvate sõnade kohta kui ka uute sõnade kohta. Andmete osadeks jagamisel tuleb tähele panna andmete olemust: üldjuhul on soovitatav koguda 20% andmeid kogu andmehulga eri osadest, et vältida ennustamist ainult üht tüüpi andmetele. Mõnel juhul muudab see aga mudeli ennustustäpsuse petlikult heaks, kus ennustamine nii treening- kui valideerimisandmestikule on väga hea, kuid uute sisendandmete korral ei anna mudel täpseid tulemusi.

Veel on olemas kolmas võimalus, kus andmestik jagatakse kolmeks: treening-, valideerimis ning testandmestikuks. Sellist meetodit kasutatakse erinevate mudelite võrdlemiseks ja vältimaks mudeli parameetrite korrigeerimist vastavalt valideerimisandmestikule, mis ei pruugi kajastada mudeli tegelikke sisendandmeid. Eesmärgiks on hinnata mudeli õigsust valideerimisandmestiku järgi, kuid hiljem hinnata

mudelit seni nägemata testandmestiku põhjal. Nii on võimalik hinnata mudeli käitumist mudeli reaalse kasutamise ajal.

Lõputöös kasutati treenimiseks esiteks väikest andmehulka, et võimalikud tekkivad erinevused tuleksid selgelt esile. Kokku oli kasutada 13 835 lausest koosnevat andmestikku, mille iga sõna sisaldub eesti keele põhivarasõnastikus. Valideerimiseks ja testimiseks valiti kummakski ligikaudu 10% andmestikust ehk 1400 lauset. Treenimiseks jäi ligikaudu 11 000 lauset.

Pärast tulemuste võrdlust kõikide vektoresituste vahel otsustati andmehulka suurendada 252 343 lauseni. Valideerimiseks ja testimiseks valiti lausete arv sama suhte alusel: 80% treeningandmeid, 10% valideerimisandmeid ja 10% testandmeid.

3.2.5 Vektoresituste ettevalmistus

Andmestikuga läbi viidavate eksperimentide käigus initsialiseeriti morfoloogilisi märgendeid ennustavat neuromudelit mitme vektoresitusega: juhuslike vektoritega, eestikeelse andmestiku põhjal eeltreenitud FastText [5] vektoritega ning lemmade põhjal treenitud FastText vektoritega.

Eeltreenitud FastText vektorid [23] on treenitud tavaliste eestikeelsete lausete alusel, mitte lemmade põhjal. On tõsi, et ka lemmad on sellises andmestikus esindatud, kuid nende kontekst erineb antud töös kasutatavast lemmatiseeritud andmestikust. Selle tõttu ei pruugi grammatiliselt korrektsete lausete põhjal treenitud vektorid sobida kõneabiprogrammi andmestiku kasutamiseks.

Eeltreenitud vektorid on internetist alla laaditavad .vec laiendiga failina [23]. Lisaks on olemas .bin laiendiga fail, mille abil on võimalik luua vektorid sõnadele, mis puuduvad algses eeltreenitud vektorite failis. Antud lõputöö treeningandmestiku analüüsimiseks loodi Pythoni programm. Selle eesmärgiks on leida treeningandmestikust sõnad, millel ei ole eeltreenitud vektorit. Neile sõnadele genereeritakse .bin faili abil vektorid ning lisatakse teiste eeltreenitud vektorite juurde.

FastTexti sõnavektorite treenimiseks lemmade alusel lemmatiseeriti kogu ETenTen korpus.

3.2.6 Juhuslikult initsialiseeritud sõnavektorid

AllenNLP *tagger* mudeli treenimiseks on võimalik eeltreenitud mudelite puudumisel kasutada klassi BasicTextFieldEmbedder,[1] mis võtab argumendiks kujutise (*dictionary*) treenimiseks kasutatavatest sõnadest ning loob sõnade arvu põhjal neile juhuslikud vektorid. Vektorite dimensiooni on võimalik vastavalt vajadusele määrata. Kuna järgmisena katsetatava FastText vektorite dimensioon on 300, otsustati ka juhuslike vektorite puhul kasutada sama dimensiooni. Vektorite väärtused ei jää juhuslikuks, vaid nende väärtusi uuendatakse pidevalt mudeli treenimise käigus.

[1] https://allennlp.org/tutorials

4 Eksperimendid andmestikuga

AAC-tüüpi kõneabiprogrammi loomisel on tähtis, et kui kasutaja valib ikoonid, mis kirjeldavad kõige paremini tema mõtteid ja soove, siis et programm suudaks võimalikult täpselt ennustada, mida kasutaja tegelikult öelda soovis. Bakalaureusetöö teise osa eesmärgiks on saada teada, kas ja milliseid sõnavektoreid kasutades saadakse lemmadele morfoloogilise märgendi ennustamisel kõige paremad tulemused. Eesmärgi saavutamiseks on võetud kasutusele AllenNLP näidismudel [24] märgendite ennustamiseks ning seda kasutatakse juhuslikult initsialiseeritud, eeltreenitud ja lemmade põhjal treenitud vektoresituste võrdlemiseks.

Mudeli ennustuse täpsus sõltub kindlasti keelemudeli parameetritest, treenimise ajast, aga suures osas ka sõnavektorite valikust. Sõnavektorite sobivus sõltub sellest, kui suure andmehulga põhjal on neid treenitud ehk kui täpselt kajastavad vektorid sõnade omavahelisi seoseid. Samuti on võimalik vektoreid treenida, arvestades iga sõna ja selle lähiümbrust, arvestades kogu lauset ning võttes arvesse ka tähtede järjekorda sõna sees.

Töö raames loodud andmestikus leiduvate sõnade programmiliseks tuvastamiseks on mitu erinevat viisi, mida antud töös uuritakse ning võrreldakse. Järgnevad alapeatükid annavad ülevaate jadamärgendamise mudeli olemusest, vaadeldavatest vektoresitustest, töö raames tehtud eksperimentidest ning saadud tulemustest.

4.1 Mudel

Programmiline kood lõputöös kasutatava mudeli treenimiseks ja selle põhjal lemmadele morfoloogiliste märgendite ennustamiseks on loodud AllenNLP *tagger* mooduli funktsionaalsuse näitel [24].

AllenNLP on teekide kogum, mille on loonud Allen Insitute for Artificial Intelligence [25]. AllenNLP kasutab Pythoni raamistikku PyTorch ning võimaldab treenida tehisnärvivõrkudel põhinevaid mudeleid, saades sisendiks eeltreenitud sõnavektorid.

AllenNLP platvorm on loodud mugavaks kasutamiseks loomuliku keele töötluse projektides. Selle abiga on lihtne luua andmeid kirjeldavaid graafikuid, jagada andmeid plokkideks (*batch*) ja lisada täidet (*padding*).

AllenNLP klass *PosDatasetReader* loeb etteantud failist laused, mis koosnevad lemmavormis sõnadest, millele on juurde lisatud vastav morfoloogiline märgend. Sõnade ja märgendite jaoks moodustatakse vastavalt väljad „*sentence*“ ja „*labels*“ ning igale sõnale määratakse unikaalne indeks.

Klass *LstmTagger* saab argumendiks sõnavektorid, koodri, mis antud töös on *lstmkooder*, ja treeningandmete põhjal koostatud sõnastiku. Klassil on olemas funktsioon *forward*, kus toimub põhiline mudeli treenimine. Andmeid loetakse sisse plokkidena, mille suuruseks on antud töös määratud 64. Kuna kõik laused ei ole plokis ühesuguse suurusega, siis teostavad AllenNLP funktsioonid täitmist, et täita tekkivad tühimikud. Saadud vektorid antakse edasi LSTM-koodrile ning lisatakse morfoloogilised märgendid. Kuna märgendite olemasolu on valikuline, siis juhul, kui need on olemas, arvutatakse nende põhjal kadu ning täpsus ja uuendatakse vastavaid muutujaid. Iga epohhi lõpus trükitakse konsooli info täpsusest ning kaost.

Mudeli optimeerimiseks kasutati Adam algoritmi [26], mille puhul uuendatakse treenimise käigus õpisammu. Adam on väga efektiivne just loomuliku keele töötlemise ja tehisnägemise ülesannete lahendamisel.

Klassile *Trainer* saab anda mitu parameetrit, enamasti on nendeks *model*, *optimizer*, *iterator*, *train_dataset*, *validation_dataset*, *patience*, *num_epochs*, *cuda_device*. Patience väärtus määrab ära epohhide arvu, mis treenitakse pärast parima tulemusega epohhi treenimist. Antud töös määrati epohhide arvuks 1000 ning patience väärtuseks 40. Lisaks on antud töös määratud parameeter *serialization_dir*, mis määrab ära tee kausta, kuhu salvestatakse pärast igat epohhi treeningu olek ning meetrika. Treeningu käigus salvestatakse parim mudeli tulemus faili „best.th“ ning lisaks treenimise lõpus saadud mudel nimega „model.th“.

4.2 Eksperimendid

Lõputöö jaoks tehtud eksperimentide eesmärgiks oli selgitada välja, kas kõneabiprogrammi loomiseks on mõistlik kasutada lemmatiseeritud andmestikku ning kui jah, siis millise vektoresituse kasutamine annab lemmadele morfoloogilist märgendite ennustamisel kõige täpsema tulemuse. Eesmärgi saavutamiseks treeniti AllenNLP jadamärgenduse mudelit, kasutades erinevaid sõnavektoreid. Nendeks on juhuslikult genereeritud sõnavektorid, eeltreenitud FastText sõnavektorid ning lemmade põhjal ise treenitud FastText sõnavektorid.

Tulemustest selgus esiteks, et 13 835 lausest koosnev treeningandmestik on liiga väike, saavutades vaid 63% suuruse täpsuse lemmadele morfoloogiliste märgendite ennustamisel. Vektoresituste vahel suuri erinevusi polnud. Nii juhuslikud kui ka eeltreenitud FastText vektorid saavutasid 63% suuruse tulemuse, kuid lemmadel treenitud vektoresitus vaid 58%.

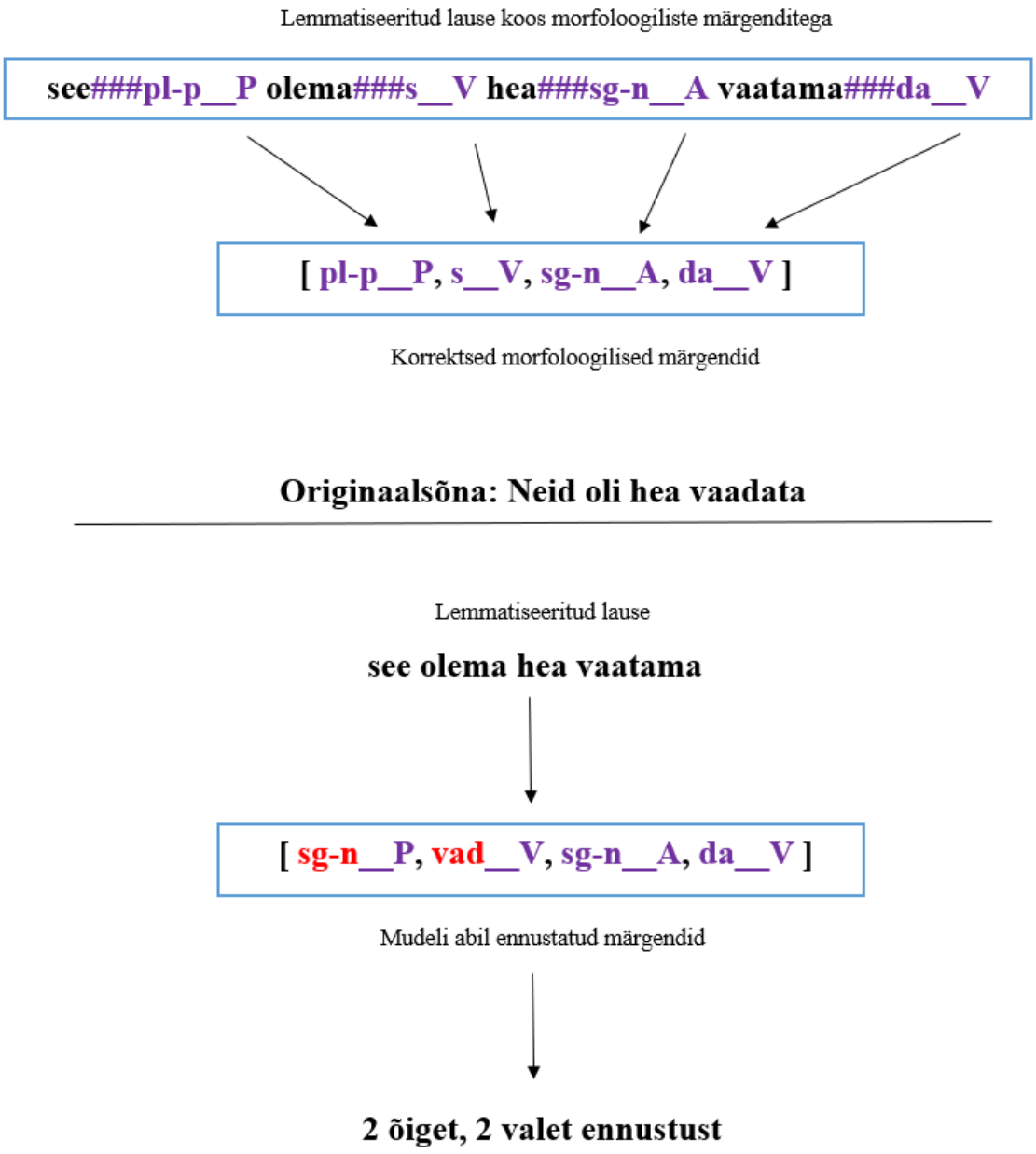
Andmemahu suurendamisel 252 343 lauseni tulemusel küll paranesid, kuid jäid kõik 70% lähedale.

Paremate tulemuste saavutamiseks tasub andmemahtu veelgi suurendada, kuid nii mudelikasutust kui ka andmestiku loomist ja töötlemist on võimalik muuta täpsemaks ja antud ülesande lahendamiseks paremini kohandada.

4.2.1 Hindamine

Mudeli hindamiseks on funktsioon *predict_endings*, mis võtab argumendiks lemmatiseeritud lause ning väljundiks on vastavatele sõnadele ennustatud morfoloogilised märgendid. Lemmavormist sõnadele algvormi tuletamine ei kuulu antud töö ulatusse. Programm loeb lemmadest ja nendele vastavtest märgenditest koosneva testfaili reahaaval mällu ning eraldab lemmadest märgendid. Lemmadest moodustatakse lause, millele ennustatakse parima treenitud mudeliga morfoloogilisi märgendeid. Saadud tulemusi võrreldakse tegelike märgenditega ning arvutatakse mudelile tegelik täpsus õigete ennustuste arvu ja kogu ennustuste arvu suhtega. Ennustamise teeb keeruliseks see, et ennustatav märgend koosneb sisulisest kahest komponendist, *form* ja *pos*. Tulemustest selgus, et paljudel juhtudel suudab mudel ennustada õigesti ühe poole märgendist, kuid kuna teine pool on vale, läheb see ennustus kirja kui vale tulemus. Järgnev Joonis 11

illustreerib olukorda, kus mudel ennustab sõnadele eelnevalt kirjeldatud olukorra sarnaselt osaliselt õigeid morfoloogilisi märgendeid.



Joonis 11. Lemmatiseeritud lausele morfoloogilistele märgendite ennustamise protsess.

4.2.2 Tulemused

Eksperimentide läbi viimiseks valiti esmalt andmestiku suuruseks 13 835 lauset. Töö alguses puudus täpne teadmine, kui suur andmehulk on antud ülesande lahendamiseks

sobiv. Mida väiksem on andmehulk, seda kiiremini toimub mudeli treenimise protsess ning kui mõne vektoresituse kasutamine peaks mõjutama treenimise tulemust rohkem kui mõne teise vektoresituse kasutamine, siis tulevad erinevused väikese andmehulga puhul paremini esile. Mudeleid treeniti i7 generatsiooni kahetuumalise CPU peal.

Tulemustest selgus aga, et 13 835 lausest koosnev treeningandmestik on liiga väike. Seda järeldati madalast täpsusest, milleks saadi 63%, lemmadele morfoloogiliste märgendite ennustamisel.

Veel selgus, et ühe vektoresituse kasutamine ei erine oluliselt teiste vektorite kasutamisest. Nii juhuslikud kui ka eeltreenitud FastText vektorid saavutasid 63% suuruse tulemuse, kuid lemmadel treenitud vektoresitus vaid 58%. Järgnev Tabel 2 kirjeldab iga vektoresituse valimise puhul mudeli täpsust.

Tabel 2. Mudeli ennustuste täpsused, kasutades erinevaid vektoresitusi ja 13 835 lausest koosnevat andmestikku.

Vektoresitus	**Täpsus testimisel**
Juhuslikud vektorid	62.16%
FastText eeltreenitud	62.46%
FastText treenitud lemmade põhjal	62.40%

Mudeli täpsust arvutati valideerimisandmestiku täpsuse põhjal. Kuigi epohhide arvuks oli iga mudeli treenimisel määratud 1000, saavutati parim tulemus juba esimestel epohhidel. Järgnev Tabel 3 kirjeldab treenimisel saavutatud tulemusi ning mitmendal epohhil saadi parim tulemus.

Tabel 3. Mudeli parima tulemuse meetrika, kasutades 13 835 lausest koosnevat andmestikku.

	Parima tulemusega epohh	**Treenimise täpsus**	**Treenimise kadu**	**Valideerimistäpsus**	**Valideerimiskadu**	**Treenimise aeg**
Juhuslikult initsialiseeritud vektorid	6	0.6871	1.0118	0.6493	1.2659	1 min, 50 s
Eeltreenitud FastText vektorid	4	0.6698	1.0901	0.6542	1.2313	1 min, 17 s
Ise lemmade põhjal treenitud FastText vektorid	4	0.6828	1.0319	0.6554	1.2070	1 min, 15 s

Andmemahu suurendamisel 252 343 lauseni tulemusel küll paranesid, kuid kõikide mudelite tulemused jäid 70% lähedale. Järgnev Tabel 4 kirjeldab suurema andmemahu puhul erinevate vektoresituste valimisel mudelite täpsust.

Tabel 4. Mudeli ennustuste täpsused, kasutades erinevaid vektoresitusi ja 252 343 lausest koosnevat andmestikku.

Vektoresitus	**Täpsus testimisel**
Juhuslikud vektorid	69.79%
FastText eeltreenitud	70.27%
FastText treenitud lemmade põhjal	70.15%

Järgnev Tabel 5 kirjeldab suurema andmemahu kasutamisel iga mudeli parima tulemuse mõõdikute väärtusi.

Tabel 5. Mudeli parima tulemuse meetrika, kasutades 252 343 lausest koosnevat andmestikku.

	Parima tulemusega epohh	**Treenimise täpsus**	**Treenimise kadu**	**Valideerimistäpsus**	**Valideerimiskadu**	**Treenimise aeg**
Juhuslikult initsialiseeritud vektorid	2	0.7112	0.9076	0.7009	0.9516	37 min, 0 s
Eeltreenitud FastText vektorid	4	0.7279	0.8433	0.6990	0.9915	56 min, 23 s
Ise lemmade põhjal treenitud FastText vektorid	4	0.7315	0.8282	0.7017	0.9810	56 min, 58 s

4.2.3 Tulemuste analüüs

Andmemahu suurendamine ning eksperimentide kordamine näitas, et suurema hulga andmete korral suureneb jadamärgendamise mudeli täpsus ennustamaks lemmadele morfoloogilisi märgendeid. Siiski ei saa 70% suurust täpsust pidada väga heaks ning järgnevalt on esitatud võimalused, mida tasub projekti edasisel arendamisel katsetada, et kogu programmi täpsust parandada.

Kõneabiprogrammi täpsuse parandamise võimalused saab jagada kaheks: andmestiku loomise protsessi parandamine ning morfoloogiliste märgendite ennustamise parandamine.

Nagu peatükis 2.1 mainitud, mõjutab kõneabiprogrammi kasulikkust baassõnade valik. Selle töö käigus uuriti vaid ühte keelekorpust, mis koosneb eestikeelsetelt veebisaitidelt kogutud lausetest, kuid võimalusel tasub uurida ka alternatiivseid keelekorpuseid. Samuti

tasub uurida, kas kõik sõnad, mida läheb programmi kasutajatel tarvis, sisalduvad rakenduses kasutatavas baassõnavaras ning kui paljudel neist on olemas ikoon.

Andmestiku loomise tehnilisest küljest tasub uurida, kas tulemused paraneksid, kui luua mõnede sõnade tähistamiseks spetsiaalne märgend. Näiteks võib andmestikus asendada numbrid ja nimed mõne tähisega, et suurendada nii nende lausete arvu, mille kõik sõnad esinevad baassõnavaras. Lisaks peaks võimalusel andmestikku suurendama ja proovima, kas programmi väljund oleks parem, kui andmestikust eemaldada lühikesed laused, näiteks kolme- ja neljasõnalised laused. Kuna antud töös kasutatav andmestik sisaldab kirjavahemärke, ei pruugi mõnesõnalised laused kanda sisulist mõtet, näiteks lause „ „No ja… ?“. Ka Dudy ja Bedricki [8] tööst selgus, et paremaid tulemusi andis andmestik, mille kõik sõnad ei sisaldunud ikoonandmestikus ning millel puudusid ka sõnavektorid. Sellised sõnad aitavad anda edasi lause sisulist mõtet ning hoida korrektset sõnade järjekorda.

Morfoloogiliste märgendite ennustamiseks saab treenida ka teistsuguse närvivõrgu kui on seda töös kasutatud AllenNLP *tagger*. Autori hinnangul kulus AllenNLP mudeli treenimiseks küllaltki vähe aega, mida võib seletada andmete väikese mahuga või muude mudeli omadustega. Samuti tasub uurida, miks saadi treenimisel parim tulemus juba esimestel epohhidel.

Kuigi nii antud töö tulemuste järeldustena kui ka Dudy ja Bedricki [8] tööst järeldades ei mõjuta sõnavektorite valik oluliselt treenimise tulemust, on soovitatav teha rohkem katsetusi lemmadel treenitud sõnavektoritega. Lemmadel treenitud vektoresituse kasutamisel saadud madalam tulemus võrreldes teiste vektoresituste kasutamisega võib tuleneda sellest, et kuigi lemmatiseeritud andmestik on antud probleemi lahendamiseks sobivam, on andmestik mahult liiga väike.

5 Kokkuvõte

Bakalaureusetöö käigus koostati kolme tüüpi andmestikku: kirjakeelsetest lausetest koosnev andmestik, lemmatiseeritud andmestik ning lemmatiseeritud andmestik koos morfoloogiliste märgenditega. Lemmatiseeritud andmestikku kasutati uute FastText [5] sõnavektorite treenimiseks ja lemmatiseeritud andmestikku koos morfoloogiliste kasutati mudeli treenimiseks, et ennustada lemmavormis sõnadele märgendeid. Tulemuseks saadi 23 046 560 lausest koosnev lemmatiseeritud andmestik.

Ikoonide valimisel põhineva kõneabiprogrammi koostamisel on oluline, et kõik sõnad, mis leiduvad kasutatavas andmestikus, on esindatud ka baassõnavaras ehk oleksid asendatavad ikoonidega. Töö autoril puudub täpne informatsioon selle kohta, kui paljudele eestikeelsete sõnade jaoks on loodud ikoon. Seetõttu tehti eeldus, et kõik sõnad, mis võiksid edaspidi ikoonidena esitatavad olla, leiduvad eesti keele põhisõnavara sõnastikus (baassõnavara). Sellest lähtuvalt võeti esmalt eksperimentide läbiviimiseks kasutusele 13 835 lauset, mille iga sõna esines baassõnavaras. Hiljem suurendati baassõnavarast koosnevate lausete hulka 252 343 lauseni.

Eksperimentide tulemustest selgus, et ligi 14 000 lausest koosnev andmestiku põhjal treenitud mudelite täpsuseks saadi umbes 63%. Sellest järeldati, et andmestik on liiga väike ning korrati samu eksperimente suurema andmestikuga. Tulemused paranesid, saades täpsuseks ligi 70%, kuid suurendades andmestikku veelgi on tõenäoliselt võimalik saada veelgi paremaid tulemusi.

Töös võrreldi kolme erinevat sõnavektorite esitust: juhuslikult initsialiseeritud vektoreid, FastText [5] eeltreenitud vektoreid ning ise lemmade põhjal treenitud FastText vektoreid. Vektorite kasutamise võrdlusest jadamärgenduse neuromudeli treenimiseks selgus, et vektoresitused erinesid üksteisest vähe. Väiksema andmestikuga mudeli treenimisel ja ise lemmade põhjal treenitud vektoresituse tulemus oli ülejäänud kahest madalam, olles ligikaudu 58%. Selle põhjuseks võib olla see, et kuigi lemmadel vektorite treenimine on antud ülesande jaoks sobivam lahendus, oli selleks kasutada liiga vähe andmeid.

Projekti edasi arendamisel tasub paremate tulemuste saavutamiseks suurenda andmemahtu. Lisaks tuleks teha katseid andmestikuga, kus on erilised sõnad, näiteks nimed ja numbrid, asendatud kindla märgendiga. Mõnesõnaliste lausete suure hulga tõttu võib mudeli täpsus jääda madalaks. Selle tõttu tasub vähendada andmestikus lühikeste lausete arvu.

Lisaks AllenNLP *tagger* mudeli treenimisele tuleks edaspidi teha katsetusi ka teiste morfoloogilisi märgendeid ennustavate mudelitega. Samuti on soovitatav treenida sõnavektoreid suurema lemmatiseeritud andmehulga põhjal ning katsetada lisaks ka teisi sõnavektoreid.

Teadaolevalt pole seni loodud spetsiaalselt ikoonide valimise põhimõttel töötavat eestikeelset andmestikku. Antud lõputöö käigus koostatud andmestikku on võimalik kasutada nii AAC programmide arendamiseks kui ka muude keeleõppe ja -tõlke programmide loomiseks.

Kasutatud kirjandus

[1] „etTenTen – Estonian corpus from the web“, *Sketch Engine*, 14-mai-2015. .

[2] „[PSV] Eesti keele põhisõnavara sõnastik“. [Online]. Saadaval: http://www.eki.ee/dict/psv/. [Vaadatud: 24-apr-2019].

[3] „What is AAC? | Communication Matters“. [Online]. Saadaval: https://www.communicationmatters.org.uk/page/what-is-aac. [Vaadatud: 31-märts-2019].

[4] J. Light *et al.*, „Challenges and opportunities in augmentative and alternative communication: Research and technology development to enhance communication and participation for individuals with complex communication needs“, *Augment. Altern. Commun.*, kd 35, nr 1, lk 1–12, jaan 2019.

[5] A. Joulin, E. Grave, P. Bojanowski, ja T. Mikolov, „Bag of Tricks for Efficient Text Classification“, *ArXiv160701759 Cs*, juuli 2016.

[6] J. H. McCarthy, I. Schwarz, ja M. Ashworth, „The availability and accessibility of basic concept vocabulary in AAC software: a preliminary study“, *Augment. Altern. Commun.*, kd 33, nr 3, lk 131–138, juuli 2017.

[7] B. A. Bracken ja E. Crawford, „Basic Concepts in Early Childhood Educational Standards: A 50-State Review“, *Early Child. Educ. J.*, kd 37, nr 5, lk 421–430, märts 2010.

[8] S. Dudy ja S. Bedrick, „Compositional Language Modeling for Icon-Based Augmentative and Alternative Communication“, *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, Melbourne, 2018, lk 25–32.

[9] D. Kondratyuk, T. Gavenčiak, M. Straka, ja J. Hajič, „LemmaTag: Jointly Tagging and Lemmatizing for Morphologically-Rich Languages with BRNNs“, *ArXiv180803703 Cs*, aug 2018.

[10] „Estnltk – Open source tools for Estonian natural language processing — estnltk 1.4.1 documentation“. [Online]. Saadaval: https://estnltk.github.io/estnltk/1.4.1/. [Vaadatud: 16-mai-2019].

[11] M. Gardner *et al.*, „AllenNLP: A Deep Semantic Natural Language Processing Platform“, *ArXiv180307640 Cs*, märts 2018.

[12] C. M. Bishop, „Neural Networks for Pattern Recognition“, lk 498.

[13] J. Slavio, *Deep Learning and Artificial Intelligence: A Beginner's Guide to Neural Networks and Deep Learning*, Kindle edition. 2017.

[14] A. Dertat, „Applied Deep Learning - Part 1: Artificial Neural Networks“, *Towards Data Science*, 08-aug-2017. [Online]. Saadaval: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6. [Vaadatud: 22-mai-2019].

[15] „Bidirectional recurrent neural networks - IEEE Journals & Magazine“. [Online]. Saadaval: https://ieeexplore.ieee.org/abstract/document/650093. [Vaadatud: 17-mai-2019].

[16] geva, „Classification with Neural Networks: Is it the Right Choice? - MissingLink“. [Online]. Saadaval: https://missinglink.ai/guides/neural-network-concepts/recurrent-neural-network-glossary-uses-types-basic-structure/. [Vaadatud: 22-mai-2019].

[17] M. Sundermeyer, R. Schluter, ja H. Ney, „LSTM Neural Networks for Language Modeling“, lk 4.

[18] T. Mikolov, K. Chen, G. Corrado, ja J. Dean, „Efficient Estimation of Word Representations in Vector Space“, *ArXiv13013781 Cs*, jaan 2013.

[19] R. Caruana, S. Lawrence, ja C. L. Giles, „Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping“, *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, ja V. Tresp, Toim MIT Press, 2001, lk 402–408.

[20] T. Dynavox ja A.-M. Rebane, „Suhtlustahvlinäidis kasutades Tobii Dynavox Communicator 5 programmi ikoone“. .

[21] „Tables — estnltk 1.2 documentation“. [Online]. Saadaval: https://estnltk.github.io/estnltk/1.2/tutorials/morf_tables.html#table-noun-form-descriptions. [Vaadatud: 12-mai-2019].

[22] Y. Goldberg, „Neural Network Methods for Natural Language Processing“, *Synth. Lect. Hum. Lang. Technol.*, kd 10, nr 1, lk 1–309, apr 2017.

[23] „Word vectors for 157 languages · fastText“. [Online]. Saadaval: https://fasttext.cc/index.html. [Vaadatud: 19-märts-2019].

[24] „AllenNLP - Tutorials“. [Online]. Saadaval: https://allennlp.org/tutorials. [Vaadatud: 12-mai-2019].

[25] M. Gardner *et al.*, „AllenNLP: A Deep Semantic Natural Language Processing Platform“, lk 5.

[26] D. P. Kingma ja J. Ba, „Adam: A Method for Stochastic Optimization“, *ArXiv14126980 Cs*, dets 2014.