TALLINNA ÜLIKOOL

Digitehnoloogiate instituut

Masinõppel rajaneva tarkvararakenduse loomine keeleoskustaseme ennustamiseks

Bakalaureusetöö

Autor: Janek Kossinski

Juhendaja: Jaagup Kippar, Pille Eslon

Autor: ..... ”.....” 2018

Juhendaja: ..... ”.....” 2018

Juhendaja: ..... ”.....” 2018

Instituudi direktor: ..... ”.....” 2018

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Sisukord

Sissejuhatus

Siinse töö eesmärk on luua rakendus, mis ennustab teksti eesti keele taset masinõpet kasutades. Rakendus peab töötama niisugusel kujul, et sellele saab teha HTTP päringuid koos ennustamistvajava tekstiga, mispeale rakendus tagastab teksti keeletaseme ennustuse.

Töö teema pakkus autorile juhendaja, huvi pakkus rakenduse loomine ning programmiline masinõppe kasutus. Kuna masinõpe on väga adapteeruv tehnoloogia, ei omanud teema valikul autori jaoks tähtsust, mida rakendus ennustama peab.

Töö teema on aktuaalne, sest masinõpet kasutavaid rakendusi eesti keele taseme ennustamiseks (avalikustatud) pole.

Eesmärgi saavutamiseks on kasutataud Java’t, WEKA’t, närvivõrkude mudelit ja mõningaid muid komponente, mida on töös täpsemalt kirjeldatud. Rakenduse versiooni haldus toimub github keskkonnas, vt https://github.com/janekos/masinoppe_ennustus

1 Eeluuring

Selles peatükis on juttu rakendustest, mis seotud keeletaseme tuvastamisega, k.a varem läbiviidud uuringud, mis seotud siinse töö teemaga, ning masinõppe mudelid, mida on mõistlik kasutada bag-of-words[1] tüüpi algandmete puhul, ja masinõppe teeki, mis on siinse töö raames arendatud rakenduses kasutusel.

1.1 Sarnane tarkvara

Järgnevates alapeatükkides loetletud rakendused näitavad, et keeletaseme tuvastamist programmiliselt on üritatud teostada ka varem. Nende rakenduste juures, kus pole töökäik lahti mõtestatud, pole autor suutnud tuvastada, kas tegu on masinõppega või mitte. Seda puuduva lähtekoodi või märkuse tõttu, mis viitaks masinõppe kasutusele. Sellistes rakendustes masinõppe kasutust on raske välistada, sest masinõpe ei ole uus tehnoloogia, mistõttu autor sellega seoses liigseid järeldusi ei teinud.

1.1.1 Elektroonilised testid

See on üks algelisemaid viise keeletaseme kontrolliks ning on võrdväärne kirjalike testidega, mida kasutatakse õppeasutustes, kuid ainukese vahega, et õppeasutustes kontrollitakse tulemust käsitsi.

Näitena võib tuua "languagelevel"[2] testimislehe, kus testitakse inglise keele taset. Selleks on tarvis vastata kõigest 15 valikvastusega küsimusele, mispeale veebileht tagastab ennustuse teksti keeletaseme kohta.

Kui uurida veebilehel oleva testirakenduse lähtekoodi, siis selgub, et tegu on kõige tavalisema kirjaliku testiga, mida saab kasutada elektroonilisel kujul. Taoliste rakenduste puhul on keeletaseme ennustamine üles ehitatud viisil, kus igale küsimusele on paika pandud õige vastus ning mida rohkem õigeid vastuseid on kasutajad valinud, seda kõrgem on kasutaja keeletase. Taolist tüüpi keerukamatel rakendustel võivad küsimuste vastused pigem viidata keeletasemetele endile, mistõttu lõpptulemus näitab iga keeletaseme protsendilist tõenäosust.

Selliste rakenduste tulemusi on raske usaldada, sest vastuste analüüs on väga algeline ning paika pandud rangete tingimustega. Samas on nende rakenduste tööviis samastatav mõne

[1] bag-of-words – algandmed, kus tekstide sees olevad sõnad on kujutatud numbriliste väärtustena.
[2] www.languagelevel.com/english/index.php

masinõppe mudeliga, nagu näiteks otsustuspuud, mistõttu võivad ka sellised rakendused olla päris head ennustajad, juhul kui algandmed on piisavalt mahukad.

1.1.2 Loetavuse kontrollijad

Loetavuse kontrollijad on rakendused, mis teatud algoritmide abil hindavad teksti loetavust. Selliste rakenduste peamine eesmärk ei ole teksti autori keeletaseme hindamine, kuid saadud tulemuste põhjal võib teha sellekohaseid järeldusi.

Näiteks readable.io [3] rakendus. Peale teksti sisestamist vastavasse välja ja kinnitusnupu vajutamist kuvatakse samale ekraanile hulk tulemusi, kus sisalduvad

- loetavuse astme tasemed, mis näitavad ligikaudu, mitu aastat teksti autor on haridust omandanud;
- loetavuse tulemused, mille hulgas on IELTS, Spache ja muud;
- teksti andmed nagu näiteks mitmes lauses on rohkem kui 30 silpi jne.

Paraku enamuse nende tulemuste abil saab teha vaid oletusi teksti autori keeletaseme kohta.

Teksti loetavust hindavatel lehtedel eelnevalt mainitud tulemuste esinemine on vahelduv, kuid readable.io'le omane tulemus on teksti CEFR[4] (Common European Framework of Reference) tase kuueastmelisel skaalal (A1, A2, B1, B2, C1, C2). Rakenduse kodulehel on kirjas, et CEFR keeleoskustaseme tulemus ei näita otseselt teksti autori keeletaset, vaid seda, mis keeleoskustasemega lugejatele sisestatud tekst sobiks (Readable.io, kuupäev puudub). Selle kohta, kuidas tulemus arvutatakse, infot kahjuks ei olnud.

1.2 Uurimuslikud tööd

Uurimuslike tööde arv keeletaseme ennustamise teemal on päris mahukas, kuid paljud neist ei tee kättesaadavaks uuringu tulemusel loodud rakendusi. Paljudes töödes käsitletakse pigem mõnda kindlat aspekti keeletaseme ennustamisel, mistõttu järgnevalt on loetletud need, mis on siinse tööga lähemalt seotud.

[3] https://readable.io/text/
[4] CEFR – rahvusvaheline standard keele oskuse kirjeldamiseks (Cambridge English, kuupäev puudub).

1.2.1 Eesti vahekeele korpuse klasteranalüüsi vahendite kasutamine teksti keeletaseme prognoosimisel

Enne käesolevat tööd on valminud Virgo Hallikul Eesti Vahekeele Korpuse (edaspidi EVKK) klasteranalüüsi tööriistu kasutav rakendus, mis aitab prognoosida kasutaja sisestatud teksti keeletaset (Hallik, 2016).

Töö alguses avab autor keelekorpuse mõiste ning loetleb Eestis loodud keelekorpuseid ja rakendusi, mis tuvastavad keeletaset. Töö põhiosas on kirjeldatud EVKK klastrileidja rakenduse kasutamise tulemused, kus V. Hallik seletab, missugused erinevatest sõnaliikidest koosnevad kolmikud (trigrammid) on eesti õppijakeeles levinumad ning milline on nende esinemissagedus A2-, B1-, B2- ja C1-tasemel. Töö lõpus kirjeldab autor loodud rakenduse töökäiku, mis järjestab sisendina antud teksti kolmikuid vastavalt keeletasemele ja teeb ennustuse.

Peamine järeldus oli, et loodud rakendus ei ole piisavalt täpne madala klastrite hulga tõttu (Hallik, 2016). Ehk teisitisõnu – teksti klasteranalüüsi puhul on tähtis valimi maht ja klastrite arv.

1.2.2 Kaidi Lõo ja Sowmya Vajjala uurimistööd

Tõenäoliselt kõige rohkem on siinse töö teemaga seotud Kaidi Lõo (2012) ning Kaidi Lõo ja Sowmya Vajjala (2013; 2014) uurimistööd, milles keeletaseme ennustamiseks on kasutatud masinõpet ning saadud tulemuse olid järkjärgult paremad. Autorid on kasutanud WEKA masinõppe teeki.

Lõo ja Vajjala tööde juures on esile toodud pigem masinõppe osa ehk see, kuidas olid algandmed ette valmistatud, mis masinõppe mudeleid kasutati ning missugused tulemused saadi.

Suhe sõnarikkuse ja eesti keele õppijate keeletaseme vahel

Uurimuse Lõo 2012 eesmärk oli teada saada, kui palju mõjutab eesti keele õppijate sõnavara vahelduvus ehk varieerumine (ingl *lexical variation*) keeletaseme ennustust.

Töö põhiosas on kirjeldatud tekstide töötlemist, kasutatud vabavaralist programmi TreeTagger, mis määrab tekstis iga sõnavormi sõnaliigi (ingl *part of speech*) ja seob vormid lemmaga (Schmid, kuupäev puudub). Peale selle oli iga teksti sõnavara rikkust kirjeldatud 15

mõõtme (ingl *lexical richness measure*) alusel ehk multidimensionaalselt. Edaspidi olid need andmed, mis saadud iga teksti kohta, kasutusel treeningandmetena valitud masinõppe mudelite jaoks, milleks olid K-nearest Neighbour, Decision Trees ja Neural Networks.

Katsed olid tehtud kõigi kolme mudeliga, mille tagajärjel ilmnes, et treeningandmete puhul olid kõige paremini toimivad masinõppe mudelid Neural Networks (täpsus 56.316%) ja RMSE (Root Mean Square Error[5] tulemusega 0.446). Töö autori arvates on need tulemused paljulubavad ja ilmselt oleksid täpsemad, kui algandmed oleksid ühtlased.

Morfosüntaktiliste dimensioonide roll eesti keele taseme klassifitseerimisel

Artiklis Lõo & Vajjala 2013 kirjeldatakse keeletaseme ennustamist nii sõnavara rikkuse (ingl *lexical richness*) kui ka sõnavormide vahelduvuse ehk varieerumise (ingl *morphological variation*) põhjal.

Erinevalt eelnevatest töödest on siin masinõppe mudelite ehitamisel kasutusel rohkem mõõtmeid ehk dimensioone (ingl *features*) ja on loobutud RMSE tulemuste vaatlemisest. Dimensioonide arvu suurendamiseks laiendati mõõdetavate üksuste hulka ning tehti öeldise, sihitise, teksti pikkuse ja sõnaliigi tunnuste alusel eraldi grupid. Seejärel eemaldati igast grupist CFS (Correlation based Feature Subset) abil madala ennustusvõimega ja üleliigseid dimensioone, jättes nelja grupi peale kokku alles ainult kümme mõõdet, mis andsid keeletaseme ennustamisel parima tulemuse.

Järgmisel etapil testiti nende kümne dimensiooni kombinatsioonide ennustusvõimekust SMO (Sequential Minimal Optimization), Logistic Regression, Decision Tree ning Stacking mudelite peal (sisaldab kolme eespool nimetatud mudelit), üritades leida nende seast paremaid tulemusi andvat mudelit ja mõõtmete kombinatsioone. Parimaks osutus Stacking mudel 63,28% täpsusega.

See tulemus ei olnud paraku rahuldav, mistõttu võtsid Lõo ja Vajjala (2013) kasutusele kahe klassi astmelise klassifitseerimise (ingl *two-class cascading classification*). Moodustati kuus binaarset klassifitseerijat, millest kolm olid kombinatsioonid keeletasemetest A, B ja C ning ülejäänud olid keeletasemed ning kirjed, mida antud keeletasemes ei olnud (näiteks A ja mitte

[5] RMSE – ennustusvigade standardhälve (Statistics How To, 2016).

A). Nende kombinatsioonide põhjal tehti kaks kaskaadi, kus ühes olid eelnevalt mainitud esimesed kolm klassifitseerijat ning teises teised kolm klassifitseerijat. Saadud tulemuste täpsus oli vastavalt 64,66% ja 66.66%. Tänu sellele saavutati ligikaudu 10% parem tulemus, millest omakorda sai kinnitust seisukoht, et eesti keele tasemete ennustamisel on oluline osa morfoloogilistel dimensioonidel.

Eesti keele õppijate keeletasemete automaatne CEFR taseme ennustamine

Artiklis Lõo & Vajjala 2014 on keeleoskustasemete ennustamise aluseks võetud CEFR kuueastmeline skaala (A1, A2, B1, B2, C1, C2), kasutatud on nii eespool kirjeldatud astmelist klassifitseerimist kui ka regressioonianalüüsi.

Erinevalt uurimuses Lõo & Vajjala 2013 kasutatud kolmeastmelisest keeleoskustasemete skaalast (A, B, C) oli 2014. aastaks loodud EVKK tuumkorpus, milles kõikide tekstide keeleoskustaset olid hinnanud kolm tunnustatud hindamisspetsialisti vastavalt CEFR kuueastmelisele skaalale. Kuna artikli kirjutamise hetkeks oli tuumkorpuses A1 ja C2 tasemega tekste kumbagi ainult üks, siis jäid need tasemed välja. Seejärel rakendasid Lõo ja Vajjala oma eelmises töös kasutatud dimensioonidele lisaks uusi mõõtmeid, mida oli kokku 78. Kohati oli muutunud ka tulemuste hindamismeetod, sest vaadeldi seost regressioonianalüüsi ja klassifikatsiooni kasutamisel saadud tulemuste vahel.

Kõigepealt testiti A2, B1, B2 ja C1 keeletaseme ennustamist SMO alusel. Tulemus tasakaalustamata andmete puhul oli 73.7% ning tasakaalustatud andmetega 72.3%. Kaheastmelise klassifitseerimise rakendamine andis paremaid tulemusi: tasakaalustamata andmete ennustamise täpsus oli 79% ja tasakaalustatud andmestiku puhul 76.9%. Klassifitseerimise testimisele järgnes regressiooni testimine. Selleks oli kasutatud Linear regression mudelit, mis määras keeleoskustaseme 76% täpsusega.

Lõpuks vaatlesid Lõo ja Vajjala, kuidas dimensioonide vähendamine mõjutab ennustamise tulemust. Selleks kasutasid nad kolme dimensiooni: nö valija mudelit (ingl *attribute selector*), milleks olid Information Gain (10 dimensiooni), CfsSubsetEval (27 dimensiooni) ja ReliefAttributeEval (10 dimensiooni). Tulemuste täpsus oli vastavalt 73,5%, 78,3% ja 74,5%. Need tulemused näitavad, et ennustamine väiksema dimensioonide arvu juures on võimalik, kuid muutub järjest täpsemaks, kui korrigeerida paremaid tulemusi andvaid dimensioone.

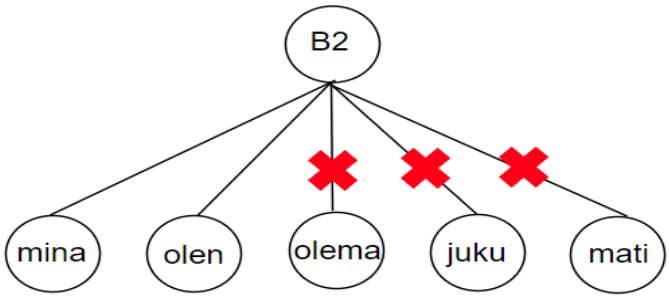
1.3 Töö tarbeks sobilikud masinõppe mudelid

Masinõppe tarbeks on loodud arvestatav hulk erinevaid mudeleid, mis enamjaolt töötavad spetsiifiliste andmestike raames ega pruugi piisavalt täpseid tulemusi anda, et teha ennustust bag-of-words tüüpi algandmeid kasutades. Seetõttu on algusest peale tarvis valida oma andmestiku kujule sobilikud mudelid.

Eelmises alampeatükis loetletud uurimuslikes töödes kasutusel olnud mudelite algandmete kuju on liiga erinev sellest, mida on kasutatud siinse töö raames loodud rakenduse jaoks. Seetõttu neid masinõppe mudeleid allpool ei loetleta, v.a närvivõrkude ja otsustuspuude mudelid, sest need on jätkuvalt sobilikud ka käesoleva uurimuse raames loodud rakenduses kasutatavate algandmete jaoks.

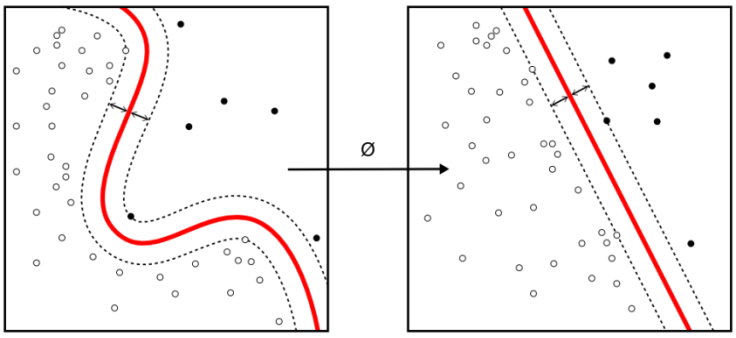
Siinse töö jaoks sobilikeks masinõppe mudeliteks on peamiselt need, millega on mõistlik töödelda bag-of-words tüüpi algandmeid.

- Naiivne Bayes (ingl *Naive Bayes*) – Bayes'i teoreemi kasutatav masinõppe mudel, kus kõige lihtsamal kujul ennustatav klass on juureks ning dimensioonid on lehtedeks (vt joonis 1). Mudeli naiivsus on tingitud sellest, et mudel eeldab, et kõik dimensioonid on üksteisest tingimuslikult sõltumatud (Russell & Norvig, 1995, lk 718).



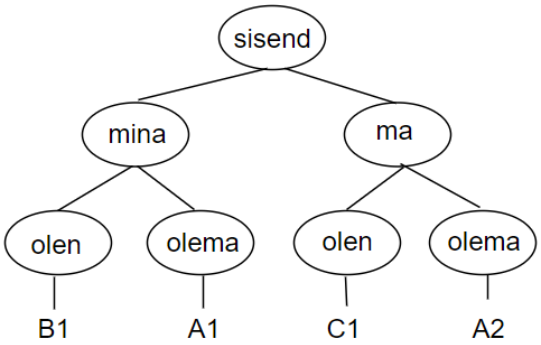
Joonis 1. Piltlik kujutus Naiivse Bayesi mudelist. Punase ristiga märgitud ühenduste puudumise tõttu oleks sisend klassifitseeritud B2 tasemeks.

- SVM (Support Vector Machine või Support Vector Networks) – masinõppe mudel, kus sisendvektorid on esialgu kaardistatud mittelineaarselt kõrge dimensioonide arvuga ruumi, kus edasipidi luuakse lineaarne otsustuspind (vt joonis 2), mille põhjalt tehtud üldistus on tüüpiliselt väga tõhus (Cortes & Vladimir, 1995).



Joonis 2. Mittelineaarsest kõrge dimensioode arvuga ruumist luuakse lineaarne otsustuspind[6].

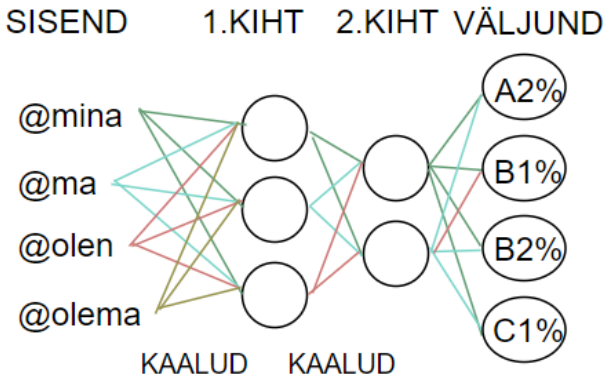
- Otsustuspuud (ingl *Decision Trees*) – masinõppe mudel, kus vastavalt sisendile tehakse mööda lahti harutatud tingimuslikke sõlmesid liikudes otsus selle kohta, mis klassi sisend kuuluda võiks. Seda mudelit piltlikult kujutades näeks see välja nagu juurestik (vt joonis 3) (Decision trees, kuupäev puudub).



Joonis 3. Otsustuspuu näide keeletaseme ennustamise kontekstis

- Närvivõrgud (ingl *Neural Networks*) – nn. „masinõppe šveitsi nuga“, sest selle mudeli jaoks sobilik andmestiku kuju on küllaltki lahtine mõiste (Andmeteadus, kuupäev puudub). Kõige lihtsam seletus närvivõrkude jaoks on arvutuslikud mudelid, mille eeskujuks on päris närvivõrgud (Frontiers, kuupäev puudub). Närvivõrkude mudel koosneb kihtidest (vt joonis 4), mille sees on sõlmed (neuronid/pertseptroonid), mis omakorda annavad väljundi vastavalt nendes paikapandud aktiveerimisfunktsioonidele (Karlik & Olgac, kuupäev puudub).

[6] https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:Kernel_Machine.png



Joonis 4. Näide närvivõrkude mudelist keeletaseme ennustuse kontekstis

Lisaks eelnevalt loetletud mudelitele leidub ka teisi, mis oleksid siinses töös kasutatavad, kuid otsing näitas, et teistest tihedamini ilmnesid justnimelt eespool kirjeldatud mudelid, mis tüüpiliselt viitab nende tõhususele.

Nagu selle alapeatüki alguses mainitud, toimivad mudelid kõige paremini kui kasutada andmestikke neile sobival kujul. Samas pole välistatud ka võimalus viia andmestikud valitud mudeli jaoks sobivale kujule.

Siinse töös on võetud kasutusele närvivõrkude mudel. Selle valiku põhjused ja mudeli täpsem töökäik on lahti mõtestatud järgnevates peatükkides.

1.4 Töö tarbeks valitud masinõppe teek

Masinõppe teekide valik on suur ja lai. Neid leidub paljude programmeerimiskeelte juures vahelduva kasutajatoe, kasutatavuse, dokumenteerituse, mudelite arvukuse ja avatusega. Siinse töö tarbeks sobilikud masinõppe teegid on kõik, mis sisaldavad närvivõrkude mudelit ja on kirjutatud Java’s, mistõttu nende loetlemine on autori silmis natuke küsitav. Seetõttu on kasutusele võetud otsingutulemustes kõige tihedamini esinev Java masinõppe teek WEKA (versioon 3.0.8). Valiku täpsemad põhjused on kirjeldatud allpool.

WEKA on Java’s kirjutatud masinõppe mudelite kogumik andmete kaevandamiseks (The University of Waikato, kuupäev puudub). WEKA’l on kaks poolt:

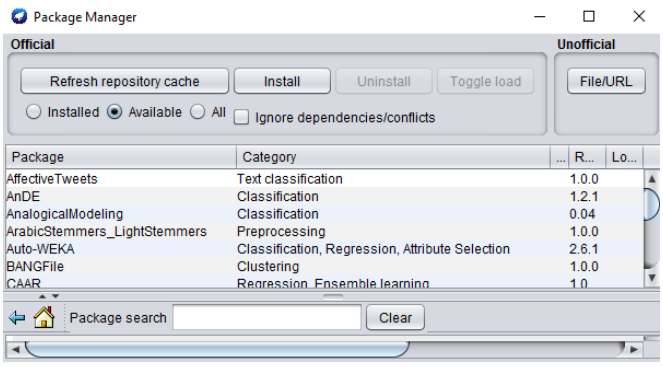
- WEKA on rakendus, mida saab kasutada programmeerimisoskusi omamata (vt joonis 5).

Joonis 5. WEKA rakendus

- WEKA on teek, kus on Java programmeerimiskeeles kirja pandud enamik levinud masinõppe mudelitest.

WEKA on üles ehitatud selliselt, et nii rakendus kui ka mudelid on ühes jar failis. Kui kasutaja soovib kasutada rakenduse poolt, siis teeb ta topelt-klõpsu jar failile. Kui aga kasutaja soovib kasutada WEKA mudeleid programmiliselt, siis on tarvis teha see jar fail sõltuvuseks oma Java projektis, mispeale on kõik sisseehitatud mudelid kasutajale koodis kättesaadavad.

WEKA on avatud lähtekoodiga, mistõttu paljud mudelid, mida WEKA'sse pole sisse ehitatud, on juurde ehitatud ja tehtud kättesaadavaks teiste kasutajate poolt WEKA's sisseehitatud pluginate halduri kaudu (vt joonis 6). See teeb WEKA küllaltki täiuslikuks.



Joonis 6. WEKA pluginate haldur, kust saab kolmandate osapoolte puginaid alla laadida

2 Rakendus ning seda hõlmavad elemendid

Selles peatükis on kirjeldatud loodud rakenduse kasutusvõimalused, koostisosad, kasutamine, struktuur, nõuded ning närvivõrkude mudeli ehitusprotsess ja selle töökäik.

2.1 Rakenduse kasutusvõimalused

Esialgse plaani järgi pidi rakendus olema lihtsalt veebirakendus, mis võtab sisendiks teksti ning väljundina tagastab kasutajale teksti keeletaseme, autori vanuse, elukoha, emakeele jm parameetrid. Tänu EVKK morfosüntaktiliselt märgendatud tekstidele ja masinõppe meetodi kasutamisele oleks see olnud tehtav, kuid rakenduse maht oleks kõvasti ületanud praeguse mahu. Ka masinõppe mudelite parajaks seadistamine ning ehitamine oleks tõenäoliselt kõvasti rohkem aega võtnud. Seetõttu ennustab rakendus hetkel ainult teksti keeleoskustaset A2, B1, B2 ja C1 tasemete piires.

Esimene ja peamine, mida rakendus teeb, on sisestatud teksti autori keeletaseme ennustamine. Selleks on valmistatud kaks närvivõrkude mudelit, mille vahel kasutaja saab valida. Need mudelid on autor koostanud EVKK tuumkorpuse morfosüntaktiliselt märgendatud tekstidest leitud kolmikuid (trigramme) kasutades. Kahjuks ei ole tasemetesse koondatud tekstid ühte tüüpi, varieerudes esseedest ametikirjadeni, mistõttu ennustus ei pruugi alati just kõige täpsem tulla. Rakendust kasutades võib tekkida olukord, kus olemasolevate mudelite algandmed on sisult või vormilt lihtsalt liiga erinevad sellest, mida kasutaja sisestab. Seetõttu on rakendusse sisse ehitatud võimalus mudelit arendada. Rakenduse kasutaja saab ehitada oma andmestikku kasutades uue mudeli ja seda koheselt kasutada. Selline mudelite ehitamise funktsionaalsus eemaldab algandmete keelelist ja vormilist piirangut. Tänu sellele saavad kasutajad ehitada mudeleid, kasutades näiteks inglisekeelseid või eripäraselt töödeldud tekste.

Lisaks eelnevalt mainitud kahele funktsionaalsusele on rakendusel ka serveri pool. Rakendust võib käivitada serveril ning teha sellele HTTP päringuid veebilehtedelt. Sisseehitatud päringute abil saab muuta

- mõningaid loodava mudeli parameetreid;
- mõningaid mudeli filtri parameetreid;
- valida, mis mudelit parasjagu ennustamiseks kasutatakse.

Saab teha päringuid, kasutades

- teksti keeleoskustaseme ennustamiseks aktiivset mudelit nii kolmikute (trigrammide) alusel kui ka ilma;
- mudeleid ja algandmeid, mis on hetkel olemas.

Samuti võib käivitada oma mudeli ehituse.

Rakendusel on ka konfigureerimise võimalus. Rakenduse esimesel käivitusel (ükskõik mis viisil) otsib see kaustas, kus rakendus käivitati, `config.txt` faili. Kui ta seda ei leia, siis loob rakendus selle faili ise vaikimisi parameetritega. Selles failis sisalduvad kõik eelnevalt mainitud parameetrid, mida sai muuta HTTP päringutega. Korduval rakenduse käivitusel võtab see need parameetrid arvesse. Autori arvates on niisuguse faili kasutamine arusaadavam lihtkasutajatele kui käsurealt mitme erineva parameetri lisamine.

2.2 Rakenduse koostisosad

Selles alapeatükis on kirjeldatud rakenduses kasutusel olevad koostisosad.

Java

Rakendus oli loodud Java't kasutades. Java on populaarne ja laialt levinud programmeerimiskeel, mille avalikustas Sun Microsystems aastal 1995 (Oracle, kuupäev puudub). Kõige uuem versioon on Java 9, kuid siinses töös loodud rakendus kasutab Java 8't, sest töö kirjutamise hetkeks on Java 9 veel varajase ligipääsu seisundis ning ei paku midagi, mis loodud rakendust mingisugusel moel täiustaks.

Java valikut mõjutas ka see, et

- autor on oma Java pädevuses piisavalt kindel;
- autor soovis olla võimeline koodi muutma madalal tasemel;
- Java rakendus töötab samamoodi mitmel operatsioonisüsteemil.

Spark Framework

Rakenduse veebiserveri osa oli loodud Spark Framework'i kasutades. Spark Framework on tasuta ja avatud lähtekoodiga lihtsate veebirakenduste loomiseks mõeldud tarkvara, mis on loodud Java's ning ei kasuta mudel-vaade-kontroller mustrit (ingl *model-view-controller*

pattern), mis teatud määral lihtsustab selle raamistiku kasutust (Spark (software), kuupäev puudub).

Spark Framework võeti kasutusele seetõttu, et

- autor ei omanud eelnevat kogemust Java veebirakenduste raamistikega ja antud juhul pole erilist vahet, mis raamistikku kasutada;
- Spark Framework tundus võrreldes teistega kõige kiiremini ja lihtsamini kasutusele võetav.

WEKA

Rakenduse masinõppe mudel oli võetud WEKA masinõppe teegist.

WEKA võeti kasutusele seetõttu, et

- autor oli eelnevalt teadlik WEKA olemasolust;
- WEKA't kasutades on eelnevalt läbi viidud uuringud eesti keele õppija keeleoskustaseme ennustamise teemal (Lõo, 2012; Lõo & Vajjala, 2013; Lõo & Vajjala, 2014), mis andis autorile kindlust selle teegi kasutamiseks;
- WEKA kasutamine Java's tundus esmapilgul olevat väga hästi dokumenteeritud.

Närvivõrgud

Teksti keeleoskustaseme ennustamisega tegelev mudel on närvivõrkude mudel.

Närvivõrkude mudel valiti seetõttu, et

- mudel on hästi adapteeruv ja selle kasutamine annab häid tulemusi sisendile vaatamata (Mohan Raju, Srivastava, Bisht, Sharma, & Kumar, 2011);
- autoril on eelnev kogemus närvivõrkude mudeli kasutamisega;
- uurimuses Lõo 2012 oli võrreldud eesti keele õppija keeleoskustaseme ennustamist kolme masinõppe mudeli abil, sealhulgas närvivõrgud, mis andis parima tulemuse. Järgnevates Lõo ja Vajjala uuringutes (2013; 2014) närvivõrke enam ei kasutatud, kuid siinse töö autori jaoks polnud ilmne, kas Lõo oli oma 2012. aasta uurimuses proovinud ka närvivõrkude mudeli parameetrite kohendamist, mistõttu otsustati

siinses töös testida, kas närvivõrkude mudeli parameetreid kohendades saab veelgi paremaid tulemusi.

EVKK morfosüntaktiliselt märgendatud tekstid ja iga keeleoskustaseme kolmikud

Närvivõrkude mudelite treenimiseks kasutatud algandmed olid loodud EVKK veebilehel kasutusel olevad morfosüntaktilised märgendid ning tuumkorpuse ja taseme tekstidest leitud kolmikud (trigrammid).

EVKK tuumkorpuse tekstidel on lisatud kokku 13 märgendit, millest enamus on informatiivsed (nt andmed autori kohta), välja arvatud tekstitüüp, sest tekstitüübi abil saab päris tõhusalt algandmeid ühtlustada, ja keeletase, mis on klassiväärtuseks lõppalgandmetes. Algandmetes võimalikud tekstitüübi valikud on essee, referaat, muu, ametikiri, harjutus, bakalaureusetöö, vastus küsimusele, isiklik kiri, analüüs, tõlge, ümberjutustus, arvamuslugu ja trükis ilumunud tekst. Tuumkorpuse sagedam tekstiliik on essee, harvem isiklik kiri, vastus küsimusele. Keeleoskustasemete valikut saab piirata kolme- ja kuueastmelise skaalaga: A, A1, A2, B, B1, B2, C, C1 ja C2.

Töö kirjutamise hetkel on EVKK-s kokku 12738 teksti, millest ainult 4064-l on keeletaseme märgend olemas (vt tabel 1). Kuna keeletasemed A, B ja C pole CEFR skaala jaotus, siis pole nende märgenditega kirjeid mudelite ehitamisel algandmetena kasutatud. Juhendaja soovitusel loobuti ka A1 ja C2 keeletaseme kirjete kasutamisest, kuna tasemeoskuste määramisel on tavaks arvestada A2 kuni C1 andmetega.

Tabel 1. Tekstide jagunevus keeletasemeti

Keeletase	A	A1	A2	B	B1	B2	C	C1	C2
Arv	1378	1	217	1154	412	228	359	130	185

Lisana on EVKK-st võimalik kätte saada tekste ka n-grammide kujul. Valida saab üksgrammidest viisgrammideni. Mudelite ehitamisel on käesolevas töös algandmetena kasutatud kolmikuid ehk trigramme, sest praktikas on nende kasutamine tavaline (vt Eslon, 2014, lk 18–19).

Mudeli väjatöötamiseks on kasutatud EVKK tuumkorpuse A2-, B1-, B2- ja C1-taseme tekste ning neist leitud kolmikuid (trigramme).

Muud osad

Sõnaliikide määramiseks kasutab rakendus estnltk python'i teeki[7]. Selleks on tehtud eraldi python'i skript, mille ülesanne on võtta vastu rakendusse sisestatud tekst ja tagastada teksti kõik sõnaliigid. Seda skripti kasutab rakendus ise.

Rakendusega tuleb kaasa ka R skript, mille käesoleva töö autor on ise kirjutanud ning kasutanud EVKK märgendatud tekstide töötlemiseks ja arff vormingus failide koostamiseks. Selle lisamise eemärk on abistada kasutajat algandmete ja arff vormingus failide loomisel.

2.3 Rakenduse kasutus

Selles alapeatükis on rakenduse kasutamiskirjeldus koos märkustega.

Rakenduse käivitus toimub käsurealt koos mõningate valikuliste parameetritega. Need parameetrid on järgmised.

- „`-a <port>`" – selle parameetriga käivitamine paneb käima rakendusse sisseehitatud veebiserveri; lisada võib ka kasutajale sobiva pordi, mille pealt server päringuid vastu võtma hakkab. Vaikimisi port on „`4568`".
- „`-m "<mudeli_nimi>"`" – käivitades rakendust selle parameetriga algab mudeli ehituse protsess. Arvesse võetakse eelnevalt mainitud `config.txt` faili sees olevaid parameetreid. Mudeli nime lisamine on valikuline, kuid lisamise puhul peab nimi olema jutumärkide vahel. Mudeli vaikimisi nimi on selle loomise hetke ajatempel. Loodud mudel salvestatakse kausta `models`, mis moodustatakse samasse kausta, kust rakendus oli käivitatud.
- „`-pn "<tekst>"`" – selle parameetriga käivitades teeb rakendus teksti esialgu kolmikuteks ning seejärel sisestab need mudelisse keeleoskustaseme ennustamiseks. Ennustamiseks kasutatav mudel peab olema paika pandud eelnevalt mainitud `config.txt` failis. Lisaparameeter `<tekst>` on selle käsu juures kohustuslik ning sisestatud tekst peab paiknema jutumärkide vahel.

[7] https://github.com/estnltk/estnltk

- „`-pt "<tekst>"`“ – see parameeter on sarnane eelmisega, erinedes ainult selle poolest, et sisestatud teksti ei töödelda kolmikuteks.
- „`-c`“ – see parameeter tuleb lisada käsu lõppu, peale eelnevalt mainitud parameetreid. Selle parameetri olemasolul teostab rakendus väljatrükki igas etapis, mida läbib. Erandina teostatakse väljatrükk alati mudeli ehitamisel ning serveri töötamise ajal.

Käsu tüüpkasutuse näide: „`java -jar me.jar -m "minu_mudel" -c`“. See käsk tähendab, et rakendus ehitab mudeli, salvestab selle nimega `minu_mudel` kausta `models` ning samaaegselt teostab väljatrükki iga sammu juures. Vaikimisi käivitub rakendus veebiserverina.

Rakendust saab käivitada ka sellele topeltklõpsates, kuid siis ei saa valida, mis funktsionaalsus käivitub ning rakenduse väljatrükki ei kuvata.

Koostisosana paikneb mainitud python'i skript väljaspool rakendust, sest selle kaasapakkimine jar faili pole eriti mõistlik. Selleks, et skript oleks rakendusele nähtav ja kasutatav, peab selle asukoht olema „`./scripts/py/text_to_postag.py`“, kus „`./`“ on kaust, milles rakendus käivitati.

2.3.1 Mudeli ehitus

Rakenduse mudeli ehitamise tarbeks on vajalik WEKA arff vormingus algandmete fail (vt joonis 7).

```
@relation Minu_andmestik //sisuliselt andmestiku nimi

//sisuliselt veerud ja nende tüübid
@attribute ngramid string //veerg nimega ngramid, mille tüübiks on string
@attribute keeletase {A2,B1,B2,C1} //klassi atribuut nimega keeletase, koos kõigi võimalike klassidega

@data
'...PSV SVG VGS...',C1
'...DVP VPS PSZ...',B2
...
```

Joonis 7. Weka arff vormingus faili näide. Tekst, mille ees on kaks kaldkriipsu, on kommentaar. Päris failis neid olla ei tohi.

See fail on tarvis omakorda panna kausta `arff`, mis kasutaja peaks eelnevalt looma samas kaustas, kust rakendus käivitatakse. `config.txt` failis on tarvis muuta `trainingData` parameetrit, kuhu on vaja sisse kirjutada „`/arff/<teie_algandmete_faili_nimi>.arff`“ (ilma jutumärkideta). Keeletasemete veerg peab olema viimane ning hetkel lubatud keeletasemed on A2, B1, B2 ja C1. Vajadusel

võib muuta ka mudeli ehitusega seotuid parameetrid. Õige parameetri kasutuse näide: „`trainingData = /arff/minuAndmed.arff`“ (ilma jutumärkideta).

Loodud mudel salvestatakse `models` kausta, mis paikneb samas kaustas, kust rakendus oli käivitatud.

2.3.2 Ennustamine

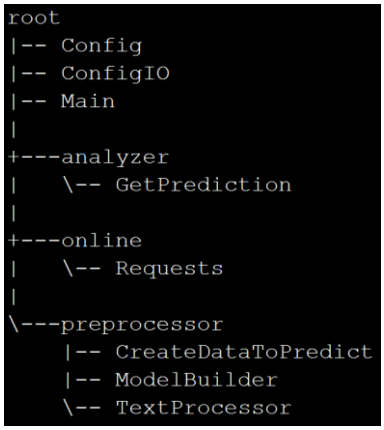
Rakendus kasutab ennustamiseks eelnevalt valmis tehtud mudeleid. Rakendusega tulevad kaasa kaks kõige parema ennustamistäpsusega mudelit, milleni autor on jõudnud. Nende või kasutaja enda loodud mudelite rakendamiseks on tarvis, et nad paikneksid `models` kaustas. `config.txt` failis on tarvis muuta `activeModel` parameetrit, kuhu tuleb sisse kirjutada „`/models/<mudeli_nimi>.model`“ (ilma jutumärkideta). Sellisel juhul ei muuda mudeli ja filtri parameetrite muutmine midagi. Õige parameetri kasutuse näide: „`activeModel = /models/minu_mudel.model`“ (ilma jutumärkideta).

Peale käsu sisestamist läheb keeleoskustaseme ennustamiseks sisestatud tekst valitud mudelisse, millele järgneb rakenduse väljatrükk, kus on kirjas tõenäosus iga klassi kohta. Kui ennustus oli tehtud läbi veebiserveri, siis tagastatakse tõenäosus HTML vormingus.

Ennustamise puhul on tähtis, et tekst, mille keeletaset ennustada tahetakse, oleks samal kujul, mis mudeli ehitamises kasutatud algandmedki. Kui algandmetes on tekstid näiteks kolmikute kujul, siis peaks ka sisestatud tekst olema kolmikuteks töödeldud (selleks peab käsus kasutama „`-pn`“ parameetrit), vastasel juhul pole tehtav ennustus korrektne. Muudel juhtudel (nt üks-, kaks- või neli-grammid) peab kasutaja ennustamiseks sisestatava teksti ise vastavale kujule töötlema.

2.4 Rakenduse struktuur

Selles alapeatükis on kirjeldatud loodud rakenduse struktuuri, selle klasse ja funktsioone ning nende omavahelisi seoseid. Rakenduse täpsem sisu leiab aadressil „`https://github.com/janekos/masinoppe_ennustus`“.



Joonis 8. Piltlik kujutus rakenduse failide struktuurist

Rakenduse juurkaustas, paketis (ingl *package*) `root`, paiknevad rakenduses esimesena kasutatavad klassid `Config`, `ConfigIO` ja `Main`. Klassis `Main` paikneb käivituskäsuga kaasatulnud parameetrite töötlemine, kus otsustatakse mis funktsioonid edasi käivituvad. Samas klassis käivitatakse ka `ConfigIO` klass, mille peamine ülesanne on `config.txt` faili haldamine – faili loomine, uuendamine või selle olemasolul sealt parameetrite lugemine. Klass `Config` vastutab rakenduses jooksvalt muutuvate muutujate hoidmise eest. Kui mõne muutuja väärtus `Config` klassis muutub, siis käivitub `ConfigIO` klassis faili uuendamise funktsioon.

Paketis `analyzer` paikneb teksti ennustamisega tegelev klass `GetPrediction`. See klass sisaldab ainult ühte funktsiooni, mille sisse laetakse töö käigus eelnevalt loodud mudel. Selle asukoht on paika pandud `Config` klassi muutujas `activeModel`, sisestatud tekst viiakse mudeli jaoks sobivale kujule, kasutades paketis `preprocessor` paiknevat `CreateDataToPredict` klassi funktsiooni. Seejärel teostatakse ennustus, mille tulemus trükitakse konsooli aknasse või tagastatakse HTML vormingus tehtud päringule.

Paketis `online` paikneb klass `Requests`, mis tegeleb rakenduse veebiserveri osaga. Selles klassis on kirja pandud kõik teekonnad, millelt rakendus HTTP päringuid ootab ning mille töö on kirjeldatud eelnevas alapeatükis (vt 2.1).

Paketis `preprocessor` paiknevad kõik klassid, mis tegelevad eeltöötlusega. Nendeks klassideks on `CreateDataToPredict`, `ModelBuilder` ja `TextProcessor`. `CreateDataToPredict` tegeleb sisendina antud teksti ümbertõstmisega mudeli jaoks sobivale kujule. Piltlikult kirjeldades moodustatakse WEKA klasse ja funktsioone kasutades

virtuaalne `arff` fail, mis on loodud mudeli jaoks sobilikul kujul. Klassis `ModelBuilder` paikneb funktsioon, mis ehitab närvivõrkude mudelit. Mudeli ehitamisel lähevad arvesse `Config` klassis paiknevad muutujad. Klassis `TextProcessor` on tekstitöötlemisfunktsioonid, mis teevad sisestatud teksti kolmikuteks (trigrammideks). Üks funktsioonidest selles klassis tegeleb python'i skripti käivitamisega.

2.5 Rakenduse kasutamise nõuded ja soovitused

Rakenduse kasutamiseks peavad olema täidetud teatud nõuded, et rakendus või selle funktsionaalsused töötaksid. Pakutud soovitused on valikulised.

- Java – arvutil, kus rakendus käivitatakse, peab olema paigaldatud vähemalt Java 8. Rakenduse arendus toimus JDK 1.8.0_151 peal ja testimine JRE 1.8.0_161 versioonil. Spark Framework'i ja WEKA't eraldi alla laadima ei pea – need on rakendusse sisse pakitud.
- Python ja estnltk – kui kasutaja soovib kolmikutega toimetada, siis selleks, et rakendus saaks teksti sõnaliike kätte, on vajalik Python 3 ning estnltk teek. Rakenduses kaasatulev Python'i skript, kus kasutatakse estnltk teeki, oli arendatud ja testitud Python 3.6 peal.
- Masinõppe mudelite ehitus – kui kasutaja soovib enda arvutis ehitada masinõppe mudeleid, siis oleks tarvis võtta arvesse, et mudelite ehitus võib kohati olla väga mahukas protsess. Autori enda kogemuse järgi võiks kasutaja arvutil olla vähemalt 8GB muutmälu ja võimekas protsessor. Tõenäoliselt on mudelite ehitamine võimalik ka lahjema riistvaraga, aga sellisel juhul oleks arvutil raske mitut mudelit korraga ehitada või oleks arvuti ehitamise ajal muudel eesmärkidel kasutamiseks kõlbmatu.

2.6 Rakenduse närvivõrkude mudelite ehitus ja töökäik

Selles alapeatükis on kirjeldatud, kuidas rakenduses toimub mudelite ehitamine ning kuidas need toimivad.

2.6.1 Närvivõrkude mudeli ehitamine

Mudeli ehitus rakenduses algab vastava käsu sisestamise või päringuga. Kõigepealt laetakse parameetrites paikapandud algandmete fail loodava mudeli treeningandmeteks ning valitakse viimane dimensioon klassi dimensiooniks (teisisõnu: selleks mida ennustada tahetakse).

Siinse töö kirjutamise hetkel võivad mudelite rakenduses kasutavate mudelite klassi dimensioonis olla ainult A2, B1, B2 ja C1 märgendid.

StringToWordVector filter

Kui algandmete sisselugemisel vigu ei tekkinud, siis rakendatakse `StringToWordVector` filter. See filter viib üle kõik stringid (vt joonis 9) numbrilisele kujule ning teeb iga stringi dimensiooniks (vt joonis 10) (teisisõnu: viib algandmed bag-of-words kujule), jättes seejuures välja kõik kirjavahemärgid ja klassi dimensiooni märgendid.

```
@attribute tekst string
@attribute keeletase {A2,B1,B2,C1}

@data
'Mina olen Juku! Mina ei lenda.',C1
'Tere. Mina olen Kati. Mulle ei meeldi süüa.',B2
'Ma olen Mati. Mulle meeldib rattaga sõita.',B1
'Mina olema Vladim. Mulle ei meeldib kool.',A2
```

Joonis 9. Näidis algandmed enne StringToWordVector'i rakendamist

```
@attribute keeletase {A2,B1,B2,C1}
@attribute ei numeric
@attribute kool numeric
@attribute meeldib numeric
@attribute mina numeric
@attribute mulle numeric
@attribute olema numeric
@attribute vladim numeric
@attribute ma numeric
@attribute mati numeric
@attribute olen numeric
@attribute rattaga numeric
@attribute sć¼ita numeric
@attribute kati numeric
@attribute meeldi numeric
@attribute sć¼ć¼a numeric
@attribute tere numeric
@attribute juku numeric
@attribute lenda numeric

@data
{0 C1,1 1,4 1,10 1,17 1,18 1}
{0 B2,1 1,4 1,5 1,10 1,13 1,14 1,15 1,16 1}
{0 B1,3 1,5 1,8 1,9 1,10 1,11 1,12 1}
{1 1,2 1,3 1,4 1,5 1,6 1,7 1}
```

Joonis 10. Näidis algandmed peale StringToWordVector'i rakendamist

Joonisel 10 on rakendatud filter ainult tõese `LowerCaseTokens` parameetriga. Sellisel kujul on kirjete väärtused eraldatud komaga, kus väärtuse vasakpoolne arv on sõna number (näiteks

sõna „ei“ number on 1) ja parempoolne arv näitab, et sõna esineb selles kirjes. Klassiväärtuse puhul pole paremal pool arvu, vaid on selle klassi nimi.

`StringToWordVector` filtri puhul võib muuta mahukat hulka parameetreid enne selle rakendamist. Filtri loomisel rakendatakse ainult järgmisi parameetreid.

- `TFTransform` – rakendab võrrandi $\log(1 + fij)$, kus *fij* on sõna *i* sagedus veerus *j* (vt joonis 11) (rakenduses vaikimisi tõene).

```
{0 C1,1 0.693147,4 0.693147,10 0.693147,17 0.693147,18 0.693147}
```

Joonis 11. Näidis algandmete esimene veerg millele on rakendatud TF transformatsioon

- `IDFTransform` – rakendab võrrandi $fij * \log\left(\frac{dokumentide\ arv}{arv\ dokumente\ kus\ sisaldub\ sõna\ i}\right)$, kus *fij* on sõna *i* sagedus veerus *j* (vt joonis 12) (rakenduses vaikimisi tõene).

```
{0 C1,1 0.287682,4 0.287682,10 0.287682,17 1.386294,18 1.386294}
```

Joonis 12. Näidis algandmete esimene veerg, millele on rakendatud IDF transformatsioon

- `OutputWordCounts` – muudab kirjete väärtustes parempoolse arvu numbriks, mis näitab, mitu korda antud sõna kordub stringis (vt joonis 13) (rakenduses vaikimisi tõene).

```
{0 C1,1 1,4 2,10 1,17 1,18 1}
```

Joonis 13. Näidis algandmete esimene veerg, millele on rakendatud OutputWordCounts parameeter

- `WordsToKeep` – paneb paika, mitu sõna säilitada dimensioonina (rakenduses vaikimisi 10000 sõna).
- `LowerCaseTokens` – muudab kõik suured tähed väikesteks (rakenduses vaikimisi tõene).
- `AttributeIndices` – paneb paika, mis dimensioone algandmetest kasutada (rakenduses vaikimisi „first-last“).

Nende parameetrite valik on tingitud sellest, et mudelite testimisel selgus, et neid parameetreid muutes muutub ka ennustamise tulemus. Nende parameetrite muutmiseks on rakendusse sisse ehitatud vastavad muutujad, mida saab muuta kas `config.txt` failist või HTTP päringuga.

Kuna WEKA's on TF-IDF[8] (Term Frequency – Inverse Document Frequency) teostatud omal moel, siis peavad selle õigeks kasutuseks olema tõesed parameetrid `TFTransform`, `IDFTransform` ja `OutputWordCounts` (TF-IDF in the StringToWordVector Filter, 2014), mille puhul on võrrand rakendatud järgneval kujul: $\log(1 + fij) * \log(\frac{dokumentide\ arv}{arv\ dokumente\ kus\ sisaldub\ sõna\ i})$. Algandmete lõplik kuju on kajastatud joonisel 14.

```
@attribute keeletase {A2,B1,B2,C1}
@attribute ei numeric
@attribute kool numeric
@attribute meeldib numeric
@attribute mina numeric
@attribute mulle numeric
@attribute olema numeric
@attribute vladim numeric
@attribute ma numeric
@attribute mati numeric
@attribute olen numeric
@attribute rattaga numeric
@attribute sєµita numeric
@attribute kati numeric
@attribute meeldi numeric
@attribute sє¼є¼a numeric
@attribute tere numeric
@attribute juku numeric
@attribute lenda numeric

@data
{0 C1,1 0.199406,4 0.316051,10 0.199406,17 0.960906,18 0.960906}
{0 B2,1 0.199406,4 0.199406,5 0.199406,10 0.199406,13 0.960906,14 0.960906,15 0.960906,16 0.960906}
{0 B1,3 0.480453,5 0.199406,8 0.960906,9 0.960906,10 0.199406,11 0.960906,12 0.960906}
{1 0.199406,2 0.960906,3 0.480453,4 0.199406,5 0.199406,6 0.960906,7 0.960906}
```

Joonis 14. Näidis algandmete lõplik kuju

Mudel

Kui filter on vastavalt konfigureeritud, siis on järgmine samm mudeli loomine. Esiteks luuakse närvivõrkude mudeli üksus, mille nimi WEKA's on `MultilayerPerceptron`, ning konfigureeritakse vastavaid parameetreid. Nendeks parameetriteks on:

- `HiddenLayers` – määrab varjatud kihtide arvu (rakenduses vaikimisi 7);
- `LearningRate` – määrab õppimise määra (rakenduses vaikimisi 0.3);
- `Momentum` – määrab õppimise hoo (rakenduses vaikimisi 0.2);
- `TrainingTime` – määrab õppimise iteratsioone (rakenduses vaikimisi 500);

[8] TF-IDF – on numbriline statistika, mis kajastab sõnade tähtsust dokumentidele mõnes kogumikus (Rajaraman & Ullman, 2011).

- `NormalizeAttributes` – määrab ära, kas dimensioone normaliseeritakse (rakenduses vaikimisi väär).

Sarnaselt filtri parameetritele on need parameetrid valitud seetõttu, et testimise käigus muutsid need ennustamistulemust. Samuti on need parameetrid muudetavad nii `config.txt` kui ka HTTP päringu kaudu.

WEKA'le omapäraselt on filtri kasutamiseks tarvis `FilteredClassifier` mudelit. See on sisuliselt ümbris programmiliselt kasutatavatele mudelitele WEKA's, millede juures tahetakse kasutada filtrit. Ilma ei saaks ennustada erinevate tekstide keeletaset, sest algandmed on kujult erinevad. Seega on vaja loodud närvivõrkude mudel ja `StringToWordVector` filter sisestada `FilteredClassifier`-isse ja alles seejärel saab alustada mudeli treenimist. Kahjuks ei saa loodud mudelit otseselt visualiseerida, sest salvestatud mudelid on mõistetavad ainult WEKA funktsioonidele.

2.6.2 Närvivõrkude mudeli töökäik

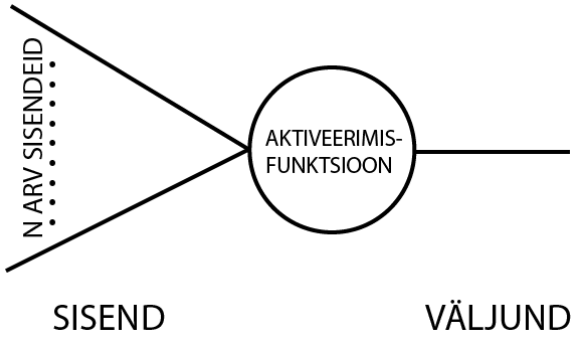
Piltlikult võib jagada närvivõrkude mudeli kolmeks osaks: sisendkiht, peidetud kihid ja väljundkiht (Smith, kuupäev puudub). See on mitmekihiline närvivõrkude mudel (ingl *multilayer neural networks*), kuid laiemas kasutuses nimetatakse seda lihtsalt „närvivõrgud". Kõikides kihtides esinevad sõlmed. Kihtide ülesanded jaotuvad järgmiselt.

- Sisendkihi ülesanne on edastada sisendina saadud töödeldud tekst. Selles kihis on sõlmi sama palju kui algandmetes dimensioone.
- Peidetud kihtide ülesanne on piltlikult rääkides teostada sisendite sõelumist ning sõltuvalt peidetud kihtide arvust tekivad aina peenemad väljundid.
- Väljundkihtide ülesanne on peidetud kihtidelt saadud väljundi viimane töötlus, mille tagajärjel tehakse teksti keeletaseme ennustus.

Sõlmed

Sõlme sisse tulevad sisendid igalt eelnevalt sõlmelt, väljaarvatud sisendkihi puhul (vt joonis 15). Igal sisendil on oma kaal (ingl *weight*) ja eelmise sõlme väljund (Nielsen, 2017). Siinjuures näitab kaal sisendi tähtsust teatult sõlmelt. Näiteks, sõna „ja" võib esineda kõigi nelja keeletaseme puhul, mistõttu selle kaal oleks mõne sõlme juures madal. Mudeli

treenimise alguses on kaalude väärtused suvalised, kuid treenimise jooksul sätitakse need parajaks.



Joonis 15. Sõlme näidis

Sisenditele vastavalt otsustab sõlm väljundi väärtust. Seda otsust kirjeldab võrrand $väljund = \begin{cases} 1\ kui\ \sum_i(kaal_i * sisend_i) + kõrvalekalle > 0 \\ 0\ kui\ \sum_i(kaal_i * sisend_i) + kõrvalekalle \leq 0 \end{cases}$. Kõrvalekalle lisab siin ennustamisele vajalikku vahelduvust, mille mudel töötab välja treeningu käigus (Determining Bias for Neural network Perceptrons?, 2011). Kuna närvivõrkude puhul on väljundid 1 ja 0 liiga lineaarsed heade ennustamistulemuste saavutamiseks, siis rakendatakse sigmoid funktsiooni (The University of Waikato, kuupäev puudub). Selle funktsiooni rakendus eelnevale võrrandile näeb välja nii: $väljund = \frac{1}{1+e^{-v}}, kus\ v = \sum_i(kaal_i * sisend_i) + kõrvalekalle$ (Kang, 2017). Tänu sellele on iga sõlme väljund vahemikus 0 kuni 1. Närvivõrkude mudelite kontekstis nimetatakse viimast võrrandit aktiveerimisfunktsiooniks.

3 Kasutatavus, ennustamistulemused ja probleemid

Selles peatükis kirjeldatakse loodud rakenduse kasutatavust, keeletaseme ennustamise tulemusi ja arenduse käigus tekkinud probleeme.

3.1 Rakenduse kasutatavus

Sissejuhatuses oli mainitud, et rakenduse loomise pakkus autorile juhendaja koostöös EVKK-d arendavate lingvistidega. Seega oleks üks võimalus rakendust kasutada EVKK veebileht. Pluss oleks kättesaadavus kõikidele soovijatele, miinus - teatud kasutus piirangud. Nimelt pole hetkel rakendusse sisse ehitatud funktsionaalsust, mis laseks kasutajal oma andmestikke või mudeleid serverisse üles laadida. Seetõttu on rakenduse kasutamine piiratud EVKK algandmetega. Rakendusse on küll sisse ehitatud mudeliehitamise funktsionaalsus läbi HTTP päringu, kuid selle avalik kasutamine võib tuua endaga kaasa teatud turvariske, sest hetkel ei nõua selle päringu kasutus autentimist. Loomulikult saaksid EVKK töötajad ise lisada kasutaja palvel vajalikke mudeleid või algandmeid, mistõttu heaks kiidetud mudelite või algandmete piirang poleks enam nii range.

Teine võimalus on kasutada rakendust iseseisvalt või oma serveris käivitades. Sel juhul kaovad mudelite ja algandmete kasutamispiirangud. Tänu eespool mainitud mudeliehitamise funktsionaalsusele saavad kasutajad keeletaseme ennustamise kitsamatest eesmärkidest lähtudes täiendada mudeleid vajalike algandmetega, mistõttu rakenduse kasutatavus tõuseb märgatavalt. Sellisel kujul oleks rakendus kasutatav tunduvalt laiemalt. Näiteks:

- Äri – rakendust saaks väga edukalt kasutada ärilistel eesmärkidel. Ettevõtted saaksid tänu rakendusse sisse ehitatud mudeli ehitamise funktsionaalsusele arendada enda tarbeks vajalikke mudeleid. Nt mõne ettevõtte tööhõives uuritakse ja võrreldakse, kas tulevas(t)el töötaja(te)l on piisav ametikirjade kirjutamisoskus vähemalt B2-tasemel. Selleks on eelnevalt ametikirjadest leitud algandmed, mille põhjal on loodud mudel, mis võimeline hindama kasutaja kirjutamisviisi vastavust ning tuvastama keeletaset.
- Pedagoogika – sarnaselt ärilistele eesmärkidele saaks rakendust kasutada ka pedagoogilistel eesmärkidel. Nt alustades õpilaste kodutöödest ja lõpetades uurimuslike töödega saaks erinevaid tekstiliike rühmitades kasutada neid algandmetena ning teha nendest mudeleid keeletaseme tuvastamiseks. Rakenduse väljastatud tõenäosused ei pea alati näitama keeletaset, seda ka piiratud

klassifitseeritava klassi muutujatele vaatamata, sest ennustus sõltub algandmetest. See tähendab, et kasutaja algandmetes võivad teksti keeletaseme märgendid tähendada midagi muud, näiteks missuguse keeleoskustasemega lugejatele antud tekst on loetav. See oleks hea abivahend nii kooliõpikute koostajatele kui ka õpetajatele.

- Kurioosum – tõenäoliselt huvitavaim väljund rakenduse jaoks oleks kasutaja üllatamine või hämmastamine rakenduse võimega ennustada mitte ainult teksti keeletaset, vaid ka esile tuua erinevate keelekasutusvariantide (nt murdekeel, kirjakeel, õppijakeel), žanrite (nt uudistekst, intervjuu, arvamuslugu), tekstiliikide (kirjeldus, jutustus, arutlus) jms keelekasutust, mis võib anda uudset materjali eesti keele kasutusgrammatika jaoks.

Nimetatud valdkondades on loodud rakendusel lai kasutuspotentsiaal, samas on rakendus analoogia põhjal kasutatav ka hoopiski kitsamatel ja spetsiifilisematel eesmärkidel.

3.2 Keeletasemete ennustamise tulemused ning nende analüüs

Kuna autor ei omanud enne siinse töö kirjutamist keeletaseme ennustamise kogemusi, siis valdav enamus saadud tulemustest on saavutatud katseeksitusmeetodiga. Lisaks loodud rakendusse pole sisseehitatud rist-valideerimine, mistõttu mudelite testimine toimus WEKA rakenduses. Mudelite ennustamistäpsuse mõõduks on, sarnaselt Lõo töödele, võetud õigesti ennustatud kirjete protsent (ingl *correctly classified instances*), mis on saadud 10-kihilisest rist-valideerimisest.

Närvivõrkude mudelite ehitamine algas käesolevas arenduses sellest hetkest, kui rakenduse esialgsed masinõppe osad hakkasid valmima. Mudelite ehitamisel tugines autor oma masinõppe-alastele teadmistele, määrates parameetreid juhuslike väärtustega ning kasutades mudelite treenimiseks EVKK-st võetud tekstide algandmeid (vt lisa 1). Seetõttu on esimesed mudelid oma vähese ennustusvõime tõttu sisuliselt kasutud (ennustamise täpsus kõigest 28.3567%, mudelite ehitamine võttis aega kuni 16 tundi).

Sõltuvalt halvast ennustamistulemusest, pikast ehitamisajast ja juhendaja soovitusest kasutada ainult nelja keeleoskustaset (A2, B1, B2, C1), otsustas autor muuta lähenemist mudelite ehitamisele. Järgmised mudelid sai loodud trigrammide ehk kolmikute ja nelja keeleoskustaseme ehk klassi algandmete põhjal (vt lisa 2), muutes parameetreid enne iga

treenimiskorda ükshaaval. Uue lähenemisega loodud mudelite ennustustulemused on kajastatud allolevas tabelis 2.

Tabel 2. Tabelis on kajastatud peidetud kihtide suuruse järjekorras mudelite parameetrid ning ennustamistulemus. Punast värvi veeru nimi on filtri parameeter ning sinises on närvivõrkude parameetrid. Parameetrid, mida rakenduses muuta saab, aga pole tabelis kajastatud, olid ehitamisel jooksul konstantsed.

Rida	TF-IDF	Hidden layers	Learning rate	Momentum	Normalize attributes	Training time	Tulemus
1	Tõene	1	0.3	0.2	Tõene	500	39.8985%
2	Tõene	1	0.3	0.2	Väär	500	49.6447%
3	Tõene	1	0.3	0.2	Väär	1000	49.7462%
4	Väär	3	0.3	0.2	Tõene	500	38.9848%
5	Tõene	3	0.3	0.2	Tõene	500	39.8985%
6	Tõene	3	0.3	0.2	Väär	500	67.9188%
7	Tõene	3	0.3	0.2	Väär	1000	67.4112%
8	Tõene	5	0.3	0.2	Tõene	500	36.2437%
9	Tõene	6	0.3	0.2	Tõene	500	36.2437%
10	Tõene	6	0.3	0.2	Väär	500	69.3401%
11	Tõene	7	0.3	0.2	Tõene	500	34.3147%
12	Tõene	7	0.4	0.2	Tõene	500	34.3147%
13	Tõene	7	0.2	0.2	Tõene	500	38.0711%
14	Tõene	7	0.3	0.1	Tõene	500	36.2437%
15	Tõene	7	0.3	0.3	Tõene	500	34.3147%
16	Tõene	7	0.3	0.2	Väär	500	69.7462%
17	Tõene	7	0.3	0.2	Väär	1000	69.6447%
18	Tõene	7	0.3	0.2	Väär	5000	69.6447%
19	Tõene	7	0.2	0.2	Väär	500	71.066%
20	Tõene	7	0.1	0.2	Väär	500	69.1371%
21	Tõene	8	0.3	0.2	Tõene	500	36.2437%
22	Tõene	8	0.3	0.2	Väär	500	70.1523%
23	Tõene	8	0.2	0.2	Väär	500	69.1371%
24	Tõene	9	0.3	0.2	Väär	500	71.9797%
25	Tõene	9	0.2	0.2	Väär	500	70.5584%
26	Tõene	10	0.3	0.2	Väär	500	70.7614%
27	Tõene	11	0.3	0.2	Väär	500	71.066%
28	Tõene	12	0.3	0.2	Väär	500	70.6599%

Kolmikute kirjete ja nelja keeleklassi algandmete kasutusega langes mudelite ehitamise aeg kuni tunnini. Tänu sellele sai autor kiiremini ja rohkem mudeleid treenida.

Enne ennustamistulemuste saamist oli autor arvamusel, et mida suurem on `TrainingTime` parameetri väärtus, seda parem on tulemus. Ent read 16, 17 ja 18, kus tulemused vastavalt 69.7462%, 69.6447% ja 69.6447%, näitavad vastupidist. Sama kordub ka ridade 6 ja 7 juures.

Ridade 2 ja 3 puhul on aga seis vastupidine – tulemuslikkus tõuseb. Ainuke erinevus nende kolme grupi vahel on `HiddenLayers` parameetrite väärtus, vastavalt 7, 3 ja 1. Ainuke oletus, mida autor siinjuures pakkuda julgeb, on see, et mudel, kus `HiddenLayers` parameeteri väärtus on üks, on lihtsalt liiga lihtne oma struktuurilt ning seda saab paremaks ainuüksi suurema `TrainingTime` parameetri väärtuse korral.

Tulemusi uurides võib tekkida tunne, et `HiddenLayers` ja `NormalizeAttributes` parameetrite seos on müstiline. Ühelt poolt, kui `NormalizeAttributes` parameeter on tõene, siis väiksema `HiddenLayers` parameetri väärtusega mudelite tulemused on paremad kui suurema `HiddenLayers` parameetri väärtusega mudelitel (näiteks ridadel 1 ja 15). Teisalt, kui `NormalizeAttributes` parameeter on väär, siis kehtib vastupidine seaduspärasus (näiteks ridadel 6 ja 16) – tulemus on ligikaudu kaks korda parem. Järelikult, mida suurem on `HiddenLayers` parameetri väärtus, seda parem tulemus. Samas on parim tulemus rea 24 puhul, kus `HiddenLayers` parameetri väärtus on vaid 9. Sellest kõrgemate `HiddenLayers` parameetri väärtustega mudelite tulemused on juba halvemad (näiteks ridadel 26, 27 ja 28). Niivõrd segaste tulemuste tõttu ei oska autor põhjuste kohta midagi oletada.

Muutes `LearningRate` parameetrit, tekkisid järjekordselt küsitavad tulemused, nagu võib täheldada ridade 18 ja 19 ning 24 ja 25 juures, mis lõid autoril arusaama tulemuste põhjustest veelgi segasemaks. Muutes `Momentum` parameetrit, muutusid tulemused aina halvemaks, mistõttu autor seda pikemalt ei testinud.

Mudelite ehitamisel osutus parimaks mudel reanumbriga 24, kus ennustamise tulemus oli 71.9797%. Halvimad tulemused olid mudelitel reanumbritega 11, 12 ja 15, kus tulemus oli 34.3147%.

Mingil hetkel mudelite loomise käigus oli mudel reanumbriga 16 parima ennustamisetulemusega, mistõttu autor otsustas veelkord proovida esimeste mudelite ehitamisel kasutatud algandmeid uue mudeli ehitamiseks, et kinnitada selliste algandmete halba ennustusvõimekust. Lisaks koostas autor mudeli loomiseks algandmeid, kus olid sees nii tekstid kui ka kolmikud. Nende mudelite valideerimine kestis ligikaudu kolm päeva ning tulemusteks olid vastavalt 72.1827% ja 71.7766%. Saadud tulemused hävitasid lõplikult autori arusaama närvivõrkude mudelite ehitamisest, aga samas on nad mõistetavad, sest

esialgsetes algandmetes on peale filtri rakendamist 36271 dimensiooni (kolmikutel 2210), mis tõenäoliselt lubavad närvivõrkude mudelil peenemaid otsuseid teha keeletaseme ennustamisel. Ajapuuduse tõttu ei jõudnud autor ristvalideerida esialgseid algandmeid kasutades mudelit, millel oleksid reanumbriga 24 mudeli parameetrid.

Autori arvates saab tulemusi kindlasti paremaks teha, kui algandmeid oleks rohkem ning need oleksid ühtlasemad. Näiteks võiks tekstidest välja jätta kõik sidesõnad, sest need on omased kõigile neljale keeleoskustasemele või jagada algandmed tekstitüüpide järgi. Kindlasti oleks tarvis ka testida rohkem erinevate parameetrite kombinatsioone.

Rakenduse kasutusvõimaluste peatükis (vt 2.1) oli mainitud, et rakendusega tulevad kaasa kaks mudelit. Üks neist on mudel reanumbriga 24 ja teine reanumbriga 16 (esialgsete algandmete ning mudeli parameetritega). Autor otsustas lisada mõlemad eemärgiga anda kasutajatele valik parematest võimalustest, mis hetkel saadaval.

3.3 Arenduse käigus tekkinud probleemid

Rakenduse idee oli algusest peale väga ambitsioonikas, seega väike arv allpool kirjeldatud probleeme on autori silmis piisavalt hea viide rakenduse kordaminekule.

Rakenduse loomine

Rakenduse koostisosade alampeatükis (vt 2.2) mainis autor, et valis programmeerimiskeeleks Java, sest oli kindel oma pädevuses antud keeles. Rakenduse loomisel aga ilmnesid teatud lüngad. Autor ei võtnud arvesse, et polnud mõnda aega Java’t programmeerimiseks kasutanud. Lisaks tulid juurde uued funktsionaalsused, nagu rakenduse serveri osa ja masinõpe, mida autor Java’s varem kasutanud pole. Seetõttu tuli rakenduse arendusaeg oodatust pikem.

Enne rakenduse serveri osa arendamist oli autor arvamusel, et ta eelnevatest serveri osa arendamiskogemustest teistes programmeerimiskeeltes piisab, kui võtta arvesse kui lihtsana oli kujutatud serveri osa loomine Spark Framework’i dokumentatsioonis. Selle asemel hakkasid serveri osa arenduse käigus tekkima erinevaid probleeme, mille lahendamine pikendas järjekordselt rakenduse arendusaega.

Autori eelnevad kogemused masinõppega tulenevad aima-python python'i programmeerimiskeele teegi [9] kasutamisest, kus võrreldes WEKA'ga teostati masinõpet natuke teistmoodi. Kuna WEKA on Java's loodud, siis sellele rakendusid ka Java reeglid. Seetõttu oli ennustamise tööle saamine küllaltki kohmakas protsess, sest Java'le omaselt oli igal elemendil oma täpne klass, mille kasutamisvajadust polnud WEKA dokumentatsioonis ja teiste kasutajate koodinäidetes tavalisest tihedamini mainitud (`FilteredClassifier`, `StringToWordVector`). Kesist dokumentatsiooni ei osanud autor ette näha, sest rakenduse eeluuringu jooksul tehtud otsingupäringute tulemused viitasid vastupidisele.

Tekstiliste väärtustega algandmete kasutamiseks masinõppe valdkonnas tavaliselt vähendatakse nende filtreeritud kuju dimensionaalsust. Rakenduse masinõppe osa arendamise käigus oli autor üritanud kasutada LSA-d (Latent Semantic Analysis), kuid ei saanud hakkama selle programmilise rakendamisega oma rakenduses, mis järjekordselt pikendas rakenduse arendusaega.

Masinõppe teostamine

Peale masinõppe tööle saamist rakenduses muutus väga ilmseks autori puudulik arusaam masinõppest. Eelnevatest masinõppe kasutamise kogemustest oli autor järeldanud, et see on väga lihtne ja ennustamiseks on tarvis kõigest algandmed mudeli jaoks mõistetavale kujule viia, mispeale ennustaks rakendus alati 100% täpsusega kõikide eestikeelsete tekstide taset eranditult.

Autor ei arvestanud, et närvivõrkude mudeli kasutamiseks on vaja enne selle parameetrid õigeks sättida. Pikalt ei olnud autor teadlik (ei saanud aru), mida mudeli ja filtrite parameetrid ja nende muutmine muudavad treeningu käigus, mistõttu mudelite loomisel kasutati katseeksitusmeetodit, mis pikendas kõvasti rakenduse arendusaega. Lisaks andsid erinevate parameetrite kombinatsioonid erinevaid tulemusi, mis tihti olid vastuolus autori eelarvamusega treenitava mudeli tulemustest, mis omakorda lisas segadust mudelite ehitamisel.

[9] https://github.com/aimacode/aima-python

Kokkuvõte

Siinse töö eesmärk oli luua rakendus, mis ennustaks eestikeelse teksti keeleoskustaset, kasutades masinõpet. Töö käigus loodi selline rakendus koos mõningate lisafunktsionaalsustega. Täpsemalt: enne ennustamist saab rakenduselt pärida ennustust käsurealt, lasta ehitada närvivõrkude mudeleid ning töödelda teksti kolmikuteks. Ehitatud närvivõrkude mudelite ennustamistulemused on jäädvustatud. Tekstide puhul oli parim tulemus 72.18% ja tekstide põhjal tehtud kolmikute alusel 71.98% (tulemused ümardatud). Autori arvates on ennustamistulemusi kindlasti võimalik täiustada, kui ühtlustada algandmeid ning kohendada mudelite parameetreid ja testida nende kombinatsioone.

Autori arvates on töö alguses püstitatud üldeesmärk täidetud. Valminud rakendus on kasutatav, kuid teatud piirangutega. Rakenduse edasiarendamiseks pakub autor järgmist.

- Rist-valideerimine – tänu sellele saaks täies ulatuses loobuda WEKA rakenduse kasutamisest ning testida mudeleid otse rakenduses.
- Dünaamiline mudeli valimine – peale rakenduse valmimist avastas autor, et mudeli valimiseks on WEKA teegis olemas võimalus lihtsalt anda käsk, kus sees mudeli nimi koos selle parameetritega. Tänu sellele saaks eemaldada ainult närvivõrkude kasutamise piirangu.
- Uurida teisi mudeleid – kui dünaamiline mudeli valik oleks rakendatud, siis võiks uurida peatükis 1.3 esitatud mudeleid ja nende ennustamistulemusi, kasutades samu või paremini töödeldud algandmeid.
- Mudeli klassi väärtuste dünaamiline valimine – hetkel on rakenduses rangelt kirjas, mis on loodava mudeli lubatud klassiväärtused. Teoreetiliselt saab neid pärida algandmetest, kuid see eeldab nende olemasolu rakenduse kasutamisel, mis omakorda lisab uue piirangu. Veel üks võimalus oleks lubada kasutajal ise kirjutada, näiteks `config.txt` faili, missugused on klassiväärtused.
- Serveri osa – hetkel on selle olemasolu autori silmis natuke küsitav, kui võtta arvesse võimalust teha ennustust otse käsurealt. Serveri osa oleks tarvis kas eemaldada või rakendada mingil paremal viisil.
- Muud tekstitöötlusviisid – hetkel on rakenduse ainuke tekstitöötlusviis kolmikute loomine. Lisana võib avada näiteks võimaluse eemaldada tekstidest teatud sõnaliigid (silmas peetud eelkõige sidesõnu).

- Parem config.txt – hetkel on `config.txt` faili sisu natuke suvalises vormingus. Oleks hea see ümber tõsta näiteks JSON vormingule.

Kasutatud kirjandus

Andmeteadus. (kuupäev puudub). *Tehislikud närvivõrgud*. Kasutamise kuupäev: 28. märts 2018. a., allikas http://andmeteadus.ee/portfolio-items/tehislikud-narvivorgud/

Cambridge English. (kuupäev puudub). *International language standards*. Kasutamise kuupäev: 27. aprill 2018. a., allikas http://www.cambridgeenglish.org/exams-and-tests/cefr/

Cortes, C., & Vladimir, V. (1995). *Support-Vector Networks.* Allikas: https://link.springer.com/content/pdf/10.1023%2FA%3A1022627411411.pdf

Decision trees. (kuupäev puudub). Kasutamise kuupäev: 28. märts 2018. a., allikas https://en.wikipedia.org/wiki/Decision_tree

Determining Bias for Neural network Perceptrons? (26. aprill 2011. a.). Kasutamise kuupäev: 25. aprill 2018. a., allikas https://stackoverflow.com/questions/5794954/determining-bias-for-neural-network-perceptrons

Eslon, P. (2014). *Adverbi sisaldavate struktuuride tekstifunktsioonidest eesti ilukirjandus- ja õppijakeeles*. – Lähivõrdlusi, 24, lk 15–46. doi:10.5128/LV24.01

Frontiers. (kuupäev puudub). *Artificial Neural Networks as Models of Neural Information Processing*. Kasutamise kuupäev: 28. märts 2018. a., allikas https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing

Hallik, V. (2016). *Eesti vahekeele korpuse klasteranalüüsi vahendite kasutamine teksti keeletaseme prognoosimisel.* Allikas: http://www.cs.tlu.ee/teemaderegister/

Kang, N. (27. juuni 2017. a.). *Multi-Layer Neural Networks with Sigmoid Function— Deep Learning for Rookies (2).* Kasutamise kuupäev: 25. aprill 2018. a., allikas https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

Karlik, B., & Olgac, A. V. (kuupäev puudub). *Performance Analysis of Various Activation Function.* Allikas:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.740.9413&rep=rep1&type=pdf

Lõo, K. (2012). The Relationship between Lexical Richness and Estonian Learners Language Proficiency Levels. Seminaritöö käsikiri. Seminar für Sprachwissenschaft. Eberhard Karls Universität Tübingen.

Lõo, K., & Vajjala, S. (2013). *Role of Morpho-Syntactic Features in Estonian Proficiency Classification.* Allikas: http://www.aclweb.org/anthology/W13-1708

Lõo, K., & Vajjala, S. (2014). *Automatic CEFR Level Prediction for Estonian Learner Text.* Allikas: http://www.aclweb.org/anthology/W14-3509

Mohan Raju, M., Srivastava, R., Bisht, D. C., Sharma, H. C., & Kumar, A. (2011). *Development of Artificial Neural-Network-Based Models for the Simulation of Spring Discharge.* Allikas: https://www.hindawi.com/journals/aai/2011/686258/

Nielsen, M. (detsember 2017. a.). *Using neural nets to recognize handwritten digits*. Kasutamise kuupäev: 25. aprill 2018. a., allikas http://neuralnetworksanddeeplearning.com/chap1.html

Oracle. (kuupäev puudub). *https://www.java.com/en/download/faq/whatis_java.xml*. Allikas: https://www.java.com/en/download/faq/whatis_java.xml

Rajaraman, A., & Ullman, J. (2011). *Data mining.* Allikas: http://i.stanford.edu/~ullman/mmds/ch1.pdf

Readable.io. (kuupäev puudub). *How CEFR can boost your content marketing*. Kasutamise kuupäev: 27. märts 2018. a., allikas https://readable.io/blog/how-cefr-can-boost-your-content-marketing/

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach.* New Jersey: Pearson Education, Inc.

Schmid, H. (kuupäev puudub). *TreeTagger - a part-of-speech tagger for many languages*. Kasutamise kuupäev: 28. märts 2018. a., allikas http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

Smith, S. (kuupäev puudub). *Neural Network Architecture*. Kasutamise kuupäev: 25. aprill 2018. a., allikas http://www.dspguide.com/ch26/2.htm

Spark (software). (kuupäev puudub). Kasutamise kuupäev: 29. märts 2018. a., allikas https://en.wikipedia.org/wiki/Spark_(software)

Statistics How To. (25. oktoober 2016. a.). *What is Root Mean Square Error (RMSE)?* Kasutamise kuupäev: 28. märts 2018. a., allikas http://www.statisticshowto.com/rmse/

TF-IDF in the StringToWordVector Filter. (23. aprill 2014. a.). Allikas: http://weka.8497.n7.nabble.com/TF-IDF-in-the-StringToWordVector-Filter-td30888.html

The University of Waikato. (kuupäev puudub). *Class MultilayerPerceptron*. Kasutamise kuupäev: 25. aprill 2018. a., allikas http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html

The University of Waikato. (kuupäev puudub). *Weka 3: Data Mining Software in Java*. Allikas: https://www.cs.waikato.ac.nz/ml/weka/

Summary

Title: Developmen of A Language Skill Prediction Software Using Machine Learning.

The purpose of this thesis is to develop a language skill prediction software that uses machine learning to make predictions.

In the first chaper author reviews similar software, previous papers, that are relatable to this thesis, machine learning models, that are often used for classifying texts, and describes the machine learning library, that he used in his software.

In the second chapter author describes every aspect of created software: what it can do, it's components, usage, structure and how neural networks model works in the software.

In the third chapter author describes where the created software can be used, displays results from various model creating attempts and reviews the problems he had while creating this software.

In conclusion the erected goal of creating a language skill predicting software that uses machine learning was met. Due to author being somewhat inexperienced in the field of machine learning, the best result that was achieved during the making of this thesis was 72.1827%, though, in authors opinion, improving it should not be much of a challange.

LISAD

Lisa 1. Tekstidega algandmete näide

```
@relation R_data_frame

@attribute tekst string
@attribute keeletase {A,A1,A2,B,B1,B2,C,C1,C2}

@data
'Suvel ma lugesin läbi ajalehe &quot;.... &quot;, kus oli väg
'Ma küsitlesin 4 inimest meie klassist: 2 tüdrukut ja 2 poiss
'Esmaspäeval, kui ma läksin koolist kodusse, nägin ühe kaupl
'I. Pange sulgudes olevad sõnad õigesse vormi! Piret Toomet
'EESTI KEELE TEST  I. Pange sulgudes olevad sõnad õigesse vo
'EESTI KEELE TEST  I. Pange sulgudes olevad sõnad õigesse vo
'I. Pange sulgudes olevad sõnad õigesse vormi! Piret Toomet
'SissejuhatusSelle töö teemaks on traditsioon ja nüüdisaeg.
'   Ühel päeval sain eesti keele õpetajalt teada, et vene õpp
'Ma tahan väga võistelda, mitte alati, aga sageli. Mitte sel
'Selles kirjas Teile ma tahaks kirjeldada ennast, et mina pa
'Miks just mina pean olümpiaadile pääsema ja oma kooli esind
' Lugupeetud olümpiaadi korraldajad ja hindamiskomisjoni lii
' Mina olen 17-aastane ja tahan jutustada, miks minust parem
'Ausalt rääkides ma mitte kunagi ei mõelnud, et ma kunagi ha
'Ma ei oleks kunagi arvanud, et võiksin osa võtta eesti keel
'Minu nimi on Marika, ma olen seitsmeteistkümneaastane ja õp
'Seda olümpiaaditööd  kirjutavad paljud õpilased erinevatest
'Soovin osaleda vene õppekeelega koolide õpilastele mõeldud
'Õpetaja ütles: \"Tule ja osale olümpiaadil! \". Mina mõtles
'Minu meelest ei ole mõtetki kaua mõtelda ja arutleda, kes p
'Minu nimi on .. ja tahan osaleda eesti keele olümpiaadil. S
'Kindlasti iga inimene selles suures maailmas arvab, et tema
'Ma ei ole väga kindel, aga ma arvan, et minust paremat pole
'Võit olümpiaadil on hea võimalus saada endale tasuta koht E
'Minu nimi on Natalja. Ma õpin eesti keelt alates lasteaiast
'Lugupeetud kohtunikud! Soovin kandideerida Tallinna Ülikool
```

Lisa 2. Kolmikutega ja ainult nelja keeleklassiga algandmete näide

```
@relation Minu_andmestik

@attribute ngramid string
@attribute keeletase {A2,B1,B2,C1}
@data
'PSV SVG VGS GSS SSV SVZ VZJ ZJG JGS GSS SSS SSV SVG VGS GSS SSZ
'PVD VDV DVZ VZD ZDD DDZ DZJ ZJD JDZ DZD ZDD DDZ DZJ ZJV JVZ VZJ
'DDP DPV PVS VSV SVJ VJP JPS PSV SVZ VZJ ZJD JDD DDS DSZ SZP ZPV
'DVP VPD PDD DDV DVA VAZ AZJ ZJP JPD PDV DVS VSV SVZ VZJ ZJJ JJP
'PVV VVD VDA DAZ AZJ ZJV JVS VSV SVG VGS GSS SSZ SZD ZDJ DJP JPG
'PSV SVH VHZ HZP ZPV PVA VAJ AJV JVS VSH SHS HSZ SZP ZPD PDD DDV
'PSV SVP VPS PSA SAS ASZ SZS ZSP SPS PSV SVA VAZ AZJ ZJP JPD PDV
'VVG VGS GSS SSS SSA SAG AGS GSS SSZ SZP ZPV PVA VAD ADN DNS NSS
'SVZ VZZ ZZV ZVJ VJS JSS SSZ SZZ ZZP ZPV PVZ VZJ ZJD JDD DDZ DZD
'PSV SVZ VZJ ZJV JVV VVG VGS GSS SSZ SZV ZVV VVS VSS SSJ SJP JPV
'DPS PSP SPA PAS ASV SVZ VZJ ZJP JPC PCV CVV VVZ VZJ ZJP JPV PVA
'PVV VVD VDA DAZ AZJ ZJP JPV PVZ VZJ ZJP JPC PCV CVZ VZD ZDP DPS
'SSV SVA VAS ASV SVP VPS PSS SSH SHS HSZ SZP ZPP PPZ PZP ZPS PSK
'PSV SVH VHZ HZP ZPV PVG VGS GSV SVS VSZ SZP ZPS PSV SVA VAP APD
'IZP ZPK PKZ KZJ ZJV JVS VSS SSJ SJV JVP VPS PSZ SZV ZVP VPS PSZ
'DZD ZDP DPS PSV SVP VPZ PZJ ZJV JVD VDD DDP DPZ PZJ ZJD JDP DPS
'VHH HHH HHS HSS SSZ SZP ZPS PSV SVS VSO SOS OSZ SZV ZVD VDV DVH
'NSN SNS NSN SNA NAS ASN SNS NSS SSN SNA NAS ASN SNY NYA YAS ASN
'HSS SSZ SZH ZHS HSS SSZ SZS ZSZ SZV ZVH VHS HSZ SZA ZAH AHZ HZD
'PSS SSV SVO VOS OSS SSD SDS DSS SSS SSV SVD VDA DAZ AZD ZDA DAV
'PZY ZYY YYZ YZV ZVH VHH HHD HDO DOS OSZ SZP ZPS PSV SVG VGS GSJ
'VDH DHZ HZV ZVD VDH DHD HDS DSZ SZK ZKS KSZ SZZ ZZV ZVK VKN KNS
'AGS GSV SVN VNY NYA YAJ AJV JVN VNY NYZ YZA ZAG AGS GSV SVN VNY
'POS OSV SVD VDZ DZD ZDV DVN VNJ NJA JAJ AJD JDA DAZ AZP ZPS PSH
'POS OSS SSV SVP VPZ PZD ZDV DVP VPS PSH SHS HSA SAS ASO SOS OSZ
'PSV SVH VHJ HJP JPS PSV SVZ VZV ZVH VHS HSS SSZ SZV ZVH VHZ HZH
'PVA VAS ASS SSK SKD KDZ DZH ZHV HVS VSS SSS SSJ SJV JVP VPS PSZ
'VDA DAS ASV SVP VPS PSZ SZJ ZJP JPV PVD VDD DDD DDZ DZJ ZJP JPV
```