LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

PACIENTU KOMENTĀRU NOLŪKA NOTEIKŠANA, IZMANTOJOT MAŠĪNMĀCĪŠANOS

BAKALAURA DARBS

Autors: **Valdis Aglonietis**

Studenta apliecības Nr.: **va16014**

Darba vadītājs: **Dr.med. Dzintars Mozgis, Rīgas Stradiņa universitāte, Asociētais profesors**

RĪGA 2020

ANOTĀCIJA

Bakalaura darba mērķis ir automatizēt procesu aptauju atvērtā tipa jautājumu nolūka noteikšanai, izmantojot uzraudzīto mašīnmācīšanos, kā arī pierādīt izvēlētā risinājuma veiktspēju. Programmatūras risinājums paredzēts pacientu komentāru nolūka noteikšanai.

Bakalaura darbs sastāv no vārdnīcas, ievada, mašīnmācīšanās teorijas, eksistējošiem risinājumiem, mašīnmācīšanās bibliotēku analīzi, datu apstrādes, modeļa parametru izklāsta, modeļa izstrādes, rezultātiem, secinājumiem, izmantotās literatūras un pielikumiem.

Bakalaura darbā aprakstītas mašīnmācīšanās bibliotēkas, ko iespējams izmantot risinājuma izstrādei. Ar konkrētu izvelētu mašīnmācīšanās bibliotēku aprakstīta mašīnmācīšanās modeļa izstrāde. Bakalaura darba beigās tika novērtēti vairāki varianti risinājumam.

Darba datu iegūšanai un apstrādei tika izmantota programmēšanas valodas *Python* bibliotēka *Keras*.

Kursa darbs ir uzrakstīts latviešu valodā, satur 5 tabulu, 6 attēlus. Darba apjoms kopā ar pielikumiem ir 38 lappuses, darba izstrādei tika izmantoti 45 literatūras avoti.

Atslēgvārdi: Mašīnmācīšanās, Python, Uzraudzītā mācīšanās, mašīnmācīšanās bibliotēkas, mašīnmācīšanās modelis

ABSTRACT

Determining the Purpose of Patient Comments Using Machine Learning

The aim of the bachelor thesis is to automate the process of determining the intention of open-ended survey questions, using machine learning, as well as to demonstrate the performance of the chosen solution. The software solution is designed to determine the intention of patient comments.

The bachelor thesis consists of dictionary, introduction, theory of machine learning, existing solutions, machine learning library analysis, data processing, outline of the model parametrs, model development, results, conclusions, list of references and appendices.

The bachelor's thesis describes machine learning libraries, which can be used to develop a solution. The development of a machine learning model is described with a specifically selected machine learning library. At the end of the bachelor thesis, several options for the solution were evaluated.

The *Keras* library of the *Python* programming languages was used to obtain and process data for the bachelor thesis.

The bachelor thesis is written in Latvian, contains 5 table, 6 images. The volume of the work with the attachments is 38 pages, 45 literature sources are used for the development of the work.

Keywords: Machine learning, Python, Supervised learning, Machine learning libraries, Machine learning model

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

Dabiskā valoda – Cilvēku radīta valoda, kas attīstījusies dabiskā ceļā ar cilvēku komunikāciju bez apzinātas plānošanas. Piemēram, latviešu vai angļu valoda.

Keras - Dabiskās valodas apstrādes ietvars

Python - Programmēšanas valoda

Nolūks - komentāra pazīme, kas nosaka vai komentārs ir pozitīvs vai negatīvs.

BKUS – VSIA "Bērnu klīniskā universitātes slimnīca"

PREM – aptauju sistēma

Datu transformācija – datu izmainīšana pēc noteiktiem algoritmiem

Datu agregācija – datu formēšana grupās

GPU – Grafiskais procesors, datora komponente grafiku apstrādei

CPU – Centrālais procesors, datora komponente kalkulāciju veikšanai

Amazon – Elektroniskās komercijas platforma

Twitter – Sociālais tīkls

Emocijzīme – sejas izteiksmes attēls, ko lieto emociju attēlošanai teksta formātā.

Regresija – datu analīzes process, lai novērtētu datu sasaisti

Klasifikācija – objektu iedalīšana grupās

SVN – *Support Vector Machines* [17] algoritms priekš datu klasifikācijas un regresijas līdzīgu datu klasifikācijai

NB – *Naive Bayes* [18] algoritms priekš datu klasifikācijas, kas datus izvērtē atsevišķi

DT – *Decision Tree* [19] algoritms priekš datu klasifikācijas, kas izmanto koka struktūru lēmuma pieņemšanai

IMDB – filmu atsauksmju platforma

Rotten Tomatoes – filmu atsauksmju platforma

LSTM – *Long Short-Term Memory* algoritms priekš dabiskās valodas apstrādes

CNN – *Convolutional neural network* algoritms vizuālu attēlu klasificēšanai

Mākslīgais intelekts – mašīnu saprātīgas izturēšanās, apmācības un pielāgošanās zinātnes nozare

Bibliotēka – Neatkarīgs programmatūras koda kopums

Google –Tehnoloģija kompānija

Simboliskais cikls – cikls, kur darbību ir iespējams izpildīt noteiktu skaitu reižu.

Sesijas turētājs – programmatūras daļa, kas satur informāciju par konkrēta izsaukuma esošo stāvokli

Python – programmēšanas valoda, ko pārsvarā izmanto datu apstrādei un analīzei

Atgriezeniskās saderība – programmatūras vecāku versiju atbalstīšana pie atjauninājumiem

Facebook – sociālais tīkls

Neironu tīkli – daudz dažādu mašīnmācīšanās algoritmu ietvars

Torch – *Python* programmēšanas valodas bibliotēkas priekš neironu tīklu rēķināšanas

Modularitāte – programmatūŗas sadalīšanā pa funkcionālām komponentēm tā, lai izmainot vienu komponent, citas komponentes netiktu ietekmētas.

Vektors – Matemātisks objekts, kuram ir noteikts garums un virziens [40]

Tenzors – Matemātisks objekts, kas ir vispārināts vektors [39]

One-hot šifrēšana - skaitļa pārveide uz masīvu, kur katra iespējamajai skaitļa vērtībai ir individuāls indekss un vērtīá ir 1 vai 0.

GDPR – Personas Datu aizsardzības regula [7]

IEVADS

Ārstniecības iestādēs svarīgs mērķis ir pacienta pieredzes uzlabošana, lai pacients justos uzklausīts, sadzirdēts, saņemtu nepieciešamo palīdzību un dotos mājās ar pēc iespējas pozitīvāku pieredzi. Pacientu pieredze ir būtiska veselības aprūpes kvalitātes rezultātu un procesa dimensiju novērtēšanai, turklāt veselības aprūpes specifikas dēļ „pieredze" šķirama no „apmierinātības", jo situācijās, kad neveselība vai nespēja ir neatgriezeniskas, apmierinātība būs nosacīta un apšaubām. Lai iegūtu atgriezenisko saiti par pacienta pieredzi tiek izmantoti dažādi informācijas līdzekļi un viens no tiem ir unikālās tiešsaistes aptaujas. Aptaujās esošos atvērtā tipa jautājumus nav iespējams automātiski viegli sagrupēt, tāpēc tie tiek apskatīti katrs individuāli un iedalīti pozitīvajās, negatīvajās un neitrālajās atsauksmēs. Ierakstu caurlūkošana un to sistematizēšana ir laikietilpīga, kas rada nozīmīgu un neefektīvu cilvēkresursu noslogojumu ārstniecības iestādēs. Šis jautājums tika aktualizēts OECD valstu veselības ministru sanāksmē 2017. gada janvārī, kad tika izvirzītas divas jaunas prioritātes – izveidot starptautiski salīdzināmus pacientu ziņotos pieredzes *(Patient-Reported Experience Measure – (PREM))* un pacientu ziņotos rezultātu mērījumus *(Patient-reported outcome measures – (PROM)).*

*(*OECD, Recommendations to OECD Ministers of Health from the high level reflection group on the future of health statistics, Strengthening the international comparison of health system performance through patient-reported indicators, January, 2017, https://www.oecd.org/els/health-systems/Recommendations-from-high-level-reflection-group-on-the-future-of-health-statistics.pdf)

2018. gadā tika aizsākta t.s. PaRIS (*Patient Reported Indicators Surveys*) iniciatīva, kurā no Latvijas puses pašlaik vēl tikai PREMs sadaļā ir iesaistījies Slimību profilakses un kontroles centrs un VSIA „Bērnu klīniskā universitātes slimnīca" (BKUS). Saskaņā ar Eiropas Savienības struktūrfondu piešķiršanas nosacījumiem Latvijai ir jāievieš sistemātiska un pastāvīga pacientu ziņotās pieredzes mērīšana (PREMs) visās ārstniecības iestādēs, kas arī nosaka tēmas aktualitāti.

(REGULATION (EU) 2017/825 of 17 May 2017 on the establishment of the Structural Reform Support Programme for the period 2017 to 2020 and amending Regulations (EU) No 1303/2013 and (EU) No 1305/2013)

Bakalaura darba mērķis ir automatizēt procesu atvērtā tipa jautājumu nolūka noteikšanai, izmantojot mašīnmācīšanos. Pacientu atsauksmes tiks apstrādātas lietošanai mašīnmācīšanās programmatūrās. Izmantojot iegūtos atsauksmju datus tiks trenēts mašīnmācīšanās modelis, lai ar to varētu noteikt jaunu atsauksmju nolūku. Izstrādāto modeli būs iespējams izmantot, lai noteiktu komentāru nolūku reālajā laikā.

Bakalaura darba sākumā tiek aprakstīta pētāmā problēma un mašīnmācīšanās jēdziens. Tālāk tiek apskatīti jau eksistējošie risinājumi, kas ir saistīti ar pētāmo problēmu. Tiek apskatīti risinājumi angļu valodā un turku valodā. Tālāk tiek apskatīti populārākie mašīnmācīšanās ietvari, kurus būtu iespējams izmantot, lai izstrādātu mašīnmācīšanās modeli. Apskatītie rīki tiek salīdzināti un tiek izvēlēts potenciāli labākais rīks mašīnmācīšanās modeļa izstrādei. Mašīnmācīšanās modeļa izstrādei dati tika iegūtie no Bērnu klīniskās universitātes slimnīcas (turpmāk tekstā kā *BKUS*) sistēmas *PREM* aptaujām no 2018. līdz 2020. gadam. Tālāk apskatīts iegūto datu formatēšanas process un ar to saistītās problēmas. Tālāk tika apskatīti izvēlētā rīka mašīnmācīšanās modeļa izstrādes principi un iespējamie mainīgie parametri, kas ietekmētu mašīnmācīšanās modeļa trenēšanu un precizitāti. Bakalaura darba praktiskajā daļā tiek izstrādāts mašīnmācīšanās modelis, izmantojot iepriekš aprakstītos mainīgos parametrus. Bakalaura darba noslēgumā tiek izvēlēts un implementēts mašīnmācīšanās modelis ar augstāko precizitāti.

Bakalaura darba mērķis: Izveidot mašīnmācīšanās modeļa prototipu ar 95% vai vairāk procentu precizitāti pacientu pieredzes komentāru nolūka noteikšanā latviešu valodā.

Bakalaura darba hipotēze: Ir iespējams izveidot mašīnmācīšanās modeli ar 95% vai vairāk procentu precizitāti pacientu pieredzes komentāru nolūka noteikšanā.

1. MAŠĪNMĀCĪŠANĀS

Pirmās nodaļas mērķis ir sniegt ieskatu par mašīnmācīšanās jēdzienu un tās iespējamajiem guvumiem. Tiks apskatīts mašīnmācīšanās jēdziens, vēsture un pielietojuma veidi

1.1. Kas ir mašīnmācīšanās

Mašīnmācīšanās ir mākslīgā intelekta apakš nozare, kas ar statistiku palīdz lielos datu apjomos atrast sakarības. Mašīnmācīšanā tiek izmantoti algoritmi, kas izpilda darbību, novērtē darbības precizitāti, novērtē darbības kļūdu pret sagaidāmo rezultātu un mēģina vēlreiz izpildīt darbību, mainot iekšējos parametrus tādā veidā, lai uzlabotos precizitāte.

Kopumu, kas satur darbības un dažādus iekšējos parametrus sauc par modeli. Modelis ir kā funkcija, kas no konkrētiem datiem, izvada rezultātu. Modelis atšķiras no algoritmiem ar to, ka tā algoritms nav strikti noteikts un to ir iespējams mainīt. Vienīgais, kas tiek noteikts ir tā mācīšanās metode. Modelim tiek noteikta konkrēta mācīšanās metode. Precizitātes uzlabošanai modelis veic darbību desmitiem, simtiem vai pat miljoniem reižu un katru reizi izmaina savu iekšējo algoritmu, lai tas būtu tuvāks sagaidāmajam rezultātam. Kad darbība ir atkārtota N reizes un parametri rezultāta noteikšanai ir pielāgoti, tad ir iespējams prognozēt iznākumu ar precizitāti [1].

Konkrētāk mācīšanās metodes var eksistēt vairākas un tās var tikt nodalītas slāņos. Slāņi var būt dažādi. Tie var saturēt datu transformācijas algoritmu, datu agregācijas algoritmu vai kāda cita tipa algoritmu, kas manipulē ar datiem. Slāņu esamība ļauj veidot kompleksus modeļus un pielietot tos dažādākajās reālās dzīves sfērās.

1.2. Vēsturisks atskats

Mašīnmācīšanās termins pirmo reizi tika lietots 1952. gadā, kad to izdomāja datorinženieris Artūrs Samuels. Termins sākās ar datorprogrammas uzrakstīšanu dambretes spēlei, taču atmiņas trūkuma dēļ bija nepieciešams izveidot algoritmu, kurš spētu noteikt pareizos gājienus. Programma vēlāk tika pilnveidota tādā veidā, ka tā, atceroties iepriekšējos gājienus un uzkrājot pieredzi, spēja pati noteikt optimālāko risinājumu [2].

20. gadsimta 60. gados tika atklāti vairāku līmeņu slāņi, kas pavēra jaunu pasaule neironu tīklu pētniecībā. Tika atklāts, ka mācīšanās algoritmi darbojas labāk, ja tie ir noslāņoti jeb ja visi dati neatrodas vienā slānī – datu sadalījums pa slāņiem ir mūsdienu mašīnmācīšanās pamats [3].

90. gados interese par neironu tīkliem, kas ir mašīnmācīšanās virsklase, bija visai zema un to tālākai pētniecībai netika pievērsta plaša uzmanība, taču zinātnieks Geofs Hintons (*G*eoff *Hinton*) pieturējās pie to pētniecības. 2006. gadā tika publicēts darbs "*a fast learning algorithm for deep belief nets*"), kas kļuva par pavērsiena punktu mašīnmācīšanā [4],[5]. Šajā darbā tika aprakstīta slāņu pilnīga nodalīšana tā, lai kāda slāņa dati nekomunicēti ar citu slāņu datiem, bet lai komunikācija no slāņa uz slāni notiktu bez to iekšējo datu komunikācijas. Komunikācija starp slāņu iekšējiem datiem sagādāja grūtības. Slāņus bija grūti koriģēt, jo tie bija saistīti viens ar otru. Minētais darbs atrisināja sarežģītās koriģēšanas problēmu un ļāva izmainīt slāņus atsevišķi neietekmējot citus slāņus.

2009. gadā tika publicēts darbs "*Large-scale Deep Unsupervised Learning using Graphics Processors*" [6], kas ierosināja izmantot *GPU*, nevis *CPU*. Līdz tam primāri mašīnmācīšanā tika izmantots *CPU*, bet tas ir salīdzinoši lēns lieliem datu apjomiem liela izmēra programmatūrās. Darbā ierosinātā *GPU* izmantošana programmatūras apmācīšanas laiku samazināja no vairākām nedēļām uz apmēram vienu dienu. Apmācības ilgums varēja samazināties pat 72 reizes [6],[4]. Mašīnmācīšanās ātrumam palielinoties vairākas reizes, tā izmantošana strauji pieauga un tagad mašīnmācīšanās tiek izmantota visās dzīves sfērās.

1.3. Kategorijas

Mašīnmācīšanās tiek iedalīta vairākās kategorijās pēc to pieejas, apmācības veida, risināmās problēmas un mērķa.

1.3.1. Uzraudzītā mācīšanās

Uzraudzītās mašīnmācīšanās algoritms veido matemātisko modeli no iepriekš definētas datu kopas, kas satur gan datus, gan sagaidāmos rezultātus [9]. Bakalaura darba ietvaros tiks izmantota uzraudzītā mašīnmācīšanās. Uzraudzītajai mašīnmācīšanai nepieciešami lieli datu apjomi, kurus ir jāsagatavo ar jau zināmiem iznākumiem. Mašīnmācīšanās modelis izmanto ievaddatus no sagatavotajiem datiem un prognozē iznākumu. Ja iznākums nav pareizs, tad modeļa iekšējais

algoritms tiek koriģēts. Ar lielākiem datu apjomiem ir iespējams iegūt augstāku precizitāti. Problēmas uzraudzītas mācīšanās gadījumā ir datu nepietiekamība vai pārtrenēšanās, kur modelis tiek uztrenēts uz dotajiem datiem un nav spējīgs precīzi apstrādāt neredzētus datus [9].

1.3.2. Neuzraudzītā mācīšanās

Neuzraudzītā mašīnmācīšanās algoritms veido matemātisko modeli no iepriekš definētas datu kopas, kas satur tikai datus bez sagaidāmajiem rezultātiem. Bez sagaidāmajiem rezultātiem mašīnmācīšanās modelis pats izlemj datu struktūras un saiknes un var veidoties iepriekš neredzētas struktūras vai saiknes. Neuzraudzītā mašīnmācīšanās tiek izmantota, lai atklātu minētās datu struktūras un starp tām esošās saiknes, lai tos būtu iespējams grupēt kategorijās [9]. Minētā mācīšanās metode tiek pielietota gadījumos, kad cilvēks nav spējīgs atrast datu sakarības [8].

1.3.3. Daļēji uzraudzītā mācīšanās

Daļēji uzraudzītā mācīšanās ir uzraudzītās mācīšanās un neuzraudzītās mācīšanās savienojums. Sākumā mašīnmācīšanās modelim tiek iedoti sagatavoti dati ar sagaidāmajiem rezultātiem, bet pēc tam tiek iedoti dati bez sagaidāmajiem rezultātiem. Minētā mācīšanās metode tiek izmantota gadījumos, kad ir lieli datu apjomi. Datus ir nepieciešams apstrādāt cilvēkam un cilvēka darbs ir dārgs un laikietilpīgs. Pārsvars datu tiek apstrādāti bez sagaidāmajiem rezultātiem, bet daļā datu sagaidāmo rezultātu nosaka cilvēks [8].

1.3.4. Stimulētā mācīšanās

Stimulētā mācīšanās veido matemātisko modeli, kas mijiedarbojas ar apkārt esošo vidi, kas ir dinamiska pozitīvu vai negatīvu stimulu veidā. Modelis tiek ielikts dinamiskā vidē un saņem informāciju par to. Atkarībā no modeļa rīcības jeb izvad-datiem, modelis saņem atbildi no vides, kas dod atalgojumu. Atalgojuma parametri var variēties. Modelis var tikt būvēts, lai palielinātu atalgojumu vai samazinātu kādu no atalgojuma parametriem, piemēram, risku. Stimulētās mācīšanās pazīstamākie piemēri ir pašbraucošās mašīnas vai šaha datorprogrammas [8].

2. EKSISTĒJOŠIE RISINĀJUMI

Teksta nolūka analīze ir angļu valodā ir ļoti plaši pētīta tēma. Internetā ir pieejamas datu kopas angļu valodā, no kurām ir iespējams izstrādāt nolūka noteikšanas modeļus. Pieejamie dati ir mērāmi miljonos. Piemēram, *Amazon* produktu atsauksmju datu kopa, kuras lielums ir 142.8 miljoni datu [11], vai *Twitter* komentāru datu kopa, kuras lielums ir 1.6 miljoni [12]. Savukārt, citās valodās teksta nolūka analīzē ir salīdzinoši maz pētījumu.

Pirmais eksistējošais risinājums ir "*Twitter Sentiment Classification using Distant Supervision*" [13]. Darbā aprakstīts, kā ar *Naive Bayes (*Turpmāk tekstā kā *NB)*, *Maximum Entropy* un *Support Vector Machines* (Turpmāk tekstā – *SVM)* algoritmiem iespējams apstrādāt komentārus ar precizitāti virs 80 procentiem. Darbs izceļas ar to, ka var apstrādāt arī komentārus ar emocijzīmēm. Darbā aprakstīts par to, ka nolūka noteikšanu un algoritma izstrādi apgrūtina tas, ka teksti ir tikai 140 burtu gari. Parasti tiek izmantotas datu kopas no atsauksmju platformām kā *IMDB* vai *Rotten Tomatoes*, jo tur pieejami gari teksti, kuros tiek piedomāts pie sintakses un gramatiskās pareizības. *Twitter* komentāros, savukārt, netiek piedomāts pie sintakses un komentāra jēga tiek izpausta pēc iespējas lakoniskāk, kas apgrūtina nolūka analīzi.

Otrais eksistējošais risinājums ir "*Sentiment analysis algorithms: evaluation performance of the Arabic and English language*"[14]. Darbā tiek salīdzināti Angļu valodas un Arābu valodas nolūka noteikšanas modeļi, izmantojot līdzīgus algoritmus. Angļu valodai ar *Decision tree (*Turpmāk tekstā – *DT*) algoritmu nolūks tika noteikts ar 97.16% precizitāti un ar *NB* algoritmu nolūks tika noteikts ar 89.52% precizitāti. Savukārt, Arābu valodai ar *DT* algoritmu nolūks tika noteikts ar 50.76% precizitāti un ar *NB* algoritmu nolūks tika noteikts ar 54.43% precizitāti. Ir redzama krasa atšķirība starp nolūka noteikšanas precizitātēm angļu un arābu valodās.

Trešais eksistējošais risinājums ir "*Baselines and Bigrams: Simple, Good Sentiment and Topic Classification*" [15],[16]. Darbā aprakstīts kā var izmantot *NB* un *SVM* algoritmus, lai iegūtu modeli, kas nosaka nolūku ar 92.5% precizitāti. Darbā izmantota *IMDB* komentāru datu kopa, kas satur daudz datu. Eksistējošais risinājums izmanto iepriekš sagatavotus vārdu vektorus, kas ir klasificēti angļu valodai.

Ceturtais eksistējošais risinājums ir "*The Impact of NLP on Turkish Sentiment Analysis*" [10]. Darbā apraksts par nolūka analīzes precizitātes uzlabošanu par 5%, sasniedzot 78.83% precizitāti. Darbā ir izmantots *SVM* algoritms.

Piektais eksistējošais risinājums ir "*DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis*" [21]. Darbā apskatīts *Long Short-Term Memory* (Turpmāk tekstā LSTM) algoritms ar dažādiem uzmanības mehānismiem. Pirms *LSTM* algoritma pielietošanas datiem tika veikta teksta sintakses sakārtošana, sadalīšana un transformācija par skaitļiem, izmantojot skaitļu-vārdu vārdnīcas.

No aplūkotajiem eksistējošajiem risinājumiem var secināt, ka angļu valodā ir iespējams izstrādāt modeļus ar augstu nolūka noteikšanas precizitāti virs 80%, taču citās valodās izstrādāt līdzīgu modeli ir grūti. Pieļaujams, ka latviešu valodā izstrādāt mašīnmācīšanās modeli ar precizitāti virs 80% pašlaik pieejamo risinājumu ietvaros būtu sarežģīti.

3. MAŠĪNMĀCĪŠANĀS BIBLIOTĒKU ANALĪZE

Mašīnmācīšanās modeļa izveidei tiks izmantots mašīnmācīšanai paredzēts ietvars, lai programmatūras kodu nebūtu jāraksta no paša sākuma. Tika aplūkoti 4 populārākie mašīnmācīšanās ietvari.

3.1. Tensorflow

Tensorflow ir atvērtā koda mākslīgā intelekta bibliotēka, kas izmanto datu plūsmas grafikus, lai būvētu modeļus. Tā ļauj programmētājiem izveidot liela apjoma neironu tīklus vairākos līmeņos. Ietvaru galvenokārt izmanto klasifikācijai un regresijai [22].

Bibliotēkai ir ļoti labs grafiskais atbalsts un ļoti laba dokumentāciju, kur var atrast visu nepieciešamo informāciju, ieskaitot atjauninājumu detalizētus aprakstus[23].

3.1.1. *Tensorflow* priekšrocības

Bibliotēka piedāvā labāko datorgrafikas vizualizāciju, kāda pieejama starp līdzvērtīgām bibliotēkām, bet vizualizācijas iespējas ir pat plašākas nekā citām bibliotēkām [24].

Pateicoties tam, ka bibliotēkas izstrādātājs ir *Google*, tā ir ievērojami pārāka ar augstāku veiktspēju, ātriem un regulāri publicētiem atjauninājumiem un jaunām versijām ar jaunu funkcionalitāti. Tas nodrošina jaunāko algoritmu pieejamību bibliotēkā [24].

Bibliotēka ļauj izpildīt atsevišķus modeļa slāņus, kas ļauj ievietot un atgūt datus atsevišķi no kopējā modeļa. Atsevišķu slāņu izmantošana nodrošina ērtu atkļūdošanas iespēju [24].

3.1.2. *Tensorflow* trūkumi

Viena no problemātiskākajām funkcionalitātēm ir simbolisko ciklu neesamība. Bibliotēka neatbalsta simboliskos ciklus, kas ir viena no svarīgākajām prasībām mainīgo garumu sekvencēs. Eksistē funkcija "*bucketing"*, kuru izmantojot, ciklu neesamību ir iespējams apiet, taču tai ir citi ierobežojumi, piemēram, atpakošana [24].

Programmēšanas kods ir sadrumstalots un bieži grūti izprotams – meklējot kādu darbību tiek piedāvāti vairāki ļoti līdzīgi varianti, radot neizpratni par pareizo risinājumu [25].

Programmēšanas kodam ir augsts sarežģītības līmenis. Katru aprēķinu nepieciešams izsaukt no sesijas turētāja (*session handler*). Tas rada situāciju, kur bibliotēkas izmantošana ir līdzvērtīga vienas valodas izmantošanai citā valodā. Nav iespējams izmantot darbību bez bibliotēkas ekvivalenta, līdz ar to nav iespējams rakstīt pat tīru *Python* kodu, neskatoties uz to, ka *Python* ir atbalstītā valoda. Nereti koda papildus nosacījumi nav minēti dokumentācijā un nav iespējams atrast par tiem informāciju. Pastāv augsts kļūdas risks, jo, piemēram, neizdevušās modeļa trenēšanas gadījumā netiek padots kļūdas ziņojums [25].

Lai gan jaunas versijas tiek izdotas regulāri reizi vienā vai divos mēnešos, kas ir priekšrocība, jaunās versijas nereti izraisa atgriezeniskās saderības (*backwards compatibility*) problēmas [25].

3.2. Pytorch

Pytorch ir atvērtā koda mašīnmācīšanās bibliotēka, kuras pamatā ir *Torch* bibliotēka. Bibliotēku izmanto tādām lietojumprogrammām, kā datora redze un dabiskās valodas apstrāde. bibliotēkas galvenais izstrādātājs ir *Facebook* mākslīgā intelekta pētniecības laboratorija, 2017.gada septembrī bibliotēka kļuva par atvērtā koda bibliotēku [26].

Bibliotēku izmato galvenokārt tīklu arhitektūras, piemēram, *CNN*, *LSTM* un citos augsta līmeņa algoritmos. To galvenokārt izmanto pētniecībā, uzņēmējdarbībā, un mašīnmācīšanās un mākslīgā intelekta attīstības nolūkos [27].

Pytorch ātri ieguva popularitāti pateicoties tam, ka tas ļāva mašīnmācīšanās pētniekiem salīdzinoši ar citiem rīkiem, vieglākā veidā, būvēt sarežģītas arhitektūras [28].

3.2.1. *Pytorch* priekšrocības

Daudzveidīgas modulāras komponentes, kuras ir viegli komplektēt. Atvieglota iespēja lietotājam pašam izveidot slāņu tipus un palaist tos uz GPU. Pieejami vairāki iepriekš apmācīti modeļi [28].

3.2.2. *Pytorch* trūkumi

Pārsvarā lietotājam nepieciešams pašam rakstīt mācīšanās kodu. [28] Dokumentācija ir nepilnīga [28].

Pytorch no lietotāju puses ir uzskatāmi dokumentēts kā ļoti lēns. Vairāki lietotāji ir novērojuši un dokumentējuši to, ka *Pytorch* ar katru nākamo versiju ir kļuvis aizvien lēnāks, iespējams tas noticis iekšējās optimizācijas trūkuma dēļ. Tas norāda uz programmatūras koda stabilitātes trūkumu. [29]

Pytorch ir ļoti grūti atrast informāciju. Pat meklējot priekšrocības un trūkumus radās grūtības atrast pietiekami daudz informācijas.

3.3. TFLearn

TFLearn ir modulāra dziļās mašīnmācīšanās bibliotēka, kas būvēta balstoties uz *Tensorflow*. Visas funkcijas ir būvētas balstoties uz tenzoriem un var tikt izmantotas arī atsevišķi no *TFLearn*. Bibliotēka tika veidot ar nolūku piedāvāt augstāka līmeņa *API TensorFlow* vajadzībām, lai atvieglotu un paātrinātu izpētes gaitu, vienlaicīgi veidojot rīku, kas ir pilnībā saderīgs ar *TensorFlow* [30].

3.3.1. *TFLearn* priekšrocības

Viegli lietojams augsta līmeņa *API*, kas paredzēts padziļinātu neironu tīklu veidošanai, kā arī šim API pieejamas vairākas pamācības un pielietojuma piemēri. Papildus ir iespējama ātra prototipēšana, izmantojot augstas modularitātes iebūvētu neironu tīklu slāņus un citas metrikas [30].

Visas funkcijas ir būvētas uz tenzoriem un var tikt izmantotas atsevišķi no TFLearn. Eksistē spēcīgas atbalsta funkcijas, kas palīdz apmācīt *Tensorflow* grafikus, piedāvājot vairākus ievades, izvades un optimizācijas risinājumus, kā arī tiek piedāvāta vienkārša un skaista grafiku vizualizācija, kas attēlo datus [30].

TFLearn piedāvā Iespējamu izmantot vairākus CPU/GPU [30].

3.3.2. *TFLearn* trūkumi

TFLearn var radīt problēmas pie jaunu versiju izmantošanas. Atjauninājumi ir bieži un jaunās versijas nereti izraisa atgriezeniskās saderības (*backwards compatibility*) problēmas [31].

Grūti atrast informāciju par *TFLearn* ārpus izcilās dokumentācijas. Problēmu gadījumā būtu grūti atrast risinājumu kādai bibliotēkas problēmai.

3.4. Keras

Keras ir augsta līmeņa atvērtā koda neironu tīklu bibliotēka, rakstīta programmēšanas valodā *Python*. Tā ir balstīta uz *Tensorflow*, bet ir integrēta arī ar citiem zemā līmeņa mašīnmācīšanās ietvariem. Keras tika radīts, lai piedāvātu lietotājiem ātri un vienkārši veikt dziļās mašīnmācīšanās eksperimentus ar neironu tīkliem. Keras galvenais fokuss ir būt lietotājam draudzīgam, modulāram un izvēršamam rīkam [32].

3.4.1. *Keras* priekšrocības

Keras ļauj ātri un ērti izveidot, un testēt neironu tīklu ar minimālu koda daudzumu. Tāpat ir iespējams arī izveidot pat ļoti sarežģītus neironu tīklus ar minimālu koda daudzumu [33].

Jebkuru funkcionalitāti ir iespējams izveidot kā moduli, kas padara tālāku kombinēšanu ļoti ērtu un vienkāršu. Pamatā *Keras* jau nodrošina plašu klāstu ar pamata komponentēm, kas paredzētas *Keras* ekosistēmas darbināšanai[33]. Keras piedāvā dažas pamata funkcionalitātes funkcijas, kas atvieglo *Keras* modeļu izveidi.

Keras ir pieejama plaša komūna. 2020.gada sākumā tika dokumentēts, ka *Keras* izmanto jau 375 000 lietotāju, kas nodrošina plašu komūnas atbalstu [34]. Kā arī *Keras* nodrošina vienu no pilnvērtīgākajām un saprotamāk organizētajām dokumentācijām, kādas pieejamas līdzvērtīgiem rīkiem [35],[36].

3.4.2. *Keras* trūkumi

Keras ir būvēts kā augsta līmeņa *API* un individuālu parametru nepieciešamības gadījumā, nereti ir nepieciešams individuālus parametrus, kas atšķiras no rīka standartā piedāvātajiem. Lai arī Keras nodrošina iespēju implementēt paša definētus parametrus, tas tomēr piedāvā zemāku dinamiskumu kā citas zema līmeņa bibliotēkas [34]. Izmantojot individuālos parametrus un saskaroties ar zema līmeņa *bibliotēkas API* var rasties problēmas, jo Keras ir veidots kā augsta līmeņa bibliotēka un saskarsme ar zema līmeņa bibliotēkas *API* nav paredzēta [38].

Keras piedāvā plašu biežāk izmantoto funkcionalitāšu klāstu, kas nepieciešams izstrādājot un pētot dziļās mašīnmācīšanās modeļus, bet tā piedāvātais klāsts ir šaurāks kā citiem rīkiem, piemēram, *Tensorflow* [34],[37].

Apskatot aprakstītās mašīnmācīšanās bibliotēkas risinājuma izstrādei nav piemērotas *PyTorch* vai *TFLearn* bibliotēkas, jo par tam ir salīdzinoši ļoti grūti atrast informāciju un nav zināms vai tām eksistē aktīvas komūnas, kur notiktu dalīšanās ar zināšanām. Gan Keras, gan Tensorflow ir stiprās puses, ka Tensorflow ir plašas rediģēšanas iespējas, bet sarežģītības līmenis ir augsts. Keras, savukārt, satur salīdzinoši maz rediģēšanas iespējas, bet satur labi definētus pieejamos algoritmus. Lai salīdzinātu bibliotēkas tika ņemta vērā salīdzināšana internetā [32]. Tika apskatīti salīdzinošie parametri – veiktspēja un draugu rekomendācijas. Pie abiem parametriem Keras tika izvelēts kā labākais rezultāts ar redzamu pārsvaru.

4. DATU APSTRĀDE

Datu apstrādes nodaļā aprakstīts kā pieejamie dati tikai filtrēti un apstrādāti, lai iegūtu mašīnmācīšanā lietojamus datus. Lai būtu iespējams trenēt mašīnmācīšanās modeli, ir nepieciešami dati. Bakalaura darba ietvaros tika iegūti dati no *BKUS* sistēmas *PREM* aptaujām no 2018. līdz 2020. gadam. Dati tika iegūti *.xlsx* datu formātā un kopā tika iegūti 7774 komentāri no pacientiem. Iegūtie dati nesatur personu sensatīvo informāciju, paļaujoties GDPR regulas parasībām. Datu apstrādi sākot, dati tiek ielādēti no *"data/comments.xlsx"* faila no direktorijas "data". Datu apstrādē izmantotās programmatūras pamat-kods ir pieejams failā *"notebooks/data_processor.ipynb"*

4.1. Datu filtrēšana

Vispirms tika iegūtas datos piejamās kolonnas un to nosaukumi tika mainīti uz kodā viegli pierakstāmiem nosaukumiem. Kolonnu nosaukumi redzami tabulā (4.1. tabula).

4.1. tabula

Kolonnas nosaukums ieejas datos	**Kolonnas nosaukums izejas datos**
Kometnārs	*comment*
Datums	*date*
Nodaļa	*section*
Komentāra veids	*type*
Apstāklis	*reason*

Tālāk tika atlasītas tikai nepieciešamās datu kolonnas "*comment*" un "*type*". Kolonnas. Pārējie dati tiek ignorēti, jo nav nepieciešami.

4.2. Datu attīrīšana

Pēc kolonu izvēlēs tika veikta datu attīrīšana no ne-latīņu tekstiem. Datu attīrīšanai tika izveidots filtrs, kas pārbauda vai teksts sastāv tikai no latīņu burtiem. Pārbaudei tikai izmantota *alphabet_detector* bibliotēkas funkcija *is_latin.*

Problēmas nesagādāja dati līdz 4434. rindai. 4434 rinda tika apskatīta un izrādījās, ka tajā komentāra vietā atradās telefona numurs. Funkcija *is_latin* tika papildināta ar ievaddatu piespiedu transformāciju uz teksta mainīgo.

Dati tika filtrēti pēc funkcija *is_latin* un pēc filtrēšanas palika 6376 komentāru rindas.

Tālāk tika apskatīti tipa kolonnā esošie dati un tika apskatītas atļautās kolonnas vērtības. 4.2. tabula attēlo atļautās vērtības

4.2. tabula

Vērtība	**Skaidrojums**
poz	Pozitīvie komentāri
neg	Negatīvie komentāri
poz/neg	Gan pozitīvie, gan negatīvie komentari

Tālāk tika atrasts visas esošās kolonas vērtības, kas neatbilst atļautajām un tās tika transformētas uz atļautajām vērtībām, ko attēlo 4.3. tabula.

4.3. tabula

Vērtība	**Vērtības transformācija**
prz	poz
Piz	poz
Plz	poz
Paz	poz

poz/	poz
Ptz	poz

Pēc komentāru tipa kolonnas attīrīšanas tika pārbaudīts vai komentāru tipa kolonna satur tikai atļautās vērtības. Pēc pārliecināšanās vērtības tika tālāk transformētas uz skaitliskām vērtībām, ko attēlo 4.4. tabula

4.4. tabula

Vērtība	**Vērtības transformācija**
poz	1
neg	0
poz/neg	2

Apstrādātie dati tika saglabāti failā *"data/comments_processed.csv*".

4.3. Datu standartizēšana

Mašīnmācīšanā ir nepieciešami standartizēti dati, lai datus būtu iespējams izmantot mašīnmācīšanā, visiem komentāriem jābūt vienāda garuma un kā skaitļiem, nevis vārdiem. Kā arī nepieciešams transformēt komentāra tipu uz klasifikatora tipu, nevis skaitli. Datu standartizācijai tika izmantota *Keras* bibliotēkas klase *Tokenizer*, kas vārdus pārvērš no teksta uz skaitļiem. *Tokenizer* klases pielietošanu ilustrē 4.1. attēls.

```
# Embed data into padded integer sequences
# Get Comment Values
comments = df.comment.values
# Create a Words to Integers array with size of
tokenizer = keras.preprocessing.text.Tokenizer(num_words=vocab_max_size)
# Get vocabulary for current data set
tokenizer.fit_on_texts(comments)
# Set the actual size of vocabulary
vocabulary_size = len(tokenizer.word_index) + 1
# Encode comments from words to integers with the defined vocabulary. Can be later decoded with .sequences_to_texts
encoded_comments = tokenizer.texts_to_sequences(comments)
# Pad all comments to the size of the sentence. Add 0 if the cells are empty. Cut too long sentences
padded_encoded_comments = keras.preprocessing.sequence.pad_sequences(encoded_comments, maxlen=sentence_size)
```

4.1. attēls

Vispirms tika izveidots "*tokenizer"* objekts, kurš satur vārdnīcu no vārdiem uz skaitļiem. Visi komentāru vārdi tiek ievietoti vārdnīcā un sagrupēti pēc to lietošanas biežuma. Biežākiem vārdiem identifikators ir mazāks. Pēc tam visi komentāri tiek pārvērsti no teksta uz skaitļu virknēm.

Pēc komentāru pārkodēšanas uz skaitļu virknēm bija nepieciešams visus komentārus iegūt kā vienāda lieluma skaitļu virknes. Tas tika īstenots ar *Keras* bibliotēkas "*Sequence"* klases *pad_sequences* funkciju, kas visus komentārus pagarināja vai saīsināja līdz 512 vārda garumam, kas tika izvēlēts. Konkrētais vārda garums tika izvelēts, jo maksimālais teksta garums komentāros ir 563 un tika izvēlēta augstākā divnieka pakāpe maksimālajam vārda garumam, kas nepārsniedz maksimālo teksta garumu komentāros.

Pēc komentāru standartizēšanas bija nepieciešams standartizēt ar komentāru tipu uz *one-hot* šifrējumu. "*One-hot"* šifrēšana ir skaitļu pārveidošana skaitļu virknē, kur katram skaitlim atbilst savs identifikators, ko ilustrē 4.2. attēls.

```
# Categorize answers.
answers = keras.utils.to_categorical(df.type.values,num_classes=3)
```

4.2. attēls

Lai dati būtu standartizēti, bija nepieciešams kategorizēt komentāra tipu . Konkrētajā gadījumā tipiem ir 3 kategorijas: 1, 0 vai 2. Katrs komentāra tips tika pārveidots uz *One-hot* šifrējumu. Komentāru tipa šifrējumu attēlo 4.5. tabula.

4.5. tabula

Tipa vērtība	**Tipa *One-hot* šifrējums**
1	[0,1,0]
0	[1,0,0]
2	[0,0,1]

5. MODEĻA PARAMETRI

Mašīnanmācīšanās modelim var būt vairāki parametri, ko mainot, izmainās māšīnmācīšanās modeļa potenciālo rezultātu iznākums

5.1. Laikmets (epoch)

Laikmets ir definējama mērvienību, kur tiek vienu reizi pāriets pāri visai datu kopai. Mainot laikmeta skaitu, tiek noteikts, cik reizes datiem tiks pāriets pāri.

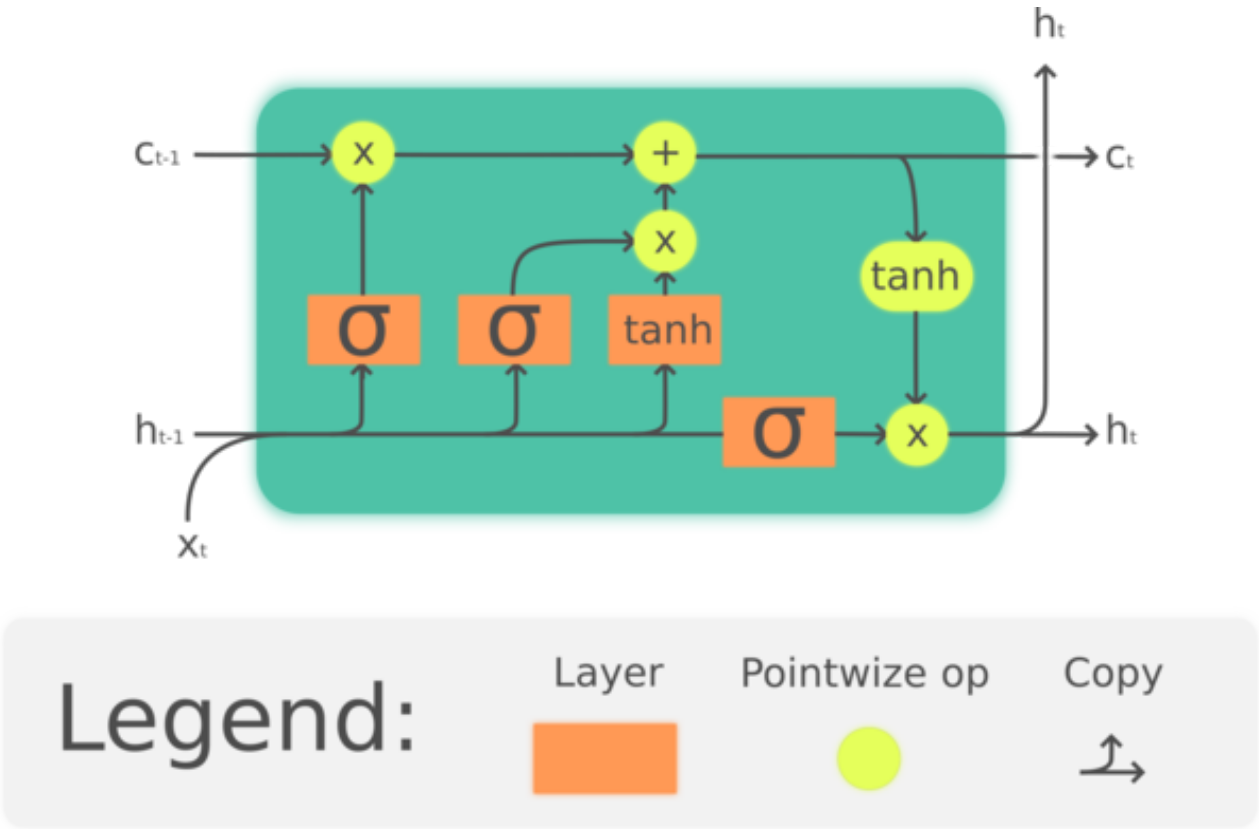
5.2. Slāņi

Slānis ir modeļa mācīšanās algoritms. Slāņus ir iespējams definēts katru atsevišķi un dažādās kombinācijās.

5.2.1. Embeding

Embeding slānis ir modeļa mācīšanās slānis, ko pielieto dabiskās valodas apstrādē, kur vārdi vai frāzes no vārdnīcas, tiek savienoti ar vektoriem vai reāliem skaitļiem. *Embeding* slānis uzlabo dabiskās valodas apstrādes algoritmu precizitāti, ja to izmanto kā pirmo slāni priekš datu ievades.[42]

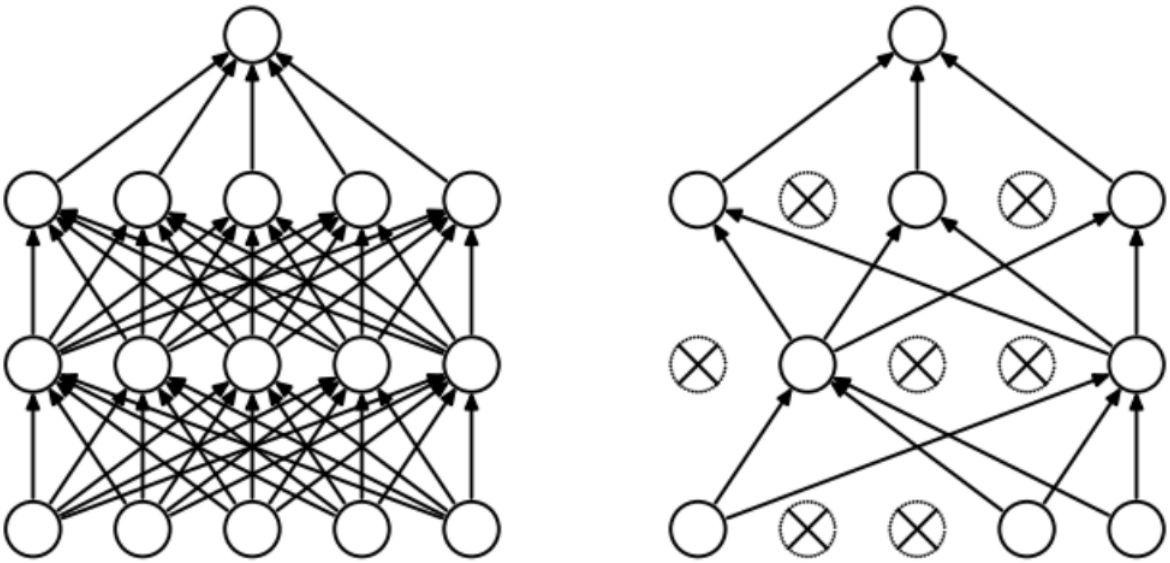
5.2.2. LSTM



5.1. attēls

LSTM slānis ir modeļa mācīšanās slānis, ko pielieto nenoteikta lieluma teksta apmācības gadījumos. *LSTM* slānis atšķiras ar atgriezeniskās saites izmantošanu tā aprēķinos. LSTM sastāv no vairākām komponentēm: šūnas, ievad vārtiem, izvadīšanas vārtiem un aizmiršanas vārtiem. Skatīt 5.1. attēlu. Katri vārti atbild par savu funkcionalitāti, kas tiek dinamiski izmantoti. LSTM slāņus izmanto klasifikācijas un apstrādes gadījumos [43].

5.2.3. Dropout



5.2. attēls

Atmesto datu kopas *(Dropout)* slānis ir modeļa mācīšanās slānis, kas nejauši izvēlas neironus, kuru vērtības uzlikt uz 0. Katrā mašinmācīšanās modeļa izpildes reizē no tīkla tiek izslēgti n neironi. Neironu izslēgšana novērš iespēju mašīnmācīšanās modeļa pārmācīšanos. 5.2. attēlā labi ilustrēta atmesto datu kopu (*Dropout)* slāņu darbiba[44].

5.2.4. Dense

Blīvas datu kopas (Dense) slānis ir parast dziļās mašīnmācīšanās neironu tīkla slānis. Tas ir visbiežāk lietotais neironu tīklu slānis. Tas darbojas pēc algoritma

```
output = activation(dot(input, kernel) + bias)
```

Kur ievaddati ir iepriekšējā slāņa dati. *Kernel* ir iepriekšējie dati no produkta. *Dot* ir skalārais reizinājums no *input* un *kernel. Bias* ir vērtība priekš modeļa optimizācijas. *Activation* ir funkcija, kas tiek izmantota izvaddatu aprēķināšanai [45].

5.2.5. Spatial_dropout1d

Spatial_dropout1d slānis tiek izmantots kā uzlabota laternatīva Atmesto datu kopas (Dropout) slānim. Tas izmet ne tikai idividuālas vērrtības [41].

5.3. Aktivizācijas funkcijas

Mašīnmācīšanās modeļos aktivizācias funkcijas tiek lietotas, lai normalizētu izejošos datus vai tos pielāgotu attiecīgi sagaidāmajām izejvērtībām.

5.3.1. Softmax

Softmax aktivizācijas funkcija saskaita visus ievaddatus un izejdatos izdod kopējo summu viens. Katra ievaddatu vērtība tiek procentuāli pielāgota uz kopējā fona.

5.3.2. Sigmoid

Sigmoid aktivizācijas funkcija nosaka lielāko vērtību no ievaddatiem un izdod izejdatus attiecībā pret to. Katrai ievadvērtībai tiek izdota izvadvērtību diapazonā no 0 līdz 1.

5.3.3. Relu

Relu aktivizācijas funkcija Ir līdzīga funkcija sigmoid, bet tā atšķiras ar to, ka algoritms nav dinamisks, bet gan statiski ass un mainās pie noteiktām vērtībām. Tas ir 0 pirms 0 un palielinās attiecīgi pēc 0.

6. MODEĻA IZSTRĀDE

Sākumā tika izstrādāts modelis ar pamata konfigurāciju [20]. Modelis tika pārveidots tāpēc, ka ir 3 iespējamās vērtības, pozitīvs, negatīvs, neitrāls (pozitīvs un negatīvs) un ar "binary_crossentropy" izdod tikai 2 atbildes, tika izmainīts atbilžu ievaddatu formāts uz "*one-hot encoding*", kur [1,0,0] atbilda negatīvam, [0,1,0] atbilda pozitīvam un [0,0,1] atbilda neitrālam. Modelī tika pārveidots tikai pēdējais slānis no *Dense*(1, activation='sigmoid') uz *Dense*(3, activation='softmax'). Vispirms tika izmainīts neironu skaits, jo tagad atbilde sastāv no masīva (*array*) garumā 3. Tad tika nomainīta aktivizācijas funkcija no "*sigmoid*" uz "*softmax*", jo "sigmoid" visas vērtības pārveidoja no 0 līdz 1, bet "*softmax*" visu slānī esošo vērtību skaitu pārveidoja par summu 1 kopumā.

Trenēšanai tika izvēlēts "*epoch*" skaits 25, jo pie tā vēl aktīvi mainījās trenētais modelis un modeļa precizitāte. Tas ir novērojams arī datos, kas atrodami pielikumā (skatīt pielikuma 1. attēlu).

Lai saglabātu modeļa svarus kļūdu (error) gadījumā vai lai izvērtētu kā mainās reālā precizitāte (accuracy) uz neredzētiem datiem, tika pievienota atgriezeniskā saite (callback). Kur katram modeļa veidam tika pievienots atskaites punkts (checkpoint) uz katru treniņa "epoch". Kā arī tikai pievienots "best_model", kur tika saglabāti modeļa svari ar labāko precizitāti (accuracy)

Pēc modeļa pielāgošanas tas tika trenēts ar standartizētajiem datiem. Attēlā redzama izveidotā modeļa konfigurācija. Pēc modeļa trenēšanas beigām, tas tika testēts ar neredzētiem datiem un tika sasniegta tikai **80.47%** precizitāte (skatīt pielikumu 1. attēlu), kamēr vēlamā precizitāte ir virs 95%.

```
# create a model
def getModel(name):
    # get a model class
    model = keras.models.Sequential(name=name)

    # models layers
    model.add(layers.Embedding(vocabulary_size, vocab_vector_size, input_length=sentence_size, name='embed'))
    model.add(layers.SpatialDropout1D(0.25, name='spatial_dropout'))
    model.add(layers.LSTM(64, dropout=0.5, recurrent_dropout=0.5, name='lstm_1'))
    model.add(layers.Dropout(0.5))

    # Last output layer
    model.add(layers.Dense(3, activation='softmax', name='dense_output'))

    # compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
    return model
```

6.1. attēls

Par iespējamo problēmu tikai izvirzīts tas, ka dotajā piemērā [20] izmantotais datu kopa bija angļu valodā un maksimālais vārdu garums bija 200. Pieejamajos datos figurēja gan gari , gan īsi komentāri un maksimālais teksta garums sasniedza 563 vārdus. Tāpēc, lai atbilstu pārsvaram vārdu ievad-datu skaits tika palielināts uz 512. Mazā datu apjoma dēļ atmetamo datu kopu (*dropout*) nebija iespējams uzlikt lielu, jo pēc rekomendācijām lieliem datu apjomiem 100k un uz augšu atmetamo datu kopu (*dropout*) ir rekomendēts uzstādīt uz 0.5 , bet maziem apjomiem rekomendētais apjoms ir 0.2. Dropout arī tika uzstādīts rekomendētais apjoms. Pēc izmaiņu veikšanas modelis tika trenēts un testēts. Pēc izmaiņām labākais iegūtais rezultāts bija **79.47%** precizitāte (skatīt pielikuma 2. attēlu). Izmaiņas negatīvi ietekmēja modeļa izveide.

79.47% precizitāte nav vēlamais gala rezultāts. Tika meklēti risinājumi, kā uzlabot *LSTM* modeļa precizitāti. Kā viena no iespējām tika atrasta vēlviena LSTM slāņa pievienošana jau esošajam. Gala rezultātā būtu 2 LSTM slāni. Tas tika izdarīts un tika iegūts modelis:

```
# create a model
def getModel(model_name):
    # get a model class
    model = keras.models.Sequential(name=model_name)

    # models layers
    model.add(layers.Embedding(vocabulary_size, vocab_vector_size, input_length=sentence_size, name='embed'))
    model.add(layers.SpatialDropout1D(0.25, name='spatial_dropout'))
    model.add(layers.LSTM(66, dropout=0.5, recurrent_dropout=0.5, name='lstm_1', return_sequences=True))
    model.add(layers.Dropout(0.5,name="dropout_1"))
    model.add(layers.Dense(512, activation="sigmoid", name="dense_1"))
    model.add(layers.LSTM(33, dropout=0.5, recurrent_dropout=0.5, name='lstm_2'))
    model.add(layers.Dropout(0.5,name="dropout_2"))
    model.add(layers.Dense(33, activation="sigmoid", name="dense_2"))
    # Last output layer
    model.add(layers.Dense(3, activation='softmax', name='dense_output'))

    # compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
    return model
```

6.2. attēls

Starp *LSTM* slāņiem tika ievietoti arī *Blīvo datu (Dense) slāņi, lai dati tiktu arī sapludināti, ne tikai papildināti. Modeļis tika trenēts un tika iegūts rezultāt* ***80.35%*** *precizitāte, kas joprojām nav apmierinošs rezultāts.*

REZULTĀTI

Pēc mašīnmācīšanās modeļa trenēšanas pie dažādām vidēm augstākā iegūtā precizitāte ir 80.47%. Baklaura darba sākumā izvirzītā hipotēze nepiepildījās, jo neizdevās darba gaitā izveidot mašīnmācīšanās modeli ar preciztiāti virs 95%.

Rezultātus var uzskatīt par neizdevušiem, ja precizitāti salīdzina ar angļu valodas mašīnmācīsanās eksistējošiem modeļiem, bet salīdzinot ar citu valodu masīnmācīšanās eksistējošiem modeļiem precizitāte ir augsta, pat pārsniedzot Turku valodā izstrādātā modeļa esošo precizitāti.

SECINĀJUMI

Pēc dažādu mašīnmācīšanās modeļu izveides un apmācības var secināt, ka veidojot līdzīgus modeļus precizitātei būtu jābūt ap 80%. Tas nozīmē, ka problēma varētu būt datu trūkums, jo līdzīga precizitāte norāda uz kāda parametra neatbilstību un mainot mašīnmācīšanās modeļa parametrus var secināt, ka potenciāli 20% datu varētu būt nestandarta, ko ir grūti klasificēt. Datu apjoms sagādāja grūtības, jo augstas kvalitātes mašīnmācīšanās modeļiem nepieciešamais datu apjoms ir mērāms desmitos tūkstošu vai pat miljonos, bet pētījuma gaitā izdevās iegūt tikai ap 7700 sākotnējo datu, kuru skaits samazinājās, kad tika izfiltrēta tikai latviešu valoda.

Autors ir nonācis pie secinājuma, ka ar esošajiem datiem nav iespējams izveidot mašīnmācīšanās modeli, kas spētu noteikt komentāra nolūku ar 95 vai vairāk procentu precizitāti. Mašīnmācīšanās algoritmi spēj pamatā atpazīt komentāru nolūkus, bet latviešu valodā pastāv šķēršļi, kas to efektivitāti samazina.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. E. Alpaydin, *Introduction to Machine Learning, fourth edition*, Massachusetts Institute of Technology, 2020, pp. 1-15.

2. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, 1992, nodaļa: 11.2 Samuel's Checkers Player.

3. A Brief History of Machine Learning (skatīts 23.05.2020.) Pieejams: https://www.dataversity.net/a-brief-history-of-machine-learning/

4. A 'Brief' History of Neural Nets and Deep Learning, Part 4 (skatīts 23.05.2020.) Pieejams: https://medium.com/@andreykurenkov/a-brief-history-of-neural-nets-and-deep-learning-part-4-61be90639182

5. G.E. Hinton, S. Osindero, Y.W. Teh, *A fast learning algotythm for deep belief nets*, Department of Computer Science, University of Toronto, 2006.

6. R. Raina, A. Madhavan, A.Y. Ng, *Large-scale Deep Unsupervised Learning using Graphics Processors*, Computer Science Department, Stanford University, 2009.

7. Eiropas Parlamenta un Padomes Regula (ES) 2016/679 (2016. gada 27. aprīlis) par fizisku personu aizsardzību attiecībā uz personas datu apstrādi un šādu datu brīvu apriti un ar ko atceļ Direktīvu 95/46/EK (Vispārīgā datu aizsardzības regula) (Dokuments attiecas uz EEZ) (skatīts 23.05.2020.) Pieejams: https://eur-lex.europa.eu/eli/reg/2016/679/oj?locale=lv

8. Types of Machine Learning Algorithms You Should Know (skatīts 23.05.2020.) Pieejams: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

9. Mašīnmācīšanās (skatīts 23.05.2020.) Pieejams: https://lv.wikipedia.org/wiki/Ma%C5%A1%C4%ABnm%C4%81c%C4%AB%C5%A1an%C4%81s

10. E. Yıldırım, F.S. Çetin, G. Eryiğit, T.Temel, *The Impact of NLP on Turkish Sentiment Analysis*, 2015, Istambul Technical University, Turkcell Global Bilgi, 2015.

11. 10 Popular datasets for sentiment analysis (skatīts 23.05.2020.) Pieejams: https://analyticsindiamag.com/10-popular-datasets-for-sentiment-analysis/

12. Mašīnmācīšanās testa dati, Stanford University (skatīts 23.05.2020.) Pieejams: http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip

13. A. Go, R. Bhayani, L.Huang, *Twitter Sentiment Classification using Distant Supervision,* Stanford University, 2009.

14. Sentiment analysis algorithms: evaluation performance of the Arabic and English language (skatīts 23.05.2020.) Pieejams:

https://www.researchgate.net/publication/328788884_Sentiment_analysis_algorithms_evaluation_performance_of_the_Arabic_and_English_language

15. A Neural Implementation of NBSVM in Keras (skatīts 23.05.2020.) Pieejams: https://github.com/amaiya/keras-nbsvm/blob/master/keras-nbsvm.ipynb

16. S. Wang, C.D. Manning, *Baselines and Bigrams: Simple, Good Sentiment and Topic Classification,* Department of Computer Science, Stanford University, 2012.

17. Understanding Support Vector Machine(SVM) algorithm from examples (along with code) (skatīts 23.05.2020.) Pieejams: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

18. Naive Bayes classifier (skatīts 23.05.2020.) Pieejams: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

19. Decision tree learning (skatīts 23.05.2020.) Pieejams: https://en.wikipedia.org/wiki/Decision_tree_learning

20. Deep Learning LSTM for Sentiment Analysis in Tensorflow with Keras API. (skatīts 23.05.2020.) Pieejams: https://medium.com/datadriveninvestor/deep-learning-lstm-for-sentiment-analysis-in-tensorflow-with-keras-api-92e62cde7626

21. C. Baziotis, N. Pelekis, C. Doulkeridis, *DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis*, Data Science Lab, University of Piraeus, Piraeus, Greece, 2017.

22. Top five use cases of TensorFlow (skatīts 23.05.2020.) Pieejams: https://www.exastax.com/deep-learning/top-five-use-cases-of-tensorflow/

23. TensorFlow oficiālā mājaslapa (skatīts 23.05.2020.) Pieejams: https://www.tensorflow.org/

24. TensorFlow Pros and Cons – The Bright and the Dark Sides (skatīts 23.05.2020.) Pieejams: https://data-flair.training/blogs/tensorflow-pros-and-cons/

25. Pros And Cons Of Using TensorFlow In The Production Environment (skatīts 23.05.2020.) Pieejams: https://neurosys.com/article/pros-and-cons-of-using-tensorflow-in-the-production-environment/

26. PyTorch (skatīts 23.05.2020.) Pieejams: https://en.wikipedia.org/wiki/PyTorch

27. What is PyTorch used for? (skatīts 23.05.2020.) Pieejams: https://www.quora.com/What-is-PyTorch-used-for

28. Comparison of AI Frameworks (skatīts 23.05.2020.) Pieejams: https://pathmind.com/wiki/comparison-frameworks-dl4j-tensorflow-pytorch

29. Pytorch getting slower with every iteration (skatīts 23.05.2020.) Pieejams: https://discuss.pytorch.org/t/pytorch-getting-slower-with-every-iteration/9089

30. TFLearn: Deep learning library featuring a higher-level API for TensorFlow. (skatīts 23.05.2020.) Pieejams: http://tflearn.org/

31. TFLearn Reviews & Product Details (skatīts 23.05.2020.) Pieejams: https://www.g2.com/products/tflearn/reviews#survey-response-1835091

32. Comparison Dashboard: TFLearn vs Keras vs TensorFlow (skatīts 23.05.2020.) Pieejams: https://www.predictiveanalyticstoday.com/compare/tflearn-vs-keras-vs-tensorflow/

33. Keras (skatīts 23.05.2020.) Pieejams: https://en.wikipedia.org/wiki/Keras

34. TensorFlow or Keras? Which one should I learn? (skatīts 23.05.2020.) Pieejams: https://medium.com/implodinggradients/tensorflow-or-keras-which-one-should-i-learn-5dd7fa3f9ca0

35. Why choose Keras? (skatīts 23.05.2020.) Pieejams: https://keras.io/why_keras/

36. Python Keras Advantages and Limitations (skatīts 23.05.2020.) Pieejams: https://data-flair.training/blogs/python-keras-advantages-and-limitations/

37. What are the drawbacks of Keras? (skatīts 23.05.2020.) Pieejams: https://www.quora.com/What-are-the-drawbacks-of-Keras

38. Python Keras Advantages and Limitations (skatīts 23.05.2020.) Pieejams: https://data-flair.training/blogs/python-keras-advantages-and-limitations/

39. What is a Tensor? (skatīts 23.05.2020.) Pieejams: https://deepai.org/machine-learning-glossary-and-terms/tensor

40. Vector (skatīts 23.05.2020.) Pieejams: https://whatis.techtarget.com/definition/vector

41. tf.keras.layers.SpatialDropout1D (skatīts 23.05.2020.) Pieejams: https://www.tensorflow.org/api_docs/python/tf/keras/layers/SpatialDropout1D

42. How to Use Word Embedding Layers for Deep Learning with Keras (skatīts 23.05.2020.) Pieejams: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

43. Long short-term memory (skatīts 23.05.2020.) Pieejams: https://en.wikipedia.org/wiki/Long_short-term_memory

44. Dropout Neural Network Layer In Keras Explained (skatīts 23.05.2020.) Pieejams: https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab

45. Dense layers explained in a simple way (skatīts 23.05.2020.) Pieejams: https://medium.com/datathings/dense-layers-explained-in-a-simple-way-62fe1db0ed75

PIELIKUMI

1. Lstm piemēra modelis – 80.47% precizitāte

	epoch	loss	categorical_accuracy
0	1.0	0.9117637872695920	0.6076585054397580
1	2.0	0.6699029207229610	0.724419355392456
2	3.0	0.589869499206543	0.7884494662284850
3	4.0	0.5521935820579530	0.7972379326820370
4	5.0	0.5765159130096440	0.7978656888008120
5	6.0	0.5886428952217100	0.7953546643257140
6	7.0	0.5852617621421810	0.8047708868980410
7	8.0	0.6449821591377260	0.8003766536712650
8	9.0	0.6672537326812740	0.7890772223472600
9	10.0	0.7751988768577580	0.7853107452392580
10	11.0	0.7059122920036320	0.8022598624229430
11	12.0	0.8039196729660030	0.8003766536712650
12	13.0	0.9235427379608150	0.784055233001709
13	14.0	0.8415976166725160	0.7746390700340270
14	15.0	0.8985702395439150	0.790960431098938
15	16.0	0.8921834230422970	0.7784055471420290
16	17.0	0.9625155925750730	0.7922159433364870
17	18.0	0.9000115394592290	0.7972379326820370
18	19.0	0.9239002466201780	0.7903327345848080
19	20.0	0.946404218673706	0.7859385013580320
20	21.0	0.9670292735099790	0.7765222787857060
21	22.0	0.9744862914085390	0.7796609997749330
22	23.0	0.8853360414505010	0.7639673352241520
23	24.0	1.089463472366330	0.7884494662284850
24	25.0	0.9356583952903750	0.7802887558937070
25	26.0	0.9697622656822210	0.7834274768829350
26	27.0	1.0222175121307400	0.7859385013580320
27	28.0	1.0284159183502200	0.77715003490448
28	29.0	1.136217713356020	0.7859385013580320
29	30.0	1.161490797996520	0.7809165120124820
	7		0.8047708868980410

2. Lstm lielākas vārdnīcas modelis – 79.47% precizitāte

	epoch	loss	categorical_accuracy
0	1.0	0.8605661988258360	0.6020087599754330
1	2.0	0.7753257155418400	0.6603891849517820
2	3.0	0.7312855124473570	0.7043314576148990
3	4.0	0.5883323550224300	0.7708725929260250
4	5.0	0.5740466117858890	0.771500289440155
5	6.0	0.6688149571418760	0.7602008581161500
6	7.0	0.6186894774436950	0.7802887558937070
7	8.0	0.5922308564186100	0.7947269082069400
8	9.0	0.6448741555213930	0.7934714555740360
9	10.0	0.6772956252098080	0.7758945226669310
10	11.0	0.6756924390792850	0.7859385013580320
11	12.0	0.7500459551811220	0.764595091342926
12	13.0	0.7041975855827330	0.7696170806884770
13	14.0	0.7579900622367860	0.770244836807251
14	15.0	0.8245254158973690	0.7595731616020200
15	16.0	0.8647134304046630	0.7558066248893740
16	17.0	0.8898164629936220	0.7570621371269230
17	18.0	0.9920495748519900	0.743879497051239
18	19.0	0.8664348721504210	0.7608286142349240
19	20.0	0.8768506050109860	0.750784695148468
20	21.0	0.9563798308372500	0.7827997207641600
21	22.0	0.9573386311531070	0.7746390700340270
22	23.0	5.670666694641110	0.6051475405693050
23	24.0	0.8822903633117680	0.7294412851333620
24	25.0	0.894808292388916	0.7313245534896850
	7		0.7947269082069400

3. LSTM - 2 LSTM slāņi modelis – 80.35

	epoch	loss	categorical_accuracy
0	1.0	1.0049078464508100	0.6020087599754330
1	2.0	0.9507278203964230	0.6020087599754330
2	3.0	0.9166691303253170	0.6020087599754330
3	4.0	0.8044782876968380	0.6020087599754330
4	5.0	0.685107946395874	0.7175140976905820
5	6.0	0.643538773059845	0.7558066248893740
6	7.0	0.6139442920684810	0.7821720242500310
7	8.0	0.7310555577278140	0.7338355183601380
8	9.0	0.6022740602493290	0.7796609997749330
9	10.0	0.6134157180786130	0.7871939539909360
10	11.0	0.6203808784484860	0.790960431098938
11	12.0	0.6223596930503850	0.7991211414337160
12	13.0	0.6719269752502440	0.7790332436561580
13	14.0	0.66825932264328	0.8035153746604920
14	15.0	0.6468348503112790	0.790960431098938
15	16.0	0.6923304796218870	0.7953546643257140
16	17.0	0.6796601414680480	0.7991211414337160
17	18.0	0.6742569208145140	0.7777777910232540
18	19.0	0.6885607242584230	0.7401130199432370
19	20.0	0.7071745991706850	0.7627118825912480
20	21.0	0.7292494773864750	0.7376019954681400
21	22.0	0.7461636066436770	0.7219083309173580
22	23.0	0.8031168580055240	0.7099811434745790
23	24.0	0.8241438865661620	0.6704331636428830
24	25.0	0.8606789708137510	0.6691776514053350
	13		0.8035153746604920

Pielikums

Dokumentārās lapas forma

Bakalaura darbs „***PACIENTU KOMENTĀRU NOLŪKA NOTEIKŠANA, IZMANTOJOT MAŠĪNMĀCĪŠANOS***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Valdis, Aglonietis*** _____ **25.05.2020**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: ***Dr.med., Dzintars, Mozgis*** _____ **25.05.2020.**

Recenzents: ***Normunds Grūzītis***

Darbs iesniegts **25.05.2020**.

Bakalauru darbu pārbaudījumu komisijas sekretārs: _____

Darbs aizstāvēts bakalauru darbu pārbaudījuma komisijas sēdē

___.06.2020. prot. Nr. _____

Komisijas sekretārs(-e): _____