LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

SABIEDRĪBAS ATTIEKSMES MODELĒŠANA, IZMANTOJOT SENTIMENTA ANALĪZI

BAKALAURA DARBS

Autors: **Dāvis Nicmanis**
Studenta apliecības Nr.: dn13008
Darba vadītājs: Mg. vad. zin. Pēteris Paikens

RĪGA 2017

ANOTĀCIJA

Šī darba mērķis ir izveidot sentimenta analīzes risinājumu, kuru paredzēts izmantot informācijas ieguves sistēmas koncepta izstrādē. Sentimenta analīze tiks veikta sociālo tīklu ziņām.

Darba izstrādes sākumā tika veikta esošo sentimenta analīzes risinājumu izpēte un to rezultātu salīdzināšana.

Tālāk tika veikta publiski pieejamo treniņdatu korpusu ievākšana. Papildus iegūtajiem datiem, tika izveidots latviešu valodai paredzēts sentimenta analīzes treniņdatu korpuss. Korpusa izveidošanas procesā tika veikta informācijas ieguves sistēmas koncepta izveide.

Pēc nepieciešamo treniņdatu savākšanas, tika veikta ilgās īstermiņa atmiņas rekurentā neirona tīkla izveidošana un optimizēšana.

Darba rezultātā tika iegūts latviešu valodas Twitter ziņu sentimenta korpuss un uz mākslīgajiem neironu tīkliem bāzēts sentimenta analīzes risinājums, kas sasniedza 82,37% precizitāti, 81,86% pārklājumu un 81,86% F_1 mēru.

Atslēgvārdi: sentimenta analīze, mākslīgie neironu tīkli

ABSTRACT

Public opinion modeling using sentiment analysis

The objective of this paper is to create a solution for sentiment analysis which will be used in the proof of concept development for information extraction system. Sentiment analysis will be performed on social network posts.

At the beginning of the research author compared existing methods and their results.

Subsequently, publicly available training data were gathered. In addition to that, author also created training data corpus for Latvian sentiment analysis. In the process of the corpus development, the proof of concept for information extraction system was created.

Finally, a long short-term memory recurrent neural network was created and optimized.

Research results consists of a new corpus of hand classified Latvian language Twitter posts and a neural network based sentiment analysis solution, which achieved 82,37% accuracy, 81,86% recall and 81,86% F_1 score.

Keywords: sentiment analysis, artificial neural networks

SATURS

APZĪMĒJUMU SARAKSTS

NLP (*natural language processing*) – dabīgās valodas apstrāde

RNN (*recurrent neural network*) – rekurents neironu tīkls

LSTM (*long short-term memory*) – ilgā īstermiņa atmiņa

Tvīts – sociālā tīkla Twitter ziņa

API (*application programming interface*) – lietojumprogrammas saskarne

REST (*representational state transfer*) – API arhitektūras veids

Sentiments – emocionālā nokrāsa

Implicīts (*implicit*) – netiešs

Eksplicīts (*explicit*) – tiešs

Korpus (*corpus*) – datu kopa, tekstu kopums

Word2vec – mākslīgo neironu tīklu modeļu kopa, kas paredzēta vārdu pārveidošanai vektrou fromātā

OAuth – atvērts drošas autorizācijas protokols

Marķieris – objekts, kas satur pierakstīšanās informāciju

Base64 – šifrēšanas standarts

Patērētāja atslēga un noslēpums – Twitter autorizācijas dati

HTTP (*hypertext transfer protocol*) – hiperteksta pārsūtīšanas protokols

HTTPS – HTTP protokols, kas nodrošina drošu saziņu

SSL (*secure sockets layer*) – stnadarta drošības tehnoloģija šifrēta savienojuma izveidei starp klientu un serveri

Retvīts (*retweet*) – Pārsūtīta Twitter sociālā tīkla ziņa

N-gramma – virkne ar n locekļiem

JSON (*JavaScript object notation*) – JavaScipt objektu notācija, datu faila formāts

CSV (*comma-seperated values*) – Ar komatiem atdalītas vērtības, datu faila formāts

IEVADS

Viedokļiem un attieksmei cilvēku dzīvē ir ļoti nozīmīga loma. Tie ir vieni no būtiskākajiem faktoriem, kas ietekmē cilvēku nostāju un lēmumu pieņemšanu, kā arī tie kalpo kā vieni no pamatelementiem cilvēku izpratnes un realitātes uztveršanas veidošanā.

Ikdienā cilvēki ietekmējas no apkārtējo cilvēku viedokļa ļoti dažādās situācijās – pildot darba pienākumus, iegādājoties dažādas preces vai pakalpojumus, veicot lēmumus, piemēram, izvēloties vietu, kur doties atvaļinājumā, pieņemot darbā jaunus darbiniekus vai izvēloties par ko balsot vēlēšanās. Attiecīgi cilvēku viedokļi būtiski ietekmē tādas dzīves sfēras kā, piemēram, uzņēmējdarbību vai politiku.

Taču viedokļiem pastāv arī plašāka mēroga ietekme, piemēram, kas vienas kultūras sabiedrībā šķiet pilnīgi pieņemami, citā kultūrā var radīt izbrīnu vai nepatiku. Tas pats attiecas arī uz cilvēku attieksmi pret citu tautību un rašu pārstāvjiem. Bieži vien nākas saskarties ar situācijām, kur tiek ietekmēta cilvēku attieksme un viedoklis attiecībā pret kādu konkrētu cilvēku grupu, balstoties tikai uz atsevišķu indivīdu uzskatiem.

Pateicoties straujajai tehnoloģiju attīstībai un publiski pieejamās informācijas apjoma pieaugumam, šī problēma paliek tikai aktuālāka. Ikviens ikdienā tiek pakļauts lielam informācijas apjomam, kas satur subjektīvu informāciju, tā rezultātā potenciāli ietekmējot lasītāja uztveri un attieksmi. Kā piemēru šim apgalvojumam var minēt informatīvo karu jeb propagandu, pseidoziņu ziņu portālus, kas pasniedz nepatiesas vai pa pusei patiesas ziņas sensacionālā manierē, lai iegūtu peļņu, naida kurināšanu un jeb kādu citu rīcību, kuras rezultātā informācija tiek pasniegta tā, lai ietekmētu lasītāja interpretāciju atbilstoši autora vēlmēm.

Kā dažus no dezinformācijas pamatcēloņiem var minēt apšaubāmus informācijas avotus, faktu trūkumu un centralizētu informācijas iegūšanu. Pārbaudot informācijas avotus un faktus, var cīnīties ar nepatiesu vai sagrozītu ziņu izplatīšanu, taču ne vienmēr tas līdz, un ne vienmēr tas atrisina visas problēmas. Piemēram, ja raksts ir patiess, un to ir sarakstījis uzticams autors, tas nenozīmē, ka tajā pasniegtā informācija nav pasniegta tā, lai lasītājam rastos noteikts priekštats par aprakstīto tēmu, un ne vienmēr autors to ir darījis apzināt. Lasot viena vai dažu autoru rakstītu informāciju, lasītājs ir neizbēgami pakļauts tikai konkrēto autoru viedokļiem. Tas it īpaši ir attiecināms uz ziņām un citiem informācijas avotiem, kuru saturā ir apgalvojumi, kas ir saistīti ar lielākas cilvēku grupas vai sabiedrības attieksmi attiecībā pret kādu tematu.

Šī bakalaura darba mērķis ir izveidot sentimenta analīzes risinājumu, kuru tālāk paredzēts izmantot publiski pieejamas sistēmas izstrādē, ar kuras palīdzību būs iespējams iegūt informāciju no plaša sabiedrības loka par sistēmas lietotājam interesējošiem tematiem, tādā veidā cīnoties pret neobjektīviem informācijas avotiem. Par informācijas avotu tiks izmantots

sociālais tīkls Twitter, kurā ik mēnesi aptuveni 313 miljoni aktīvu lietotāju ievieto maksimums 140 simbolu garas sīkziņas jeb tvītus, kuri tiks iegūti ar Twitter REST API palīdzību. [1] Pēc informācijas iegūšanas, tai tiks veikta sentimenta analīze ar garo īstermiņa atmiņas (LSTM) rekurento neironu tīklu (RNN) palīdzību, lai noteiktu katras ziņas emocionālo nozīmi, kuru apkopojot varēs noteikt kopējo sabiedrības attieksmi attiecībā pret konkrētu tematu. Daudziem tvītiem ir pieejama arī informācija par ģeolokāciju, kas papildus ļaus lokalizēt iegūto informāciju, kā arī dos iespēju salīdzināt sabiedrības viedokļu atšķirības atkarībā no lokācijas. Sentimenta analīzes risinājuma izstrādes laikā paredzēts izstrādāt latviešu valodas Twitter sentimenta korpusu.

Darba teorētiskā daļa ir sadalīta piecās nodaļās. Pirmajā nodaļā ir aprakstīts, kas ir sentimenta analīze un kādas pētāmās problēmas tā ietver. Otrajā nodāļā tiek virspusīgi izklāstīti esošie sentimenta analīzē izmantotie risinājumi. Trešajā nodaļā ir aprakstīta pamatinformācija par mākslīgajiem neironu tīkliem, to veidiem. Ceturtajā nodaļā ir aprakstīts kādā veidā tika veikta sentimenta analīzei nepieciešamo datu iegūšana un apstrādāšana. Piektajā nodaļā ir aprakstīta mākslīgā neironu tīkla izveidošanas un optimizēšanas process, kā arī iegūtā risinājuma tālāka pielietošana.

1. SENTIMENTA ANALĪZE

Šajā nodaļā ir aprakstīts, ko šī darba kontekstā nozīmē sentiments, kas ir sentimenta analīze un kādi aspekti tiks ņemti vērā šī darba izstrādes laikā.

Sentimentam eksistē daudz un dažādai raksturojumi un definīcijas, jo tam pēc tā būtības piemīt nenoteiktība. Visbiežāk sentimentu mēdz definēt kā vienu no viedokļa četrinieka vai piecinieka sastāvdaļām. Viedokļa četrinieks vai piecinieks ir daudzos sentimenta analīzes pētījumos izmantots veids, kā definēt viedokli. [2, 3, 4] Piemērs viedokļa četriniekam – *(g, s, h, t)*, kur *g* ir viedokļa vai sentimenta mērķa entītija, *s* ir sentiments attiecībā pret *g*, *h* ir viedokļa turētājs un *t* ir laiks, kad šis viedoklis tika izteikts. [4]

Jēdziens sentimenta jeb noskaņojuma analīze, ko pazīst arī kā viedokļu ieguvi, tiek skaidrots kā pētniecības nozare, kas analizē cilvēku viedokļus, noskaņojumus, vērtējumus, atzinumus, nostājas un emocijas attiecībā pret kādu konkrētu entītiju. Šī entītija var būt jeb kas, par ko cilvēks varētu izteikt viedokli – produkts, notikums, cilvēks, pakalpojums, temats u.tml. [4]

1.1. Analīzes līmeņi

Kopumā sentimenta analīzes process satur plašu veicamo uzdevumu kopu, kas variējas atkarībā no situācijas un mērķa, kam tā tiek pielietota. Sentimenta analīzi parasti iedala trīs analīzes līmeņos, balstoties uz teksta apjomu, kam tā tiks veikta:

- **Dokumenta līmeņa analīze** – šī līmeņa uzdevums ir klasificēt, vai dokumenta kopējais sentiments bija pozitīvs vai negatīvs. Kā piemēru šāda līmeņa analīzei var minēt produkta vai filmas atsauksmes sentimenta klasificēšanu. IMDB movie database. Šajā analīzes līmenī tiek pieņemts, ka dokumentā ir aprakstīta informācija tikai par vienu entītiju, jo pretējā gadījumā nebūtu iespējams noteikt par kuru no entītijām ir izteikts viedoklis;
- **Teikuma līmeņa analīze** – šī līmeņa uzdevums ir klasificēt vai teikumā esošās informācijas sentiments ir pozitīvs, neitrāls vai negatīvs. Pateicoties neitrālajai sentimenta klasei, šī līmeņa analīze ir cieši saistīta ar subjektivitātes klasifikācijas uzdevumu, kura mērķis ir atšķir teikumus, kas satur uz faktiem bāzētu informāciju, no teikumiem, kas satur subjektīva rakstura informāciju;
- **Entītijas un aspekta līmeņa analīze** – atšķirībā no iepriekšējiem, šajā līmenī uzdevums ir identificēt par kuru entītiju un tās atbilstošo raksturiezīmi tika izteikts

viedoklis. Attiecīgi šī līmeņa analīzes veikšanai nepieciešami vēl papildus uzdevumi saistībā ar entītiju un to aspektu ekstrakciju, tādējādi sarežģot sentimenta analīzes procesu. [4]

1.2. Viedokļu veidi

Lai nodemonstrētu sentimentam piemītošās īpašības, tiks izmantots vienkāršs viedokļa četrinieka modelis ar sekojošām sastāvdaļām – *[Tēma, Turētājs, Apgalvojums, Sentiments]*. Balstoties uz šo viedokļa modeli, var apgalvot, ka katram viedoklim ir autors jeb viedokļa *turētājs,* kas ir izteicis *apgalvojumu* par kādu konkrētu *tēmu*, taču ne katram viedoklim var piemist *sentiments*. Piemēram, viedoklim "*šodien ārā ir saulains laiks*" piemīt apgalvojums, bet tas nesatur sentimentālu informāciju. Savukārt viedokļi, kas satur sentimentālu informāciju, tiek iedalīt implicītos vai eksplicītos, atkarībā no tā, kādā veidā tiek pasniegta sentimentālā informācija. Piemēram, "*es domāju, ka viņš izdarīja visu, ko varēja*" ir viedoklis ar implicītu sentimentu, bet "es esmu apmierināts ar viņa veikumu" ir viedoklis ar eksplicītu sentimentu. [3]

Vēl viens veids kā iedalīt viedokļus ir iedalīt tos tiešos vai netiešos regulāros viedokļos un salīdzinošos viedokļos. Šādā viedokļu tipu dalījumā regulāri viedokļi tiek skaidroti kā vienkārši viedokļi, kas tiek iedalīti tiešos un netiešos. Piemēram, tiešs regulārs viedoklis ir "*šī mūzika ir ļoti skaista*", bet netiešs regulārs viedoklis ir "*jaunais programmatūras atjauninājums piereģistrēšanas procesu padarīja tikai sarežģītāku*". Savukārt, salīdzinošs viedoklis parasti satur divu entītiju salīdzinājumu, piemēram "*manuprāt rudens ir labāks par pavasari*". [4]

1.3. Twitter sentimenta analīze

Ņemot vērā, ka šajā darbā sentimenta analīzei paredzētā informācija tiks ņemta no Twitter sociālā tīkla, kurā ziņu garumi tiek limitēti līdz 140 simboliem, var secināt, ka visatbilstošākā ir teikuma līmeņa sentimenta analīze. To apliecina arī iepriekšējie pētījumi saistībā ar sentimenta analīzes veikšanu Twitter datiem. [5]

Twitter sociālajā tīklā pieejamo informāciju pamatā var iedalīt atkarībā no lietotāja veida. Twitter lieto gan privātā/personīgā līmenī, gan organizācijas līmenī. Personīgie lietotāju profili parasti tiek identificēti ar attiecīgās personas vārdu un uzvārdu, bet organizāciju profili ar attiecīgās organizācijas nosaukumu. Kā piemēru organizācijas profiliem var minēt tādus Twitter profilus, kā "*LatvijasUniversitāte*", "*NASA*", "*BBC News*", "*MIT Media Lab*" u.tml.

Šādu profili tvīti visbiežāk satur informāciju, kurai nepiemīt sentimentāla vērtība, attiecīgi būtu vēlams minimizēt informācijas apjomu, kas ņemta no šādu veidu profiliem.

Salīdzinot citiem klasiskiem sentimenta analīzes uzdevumiem, Twitter sentimenta analīze izceļas ar valodu. Visbiežāk tvītos tiek izmantota neformāla valoda, kas var apgrūtināt vārdu atpazīšanu gadījumā, ja lietoti gramatiski nekorekti vārdi. Ņemot vērā arī iepriekš minēto simbolu skaita limitu, tvītos bieži vien tiek izmantoti vārdu saīsinājumi, piemēram, "*tlt*", "*kpc*", kā arī vārdi vai teikuma struktūra var tikt izmainīta, lai iekļautos noteiktajā simbolu skaita limitā. Vēl viena būtiska atšķirība starp citiem klasiskiem sentimenta analīzes uzdevumiem un Twitter sentimenta analīzi ir tāda, ka tvītos esošā informācija var būt visdažādākajā kontekstā. Piemēram, veicot produktu atsauksmju sentimenta analīzi, ir zināms, ka analizējamā informācija būs ar sentimentālu raksturu un tā būs attiecināta uz vērtējamo produktu. Savukārt, tvīti var saturēt visdažādākā veida informāciju un tiem nav noteikti tematiskā rakstura rāmji. Attiecīgi, sentimenta analīzei paredzēto datu priekšapstrādē būs nepieciešams apstrādāt aprakstītās situācijas.

Sentimenta analīze parasti tiek veikta vienas valodas ietvaros, taču, piemēram, latviešu sarunvalodā dažkārt tiek izmantoti arī angļu valodas vārdi, piemēram, "*šis ir kind of good piemērs*". Pastāv iespēja veikt arī daudzvalodu sentimenta analīzi, taču tas prasa papildus uzdevumu veikšanu, piemēram, mašīntulkošanu. [6] Ņemot vērā šajā darbā izmantoto sentimenta analīzes risinājuma specifiku, daudzvalodu sentimenta analīzi iespējams veikt arī neizmantojot mašīntulkošanu. Ceturtās nodaļas ceturtajā apakšnodaļā tiks sīkāk aprakstīts kādā veidā ar šajā darbā izmantoto risinājuma metodi iespējams panākt daudzvalodu sentimenta analīzi. Ceturtajā nodaļā tiks arī apskatīts kāds ir svešvalodu vārdu īpatsvars iekš darba praktiskajā daļā iegūtā tvītu korpusa, lai noteiktu daudzvalodu sentimenta analīzes nepieciešamību.

Iepriekš tika minēts, ka teikuma līmeņa sentimenta analīzē parasti netiek veikta sentimenta mērķa entītijas identificēšana. Šī darba ietvaros radītās sistēmas lietojuma scenārijā būs jau zināms kādai entītijai tiks veikta sentimenta klasificēšanai, jo tā tiks veikta tvītiem, kuri tiks atlasīti atbilstoši lietotāja ievadītajiem atslēgas vārdiem.

2. ESOŠIE RISINĀJUMI

Kā jau tika minēts iepriekšējā nodaļā, sentimenta analīze tiek pielietota dažāda veida informācijai, un attiecīgi eksistē daudz un dažādi risinājumi. Tāpēc, ņemot vērā, ka par datu avotu ir izvēlēts Twitter sociālais tīkls, šajā nodaļā tiks aprakstītas populārākās praksē lietotās metodes Twitter sentimenta analīzei.

2.1. Novērtēšana

Lai varētu novērtēt esošo risinājumu, kā arī šajā bakalaura darbā izstrādātā risinājuma darbības rezultātus, nepieciešams definēt pēc kādiem veiktspējas rādītājiem risinājumi tiks vērtēti. Balstoties uz šī darba izstrādes laikā apskatītajiem pētījumiem, visbiežāk izmantotās veiktspējas novērtēšanas metrikas ir **Precizitāte (P), Pārklājums (R)** un **F_1 mērs (F_1)**. Papildus iepriekš minētajiem rādītājiem, klasifikācijas problēmu risinājumos, lai atainotu klasifikatora rezultātus, bieži tiek izmantota arī pārpratumu matrica, kurā uzskatāmi tabulas veidā (skat. 2.1. tabulu) var uzrādīt klasifikatora novērtējumu sadalījumu pa klasēm.

	Prognozētais P	**Prognozētais N**
Patiesais P	**Patiesais Pozitīvs (TP)**	**Nepatiesais Negatīvs (FN)**
Patiesais N	**Nepatiesais Pozitīvs (NP)**	**Patiesais Negatīvs (TN)**

2.1. tabula **Pārpratumu matrica divām klasēm**

Balsoties uz iepriekš minēto pārpratumu matricu, darbā izmantotās metrikas tiek definētas šādi:

- **Precizitāte (P)** apzīmē to, cik liela daļa no prognozētajām vērtībām ir pareizi prognozētas vērtības. To var definēt arī kā varbūtību, ka nejauši izvēlēts elements no prognozēto vērtību kopas būs pareizi prognozēto vērtību kopā. To aprēķina pēc šādas formulas:

$$P = \frac{TP}{TP + FP}$$

- **Pārklājums (R)**, ko mēdz definēt arī kā jutīgumu, ir pareizi prognozēto vērtību attiecība pret kopējo pareizo vērtību skaitu. To var definēt arī kā varbūtību, ka nejauši izvēlēts elements no visu pareizo vērtību kopas būs prognozēto vērtību kopā. To aprēķina pēc šādas formulas:

$$R = \frac{TP}{TP + FN}$$

- **F_1 mērs (F_1)** ir precizitātes un pārklājuma harmoniskais vidējais. To aprēķina pēc šādas formulas

$$F_1 = 2 \times \frac{P \times R}{P + R}$$

Papildus F_1 mēram tiek lietoti arī F_2 mērs un $F_{0.5}$ mērs, kur F_2 palielina pārklājuma svaru, bet $F_{0.5}$ samazina pārklājuma svaru. Šajā darbā tiks izmantots tikai standarta F_1 mērs. F mēra vispārējā formula ir šāda:

$$F_\beta = (1 + \beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R}, kur\ \beta \in \mathbb{R}\ un\ \beta > 0$$

Neskatoties uz iepriekš definētajām metrikām, jāatceras, ka sentimenta klasifikācija ir savā ziņā subjektīva problēma, jo teksta emocionālā nokrāsa nav diskrēts lielums, un katrs cilvēks emocijas interpretē citādāk. Par to liecina arī pētījums, kura rezultātos tika novērots, ka cilvēku veiktas sentimenta analīzes rezultāti sakrita tikai 79% gadījumos [7], taču šajā rakstā netika minēts, kā tika iegūti šie rezultāti, kas mazina šī apgalvojuma ticamību.

2.2. Klasiskās mašīnmācīšanās metodes

Šajā apakšnodaļā ir apskatīti trīs populārāko sentimenta analīzes risinājumu rezultāti. Par pamatu šo metožu rezultātiem tiks ņemts Stenfordas Universitātē izstrādātais pētījums, [8] jo šajā pētījumā ir sasniegti salīdzinoši labi rezultāti (skat. 2.2. att.), kā arī šajā pētījumā izmantotā datu kopa ir publiski pieejama, kas ļaus veikt precīzu iepriekšējo risinājumu un bakalaura darbā izstrādātā risinājuma salīdzināšanu. Precīzāks datu kopas apraksts ir pieejams 4. nodaļā.

Table 6: Classifier Accuracy

Features	Keyword	Naive Bayes	MaxEnt	SVM
Unigram	65.2	81.3	80.5	82.2
Bigram	N/A	81.6	79.1	78.8
Unigram + Bigram	N/A	82.7	83.0	81.6
Unigram + POS	N/A	79.9	79.9	81.9

2.2. att. **Stenfordas Universitātē izstrādātā pētījuma rezultāti** [8]

2.2.1. Naivais Baiesa klasifikators

Viens no populārākajiem un salīdzinoši vienkāršākajiem sentimenta analīzes risinājuma veidiem ir izmantot naivo Baiesa (*Naive Bayes*) klasifikatoru. Naivais Baiesa klasifikators ir viens no varbūtisko klasifikatoru veidiem, kas pamatā balstās uz Baiesa teorēmu. [9]

Neskatoties uz savu vienkāršumu, šis klasifikators ir ļoti efektīvs, un, pie nelieliem datu apjomiem, tas bieži vien sniedz labākus rezultātus nekā citi sarežģītāki risinājumi. [10]

Kā redzams 2.2. attēlā, naivais Baiesa klasifikators uz 1,6 miljonu tvītu treniņu datu kopas labākajā gadījumā sasniedza 82,7% precizitāti.

Vēl viens šīs metodes praktiska pielietojuma piemērs ir pieejams *ISWC 2012* konferences darbā esošajā pētījumā par Twitter semantisko sentimenta analīzi. [11] Šajā pētījumā tika izmantotas trīs Twitter datu kopas, no kurām viena (Stenfordas Twitter sentimenta korpuss) ir ņemta no iepriekš minētā pētījuma. Apkopojot visu trīs datu kopu rezultātus, vidēji tika sasniegti šādi rādītāji: 77,18% precizitāte, 75,33% pārklājums un 75,95% F_1 mērs. Izmantojot tikai Stenfordas Twitter sentimenta korpusu, pētījuma autori apgalvo, ka ir sasnieguši šādus rezultātus: 84,25% precizitāte, 83,80% pārklājums un 83,90% F_1 mērs. Jāpiebilst, ka šajā pētījumā tika izmantoti tikai 60 000 tvīti no visa kopējā 1,6 miljonu tvītu lielā Stenfordas Twitter sentimenta korpusa.

2.2.2. Maksimuma entropijas klasifikators

Maksimuma entropija jeb multinomiālā loģistiskā regresija ir varbūtības sadalījuma novērtēšanas veids, kas tiek pielietos dažādiem NLP uzdevumiem. [12] Kā redzams 2.2. attēlā, šī metode uz 1,6 miljonu tvītu treniņu datu kopas labākajā gadījumā sasniedza 83% precizitāti, kas, atbilstoši pētījuma autoru izteiktajai prognozei, pārsniedz naivā Baiesa metodes sasniegtos rezultātus. [8]

2.2.3. Atbalsta vektoru mašīnas

Atbalsta vektoru mašīnas (SVM) ir vēl viens izplatīts klasifikācijas paņēmiens. Kā redzams 2.2. attēlā, šī metode uz 1,6 miljonu tvītu treniņu datu kopas labākajā gadījumā sasniedza 82,2% precizitāti. Lai arī SVM palīdzību dažkārt var iegūt nedaudz labākus rezultātus, šī klasifikatora izmantošana būtiski palēnina procesu. [13]

2.3. Mākslīgo neironu tīklu risinājumi

Šajā apakšnodaļā ir aprakstīti uz mākslīgajiem neironu tīkliem bāzēta Twitter sentimenta analīzes risinājuma iegūtie rezultāti.

Kā tika novērots IBM pētnieku veiktajā pētījumā [14], ar dziļo konvolūcijas tīklu palīdzību iespējams sasniegt ievērojami labākus rezultātus nekā ar klasiskajām mašīnmācīšanās

metodēm. Pētījumā tika izmantota iepriekš aprakstītā Stenfordas Universitātē izstrādātā Twitter datu kopa. Izmantojot pētījumā aprakstīto no simbolu uz teikumu līmeņa informācijas bāzēto konvolūciju tīklu (CharSCNN), pētījuma autoriem labākajā gadījumā izdevās iegūt 86,4% precizitāti, kas pētījuma izstrādes brīdī, balstoties uz autoru apgalvojumu, bija labākais uz attiecīgās datu kopas iegūtais rezultāts.

3. MĀKSLĪGIE NEIRONU TĪKLI

Šajā nodaļā ir īsumā aprakstīta pamatinformācija saistībā ar mākslīgajiem neironu tīkliem un to darbību.

3.1. Vēsture

Par mākslīgo neironu tīklu pirmsākumiem tiek uzskatīts neirofiziologa W. McCulloch un loģiķa W. Pits 1943. gadā radītais mākslīgā neirona matemātiskais modelis. [15]

Kā nākošo soli mākslīgo neironu tīklu vēsturē var uzskatīt fiziologa D. Hebb 1949. gadā izdoto grāmatu, kurā tika izteikta hipotēze, ka mācību procesā tiek izmainīta smadzeņu šūnu struktūra –konkrēts ceļš smadzeņu neironu tīklā tiks pastiprināts pēc katras tā izmantošanas reizes. Šajā grāmatā minētā teorija kalpo kā pamats mācību procesa aprakstam un to mēdz saukt arī kā Hebba postulātu. [16]

Nākošais būtiskais solis mākslīgo neironu tīklu vēsturē ir 1986. gadā izdotais raksts, kurā tika aprakstīta kļūdu atgriezeniskā izplatīšanas (*back-propagation*) metode, kas ļāva veikt vairāku slāņu neironu tīklu trenēšanu. [17]

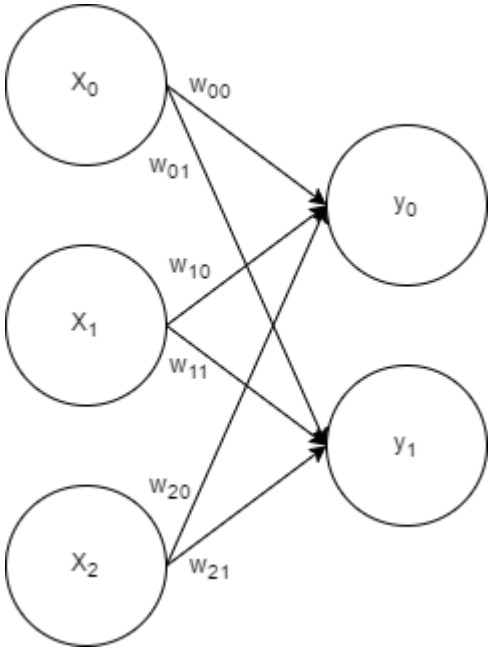
Mūsdienās mākslīgie neironu tīkli savu popularitāti atguvuši pateicoties pieejamo tehnoloģiju jaudai. NVIDIA kompānijas izveidotā CUDA paralēlās skaitļošanas platforma sniedz ļoti būtisku skaitļošanas veiktspējas uzlabojumu, kas vairākkārt paātrina apjomīgu skaitļošanas problēmu risināšanu, attiecīgi uzlabojot arī mākslīgo neironu tīklu pielietošanas iespējas. [18]

3.2. Mākslīgo neironu tīkla uzbūve un darbība

Pašlaik viena no vispopulārākajām mākslīgo neironu tīklu arhitektūrām ir vairākslāņu perceptrons (**MLP**), kurš parasti tiek trenēts ar kļūdu atgriezenisko izplatīšanas metodi. MLP struktūras pamatā ir vairāki slāņi ar savstarpēji savienotiem mākslīgajiem neironiem. Vairākslāņu perceptrona ieejai tiek izmantots ieejas slānis, kas ieejā padotās vērtības padod tālāk uz slēptajiem slāņiem. Šo var uztver kā mākslīgo neironu aktivēšanu, kuras rezultātā tiek veikta katra slēptā slāņa neirona ievadu svērtā summēšana, kur katram summēšanas rezultātam tiek pieskaitīta attiecīgā neirona nosliece (*bias*). Pēc noslieces pieskaitīšanas, iegūtais rezultāts tiek padots neirona aktivizācijas funkcijai, ar kuras palīdzību tiks aprēķināta neirona galējā

vērtība, kura mākslīgā neirona aktivizācijas rezultātā tiks izvadīta pa mākslīgā neirona izejas savienojumu ar nākošo slēpto slāni vai izejas slāni.

Vienslāņa perceptrona gadījumā (skat. 3.1. att.) tīkls sastāv tikai no ieejas slāņa un izejas slāņa. Attiecīgi, izejas slāņa neironu aktivizācijas funkcijas rezultāts ir tīkla kopējais rezultāts.



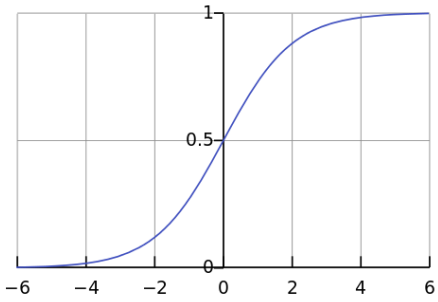
3.1. att. **Vienslāņa perceptrona piemērs**

Dotajā piemērā esošā vienslāņa perceptrona izejas slāņu mākslīgo neironu atgrieztās vērtības var aprēķināt ar šādu formulu:

$$y_i = \sigma(S_i), \;\; kur\; S_i = \sum_{j=0}^{n} X_j w_{ij} + b_i \,, kur\; n = ievades\; slāņa\; neironu\; skaits$$

Šajā formulā ar $\sigma(S_i)$ apzīmē aktivizācijas funkciju. Aktivizācijas funkciju parasti piemeklē tā, lai to būtu viegli atvasināt (skat. 3.3. apakšnodaļu), un visbiežāk tā ir sigmoīda funkcija (skat. 3.2. att.), kuru aprēķina šādi:

$$\sigma(x) = \frac{1}{1 + \; e^{-x}}$$



3.2. att. **Sigmoīda funkcijas grafiks**[1]

[1] Pieejams https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Logistic-curve.svg/500px-Logistic-curve.svg.png

Vairākslāņu perceptrona gadījumā saglabājas tie paši vienslāņa perceptrona darbības principi. Atšķirība ir tāda, ka starp ieejas slāni un izejas slāni ir slēptie slāņi, kur parasti pakāpeniski viens no otra saņem vērtības virzienā no ieejas slāņa uz izejas slāni. [19]

3.3. Trenēšana

Mākslīgo neironu tīklu trenēšanu parasti iedala trīs paradigmās:

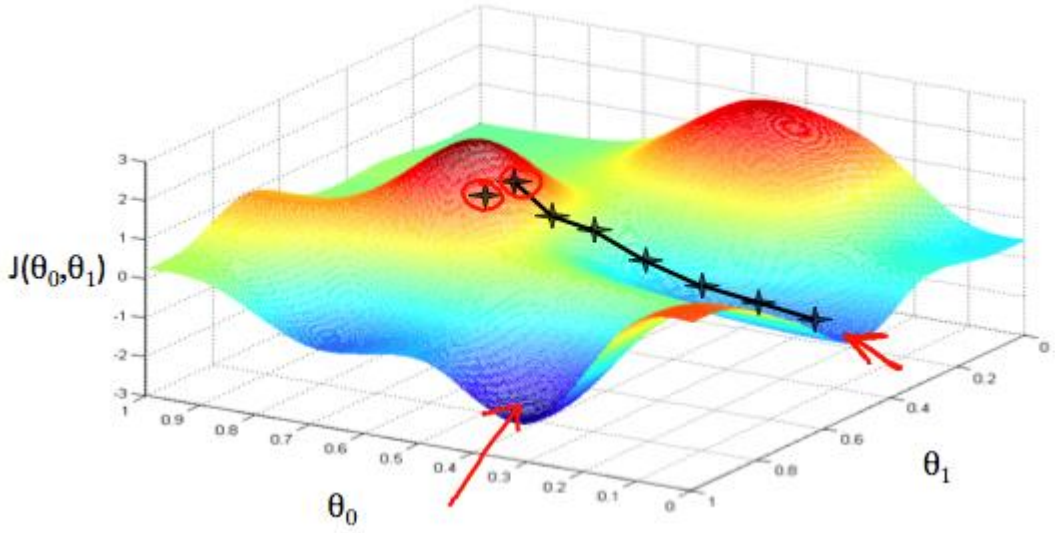
- **Pārraudzītā mācīšanās** (*supervised learning*) – šajā mācīšanās procesā tīklam katru reizi tiek padota pareizā atbilde, atbilstoši padotajam ievadam. Tādējādi mākslīgais neironu tīkls iemācās sakarības starp to, kādiem ievaddatiem atbilst kādi izvaddati;
- **Stimulētā mācīšanās** (*reinforcement learning*) – šis ir viens no pārraudzītās mācīšanās veidiem. Šajā procesā mākslīgajam neironu tīklam pareizo atbilžu vietā tiek padots tikai novērtējums par to, cik pareizs vai nepareizs ir tīkla rezultāts;
- **Nepārraudzītā mācīšanās** (*unsupervised learning*) – šajā mācīšanās procesā tīklam netiek padotas atbildes. Tīkla darbība balstās tikai uz ievaddatu struktūras izpēti. [20]

Ņemot vērā, ka šim bakalaura darbam uzstādīta problēma tiks risināta ar pārraudzītās mācīšanās metodi, turpmāk tiks sīkāk aprakstīts tikai pārraudzītās mācīšanās process.

Vairākslāņu perceptrona trenēšana ir process, kurā tiek pielāgotas mākslīgo neironu savienojumu svaru un nosliečų vērtības. Šīs vērtības var uzskatīt par perceptrona maināmajiem parametriem. Trenēšanas procesā, perceptronam tiek padoti treniņa piemēra ieejas dati, kā rezultātā notiek mākslīgo neironu aktivizācija un izejas slānī tiek iegūts rezultāts atbilstoši uzstādītajiem svariem un nosliecēm. Iegūtais rezultāts pēc tam tiek salīdzināts ar treniņa piemēra izejas datiem, lai noteiktu perceptrona kļūdas intensitāti. Trenēšanas rezultātā, mainīgie parametri tiek pārmainīti tā, lai minimizētu perceptrona kļūdas intensitāti. Attiecīgi, nepieciešams definēt pēc kāda principa tiks mainīti parametri, lai minimizētu kļūdas intensitāti. [21]

Viens no paņēmieniem kļūdas intensitātes mazināšanai ir gradienta nolaišanās (*gradient descent*) metode. Šī metode balstās uz kļūdas noteikšanas funkciju. Gradienta nolaišanās metodes procesā tiek veikta kļūdas noteikšanas funkcijas atvasināšana, kā rezultātā tiek iegūta informācija par to, cik ļoti kļūdas funkcija izmanās konkrētā funkcijas definīcijas punktā, kas ļauj noteikt, kurā virzienā nepieciešams iet, lai samazinātu kļūdas funkcijas vērtību. Attiecīgi, perceptrona parametri atbilstoši soļa lielumam tiek mainīti tā, lai tiktu iegūta stāvākā gradienta

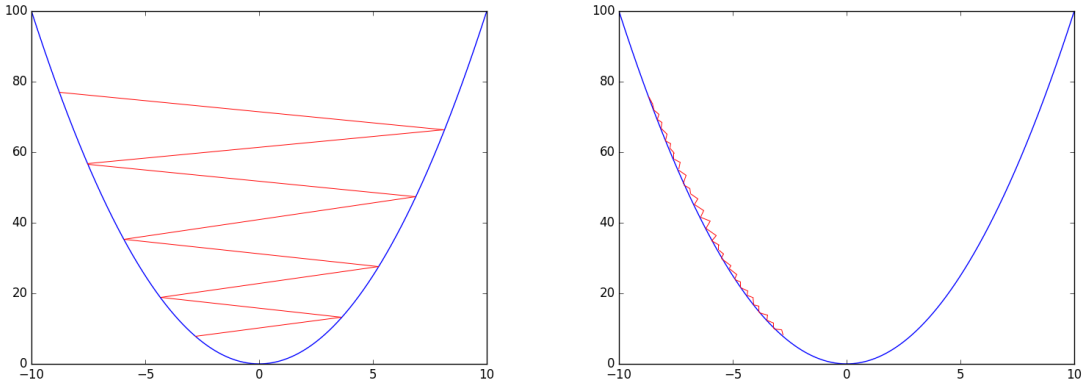
nolaišanās, t.i. tiek meklēts virziens, kurā ir visstāvākais kļūdas funkcijas kritums (skat. 3.3. att.). Ar soļa lielumu šajā kontekstā saprot to cik ļoti tiek izmainīti perceptrona parametri konverģences virzienā, un to apzīmē kā mācīšanās intensitāti (*learning rate*). Šis process tiek atkārtots līdz kļūdas funkcija konverģē kādā no minimumiem.



3.3. att. **Gradienta nolaišanās piemērs divargumentu kļūdas funkcijai** [22]

Kā redzams gradienta nolaišanās piemērā (skat. 3.3. att.), kļūdas funkcija var konverģēt dažādos minimumos, atkarībā no sākuma stāvokļa. No tā var secināt, ka kļūdas funkcija ne vienmēr konverģēs globālajā minimumā, kas attiecīgi var novest pie situācijas, kurā trenēšanas procesa turpināšana vairs neuzlabo perceptrona parametrus, kaut gan pastāv iespēja iegūt labāku rezultātu. Viens vieds, kā cīnīties ar funkcijas konverģēšanu neoptimālā lokālajā minimumā, ir palielināt mācīšanās intensitāti, tādējādi palielinot soļu attālumu, kas attiecīgi ļautu "izkāpt" ārā no funkcijas lokālā minimuma, taču pārāk liela mācīšanās intensitāte var likt pāriet pāri lokālajam minimumam, tā rezultātā lieki oscilējot ap lokālo minimumu vai pat likt funkcijai diverģēt. Savukārt, pārāk maza mācīšanās intensitāte var ievērojami palēnināt konverģēšanas procesu (skat. 3.4. att.). [23]

Vēl viens veids, kā optimizēt gradienta nolaišanos, ir pielietot adaptīvu mācīšanās intensitāti. Šīs metodes pamatideja ir paātrināt mācīšanās ātrumu, kur tas nepieciešams, un to palēnināt, kad funkcija sāk tuvoties minimumam. Treniņa procesa sākumā lielāka mācīšanās intensitāte ļauj izvairīties no funkcijas konverģēšanas neoptimālā lokālajā minimumā, bet, tuvojoties lokālajam minimumam, palēnināta mācīšanās intensitāte ļauj izvairīties no oscilēšanas ap lokālo minimumu. Tomēr, ne vienmēr ir nepieciešams veikt mācību intensitātes izmaiņu, jo gradienta nolaišanās tuvojoties minimumam automātiski veiks mazākus soļus. [24]



3.4. att. **Mācīšanās intensitātes ietekme uz funkcijas konverģenci**

Vēl viens būtisks aspekts pārraudzītās mācīšanas trenēšanas procesā ir tas, kādā veidā un cik daudz tiek padoti treniņa dati. Treniņa datus var padot vai nu pa vienam, vai pa partijām (*batch*). Padodot treniņa datus pa vienam, gradienta konverģēšana parasti notiek optimālāk, taču šāda veida treniņu datu padošana var būt neefektīva resursu izmantošanas ziņā, jo izmantojot šo pieeju, lielākā daļa no treniņa procesa laika tiek patērēta datu apmaiņā, nevis rezultātu skaitļošanā, attiecīgi šī pieeja neļauj izmantot skaitļošanas resursu pilno potenciālu. Lai risinātu šo problēmu, treniņa datus var padot pa partijām, tādējādi vienā reizē padodot lielāku treniņu datu skaitu, kas attiecīgi ļauj vienlaikus apstrādāt vairāk datus un mazāk patērēt laiku uz datu apmaiņu. Otrs aspekts, kas ietekmē mākslīgā neironu tīkla treniņa rezultātu ir tas, cik daudz reižu tīklam tiek padoti treniņa dati. Par epohu (*epoch*) sauc treniņa procesa ciklu, kurā mākslīgajam neironu tīklam tiek padoti visi treniņa dati. Ņemot vērā iepriekš aprakstīto problēmu par gradienta konverģēšanu neoptimālā lokālajā minimumā nevis globālajā minimumā, treniņa procesā parasti tiek veiktas vairākas epohas, lai nodrošinātu optimālāku gradienta konverģēšanu, taču pārāk liels epohu skaits var novest pie pārpielaikošanas (skat. 3.4. apakšnodaļu), kas nozīmē, ka tīkls sāk veikt datu iegaumēšanu nevis vispārējo sakarību iemācīšanos.

4. DATI

Jebkuras mākslīgo neironu sistēmas pamatā ir dati. Tie ir vieni no visbūtiskākajiem faktoriem, kas ietekmē mākslīgā neironu tīkla rezultātus un precizitāti. Neatbilstoša treniņu datu kopa var novest pie neprecīziem rezultātiem neatkarīgi no tā, cik labi ir izveidots mākslīgais neironu tīkls. Tāpēc ir svarīgi apzināties un izpētīt ar kādiem datiem būs nepieciešams saskarties darbam uzstādītās problēmas risināšanā. Attiecīgi šajā nodaļā ir aprakstīta informācija par datiem, kas tiek izmantoti sistēmas darbībā, kā arī ir aprakstīts, kā šī bakalaura darba uzstādītās problēmas risinājuma ietvaros iegūtie dati tiek apstrādāti, lai varētu veikt sentimenta analīzi.

4.1. Datu iegūšana

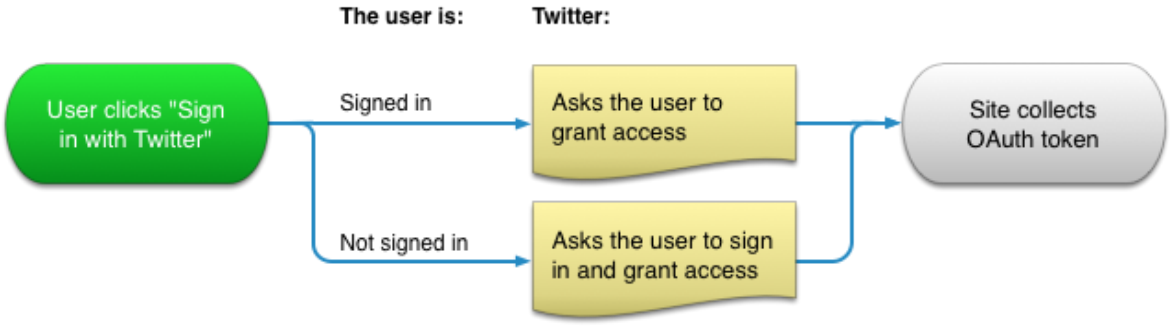
Twitter nodrošina dažādus veidus, kā iegūt nepieciešamos datus. Pamatā, tvītu iegūšanu var realizēt izmantojot vai nu Twitter REST API vai Streaming API. Balstoties uz Twitter sniegto informāciju, REST API ir paredzēts, lai veiktu singulārus meklējumus, lasītu lietotāju profilu informāciju, veiktu tvītu izlikšanu u.c. darbības, kas parasti tiek veiktas ar REST API palīdzību. Savukārt, Streaming API nodrošina reāla laika piekļuvi datiem, ko parasti izmanto, lai sekotu konkrētam tematam vai lietotājam, kā arī, lai veiktu datizraci. [25, 26]

Tā kā šajā darbā izstrādātās sistēmas nolūks ir veikt tvītu meklēšanu atbilstoši lietotāja ievadītiem atslēgas vārdiem, tad sīkāk tiks apskatīts Search API, kas ir daļa no iepriekš minētā REST API. Search API nodrošina dažādu vaicājumu veikšanu, kuros var iekļaut tādus nosacījumus, kā meklēšanu pēc atslēgas vārdiem, sasaistot tos vai nu ar disjunkciju vai ar konjunkciju, frāžu meklēšanu, konkrētu atslēgas vārdu neiekļaušanu, attēlu vai video saturošu tvītu izfiltrēšanu u.c. Būtiski ir piebilst, ka, izmantojot Search API, ir pieejami tikai pēdējo 7 dienu dati un šis API fokusējas uz rezultātu atbilstību nevis pilnību, kā arī Search API tiek pakļauts atbilstošajiem REST API pieprasījumu ierobežojumiem. [25]

Kā alternatīvu iepriekš minētajām bezmaksas metodēm, Twitter piedāvā komerciālu pakalpojumu Gnip, kas ir uzņēmumiem paredzēts API. Atšķirībā no bezmaksas Search API, Gnip nodrošina piekļuvi visiem Twitter vēsturiskajiem datiem. [27]

Lai varētu izmantot iepriekš minēto Search API, nepieciešams veikt OAuth autorizāciju. To iespējams veikt divos līmeņos – lietotāja līmeņa autentifikācija vai lietotnes līmeņa autentifikācija. Lietotāja līmeņa autentifikācijā, sistēmas lietotājam nepieciešams autorizēties ar savu Twitter lietotāja profilu, kā rezultātā viņš iegūst piekļuves marķieri, ar kura palīdzību būs iespējama tālāka Twitter API lietošana. Šādā gadījumā arī REST API ierobežojumi tiek

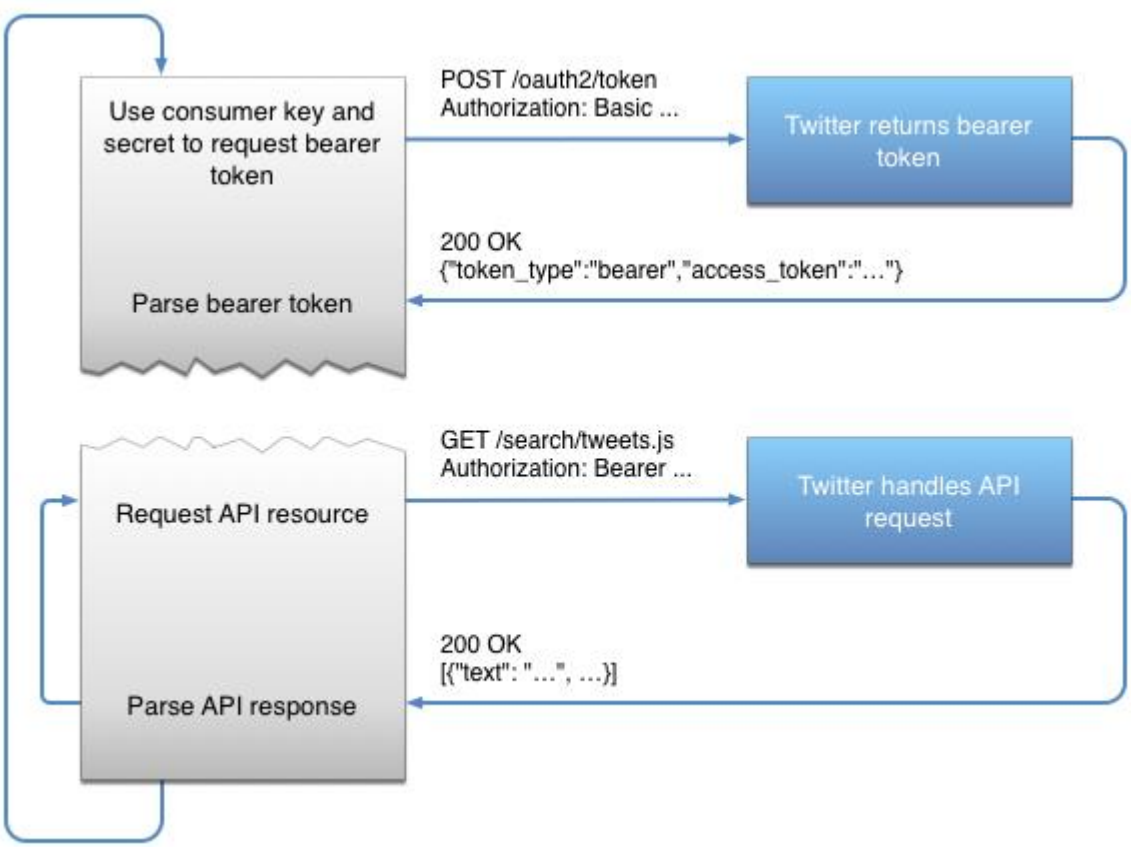
pielietoti lietotāja līmenī. Attēlā 4.1. ir redzami iespējamie lietotāja autorizēšanās stāvokļi šī līmeņa autentifikācijā. [25]



4.1. att. **Twitter API lietotāja līmeņa trīs soļu autorizācija**[2]

Vadoties pēc Twitter izstrādātāju dokumentācijas, lietotnes līmeņa autentifikācija ir veidota atbilstoši OAuth 2 specifikācijai. Tā kā autentifikācija notiek lietotnes līmenī, lietotnei nebūs iespēja piekļūt lietotāja konteksta līmeņa API galapunktiem. Šī līmeņa autorizācija tiek veikta nosūtot ar Base64 kodējumu nokodētu lietotnei piešķirtā patērētāja atslēgu un noslēpumu, kā rezultātā tiek saņemts nesēja marķieris, kas nodrošinās tālāku API izmantošanu. Lai varētu izmantot šo autentifikācijas veidu, visos pieprasījumos nepieciešams, lai abi pieprasījuma galapunkti izmantotu HTTPS, attiecīgi nepieciešams arī SSL sertifikāts. Lietotnes līmeņa autentifikācijai ir atšķirīgi API pieprasījumu skaita ierobežojumi, taču tie tiek pielietoti lietotnes līmenī. Šāda līmeņa autorizācijas priekšrocība ir tas, ka lietotājam nav nepieciešams autorizēties ar Twitter profilu, bet trūkums ir tas, ka API pieprasījumu sakaita ierobežojums var tikt sasniegts daudz ātrāk, ja lietotnei ir vairāki aktīvi lietotāji. Attēlā 4.2. ir ilustrēts lietotnes līmeņa autentifikācijas process. [25]

[2] Pieejams https://dev.twitter.com/oauth/3-legged



4.2. att. **Twitter API lietotnes līmeņa autentifikācijas process**[3]

Šajā darbā izstrādātas sistēmas darbības ietvaros nepieciešamos datus var iedalīt divās daļās – trenēšanai paredzētie dati un sistēmas lietojumam paredzētie dati. Trenēšanai paredzētās datu kopas tiks izmantotas sistēmas izstrādes procesā, kā arī mākslīgā neironu tīkla trenēšanas un testēšanas procesā, bet sistēmas lietojumam paredzētie dati tiks izmantoti sentimenta analīzes procesā, sistēmas lietošanas laikā.

Sistēmas izveidošanas un mākslīgā neironu tīkla trenēšanas procesā tiks izmantoti sekojošie tvītu korpusi:

- Sentiment140 – 1,6 miljonu tvītu datu kopa (pazīstama arī kā Stenfordas Twitter sentimenta korpuss),[4] kas satur bināri klasificētus tvītus angļu valodā. Balstoties uz datu kopas autoru publicētajā projekta pārskatā esošo informāciju, tie ir anotēti automatizētā procesā balstoties uz tvītā esošajām emocijzīmēm, piemēram, tvīts, kas satur ‘:)’ emocijzīmi, tiek uzskatīts kā pozitīvs, bet tvīts, kas satur ‘:(’ emocijzīmi, tiek uzskatīts kā negatīvs. Pēc tvītu anotēšanas, tekstā esošās emocijzīmes ir noņemtas; [8]

[3] Pieejams https://dev.twitter.com/oauth/application-only
[4] Pieejams http://help.sentiment140.com/for-students/

- Jāņa Peisenieka bakalaura darbā izveidotais latviešu tvītu korpuss, kas satur 1177 pozitīvi, neitrāli vai negatīvi cilvēka klasificētus tvītus. Publiski pieejams **FnTm/latvian-tweet-sentiment-corpus** GitHub repozitorijā;[5] [28]
- Darba autora izveidots latviešu tvītu korpuss, kas satur 3131 pozitīvi, neitrāli vai negatīvi cilvēka klasificētus tvītus. Publiski pieejams **nicemanis/LV-twitter-sentiment-corpus** GitHub repozitorijā;[6]

Sistēmas lietojumam paredzētie dati tiks iegūti sistēmas lietošanas laikā, ar iepriekš aprakstīto Twitter Search API palīdzību. Ņemot vērā, ka sākotnēji sistēmai netiek paredzēts liels aktīvo lietotāju skaits, tiks izmantota lietotnes līmeņa Twitter API autentifikācija. Meklēšanas vaicājumiem tiks pievienoti sekojošie uzstādījumi:

- "*exclude:retweets*" – šis operators atgriežamo tvītu kopā neiekļauj tvītus, kas ir retvīti. Tas attīra rezultātus no tvītiem ar vienādu saturu, jo, ja vaicājumā tiek meklēti aktuāli atslēgas vārdi, tad rezultātos visticamāk tiks atgriezti daudzi rezultāti ar vienādu saturu, jo tie saturēs retvītus no zināmu organizāciju vai prominentu cilvēku profiliem;
- "*-RT*" – šo operatoru konkatenē pie vaicājumā esošajiem atslēgas vārdiem. Tas attīrīs rezultātus no tvītiem, kuru saturā ir vārds "RT". Tas nepieciešams, lai attīrītu datus no retvītiem, kas sistēmas līmenī netiek uzskatīti par retvītiem;
- "*lang:lv*" – šis operators nodrošina to, ka vaicājuma rezultātā atgrieztā informācija būs tajā valodā, kuras kods tiks padots pieprasījumā. Valodu kodi atbilst ISO 639-1 standartam.[7] Šāda funkcionalitāte nepieciešama, lai nodrošinātu korektu sentimenta analīzi, atbilstoši izvēlētajai valodai. [25]

4.2. Treniņdatu sagatavošana

Darbam izstrādes sākumā tika veikta latviešu valodas Twitter ziņu sentimenta korpusa izveidošana. Sākotnēji dati tika vākti ar darba ievadā minēto informācijas ieguves sistēmas izveidotā koncepta palīdzību. Datu korpusa izstrādes turpinājumā tika izveidots optimālāks nepieciešamo datu iegūšanas veids, kas ar Twitter API palīdzību ieguva viena lietotāja visu sekotāju pēdējos tvītus. Tā rezultātā tika savākts pietiekošs daudzums ar latviešu valodas Twitter datiem, kuriem pēc tam tika veikta manuāla trīs klašu (pozitīvs, neitrāls vai negatīvs)

[5] Pieejams https://github.com/FnTm/latvian-tweet-sentiment-corpus
[6] Pieejams https://github.com/nicemanis/LV-twitter-sentiment-corpus
[7] Pieejams http://www.loc.gov/standards/iso639-2/php/code_list.php

sentimenta klasifikācija. Papildus izveidotajam korpusam, tika veikta arī Sentiment140 datu apstrāde, lai tos pielāgotu darba gaitā izstrādātajai mākslīgajam neironu tīklam.

4.3. Vārdu vektoru reprezentācija

Lai varētu sekmīgi veikt sentimenta analīzes uzdevumu, nepietiek tikai ar labas treniņu datu kopas iegūšanu. Tā pat kā daudzos citos NLP uzdevumos, ļoti būtisks, rezultātus ietekmējošs, faktors ir laba valodas modeļa izmantošana, ar kura palīdzību analizējamais teksts tiek pārveidots izvēlētajai sentimenta analīzes risināšanas metodei piemērotā formātā. Viens no populāriem valodas modeļa veidiem ir izmantot n-grammu modeļus [13, 29, 30], taču tie savā uzbūvē ir vienkārši, un, ņemot vērā pēdējo gadu progresu mašīnmācīšanās nozarē, uz neironu tīkliem bāzētie valodu modeļi sniedz daudz labākus rezultātus. [31]

Ļoti izplatīta šāda veida modeļu grupa ir Google pētnieku komandas radītais word2vec. [32] Ar šo modeļu palīdzību vārdi tiek pārveidoti liela dimensiju skaita (parasti ap 300) vektoros, kas satur reālus skaitļus. Turklāt, šie vārdu vektori ir izkārtoti tā, lai starp līdzīgiem un saistītiem vārdiem veidotos sakarības (skat. 4.3. att.). Tā rezultātā, izmantojot šāda veida vārda vektoru reprezentācijas, ir iespējamas veikt matemātiskas darbības, lai iegūtu attiecīgus saistītos vārdus. Piemēram, izteiksmes *vec("Madride") – vec("Spānija") + vec("Francija")* rezultātā iegūtais vektors atradīsies vistuvāk *vec("Parīze")* vektoram. Šīs sakarības tiek iegūtas trenējot word2vec modeli uz ļoti liela apjoma teksta korpusa, piemēram, dažādiem ziņu rakstiem, kā rezultātā modelis iemācās valodā izmantoto vārdu savstarpējās sakarības.

Šī darba ietvaros ir izmantoti sekojošie word2vec modeļi:

- Google atvērtā pirmkoda projektā izveidotais vārda līmeņa word2vec modelis[8];
- Artūra Znotiņa veidotais latviešu valodas word2vec modelis [33]

Modeļi tika izmantoti datu priekšapstrādes procesā, vārdu vektoru reprezentāciju iegūšanai.

4.4. Emocijzīmju vektoru reprezentācija

Līdzīgi, kā tas tika darīts ar vārdiem, arī emocijzīmes ir nepieciešams pārveidot vektoru formātā, lai tās varētu tik padotas mākslīgajam neironu tīklam. Ņemot vērā to, ka šajā darbā izmantotajās vārdu vektoru vārdnīcās ir pieejami tulkojumi tikai dažām emocijzīmēm,

[8] Pieejams https://code.google.com/archive/p/word2vec/

nepieciešams izveidot metodi, ar kuras palīdzību tiks veikta emocijzīmju pārveidošana vektoru formātā.

Darba izstrādes gaitā, emocijzīmju pārveidošanai, tika izveidota emocijzīmju vārdnīca, ar kuras palīdzību emocijzīme tiek pārvēsta trīs dimensiju vektora formā, atbilstoši tās sentimenta klasei. Emocijzīmes zīmes vektora dimensiju skaits atbilst trīs paredzēto sentimenta klašu (pozitīvā, neitrālā un negatīvā) skaitam, kur vektora pirmā dimensija atbilst negatīvajai klasei, vektora otrā dimensija atbilst neitrālajai klasei un vektora trešā dimensija atbilst pozitīvajai klasei (skat. 4.3. att).

Emocijzīmes ikona	Sentimenta klase	Vektora reprezentācija
😭	Negatīvs	[1, 0, 0]
☯	Neitrāls	[0, 1, 0]
😂	Pozitīvs	[0, 0, 1]

4.3. att. **Emocijzīmju un to attiecīgo vektoru piemērs**

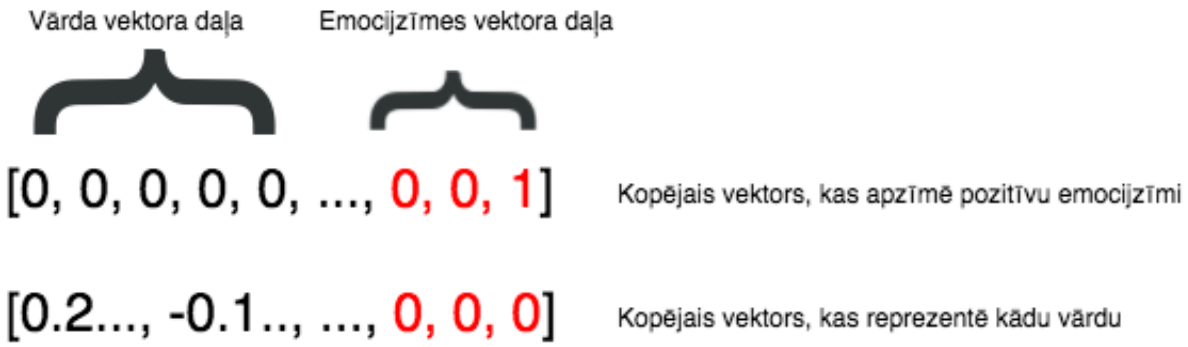
Kopumā emocijzīmju vārdnīca sastāv no 1098 emocijzīmēm, kuras tika iegūtas no sekojošajiem leksikoniem:

- Petra Karlj Novak un citu [34] veidotais emocijzīmju leksikons, kas sastāv no 751 viena simbola emocijzīmes. Šo emociju sentimenta klasifikācija tika veikta automatizētā veidā. Sentimenta novērtējums tika balstīts uz 70 000 ar rokām anotētiem tvītiem, kurus anotēja 83 cilvēki 13 dažādās Eiropas valodās;
- Wikipedia pieejamais emocijzīmju saraksts[9]. No šī saraksta tika ņemtas tikai pirmās sadaļas 347 emocijzīmes, kas ir uz sāniem sagrieztās latīņu simbolu emocijzīmes. Šo emocijzīmju sentimenta klasifikācija tika veikta manuāli, atbilstoši pievienotajiem emocijzīmju skaidrojumiem.

Ņemot vērā to, ka nepieciešams izmantot gan vārdu vektorus, gan emocijzīmju vektorus, nepieciešams izveidot abu vektoru apvienojumu, lai tos varētu padot mākslīgajam neironu tīklam. Apvienojot abus vektorus, emocijzīmes tulkošanas rezultātā iegūtais emocijzīmes vektors tiek konkatenēts ar nullēm aizpildītam vektoram, kura dimensiju skaits atbilst vārda vektora dimensiju skaitam. Tas tiek darīts ar pieņēmumu, ka izmantotajā vārda vektoru modelī

[9] Pieejams https://en.wikipedia.org/wiki/List_of_emoticons

šāds vektors apzīmē tukšu jeb nekādu vārdu. Attiecīgi, arī vārda vektoriem galā tiek konkatenēts ar nullēm aizpildīts vektors, kas atbilst emocijzīmju vektora dimensiju skaitam. Tā rezultātā tiek iegūts apvienotais vektors, ar kura palīdzību var reprezentēt gan vārdu, gan emocijzīmi, jo gadījumā, ja tiek pārveidots vārds, vektora emocijzīmes daļā būs nulles, kas apzīmēs, ka dotais vektors nereprezentē emocijzīmi, bet, ja tiek pārveidota emocijzīme, vektora vārda daļā būs nulles, kas apzīmēs, ka dotais vektors nereprezentē vārdu (skat. 4.4. att.).



4.4. att. **Vārda un emocijzīmes vektoru apvienojuma piemērs**

4.5. Datu analīze

Šīs apakšnodaļas ietvaros ir aprakstīti dažādi novērojumi saistībā ar iepriekšējā nodaļā aprakstītajiem datiem. Twitter Search API dati tiks testēti ar tipiskiem sistēmas lietošanas scenārijā paredzētiem vaicājumiem. Vaicājumi tiks veikti gan angļu, gan latviešu valodā, izmantojot sekojošos atslēgas vārdus:

- Angļu valodā – {“Donald Trump”, “Iran nuclear”, “Russia”, “Turkey Coup”, “Global warming”};
- Latviešu valodā – {“vēlēšanas”, “hokejs”, “eirovīzija”, “Krievija”, “muzeju nakts”}.

4.5.1. Twitter Search API

Veicot atkārtotu vaicājumu veikšanu (skat. 4.3. tabulu), tika novērots, ka ir reti sastopami tvīti, kuriem ir pievienota precīza ģeolokācijas informācija, ko nodrošina Twitter gadījumā, ja ir lietotājs ir iespējojis ģeolokāciju, taču vairumā gadījumos tvīta autora profila informācijā tika norādīta lietotāja lokācija, ko lietotājs ievadījis ar roku. Tā nav tik precīza kā Twitter pievienotā ģeolokācija, bet ar ģelokācijas servisu palīdzību ir iespējams noteikt ievadītas lokācijas informācijas atbilstošās koordinātas. No iegūtajiem rezultātiem var secināt, ka aptuveni 60-80%

no vaicājumā atgrieztajiem tvītiem satur vismaz lietotāja ievadītu lokācijas informāciju, kas attiecīgi ļauj veikt sabiedrības attieksmes modelēšanu atkarībā no atrašanās vietas valsts robežu mērogā. Tomēr, lai varētu pilnībā redzēt sabiedrības attieksmes izmaiņas globālā līmenī, nepieciešams veikt vaicājumus visās valodās, jo, piemēram, veicot vaicājums latviešu valodā, iegūtie rezultāti vairumā gadījumu būs nākuši no Latvijas. Kā alternatīvu var izmantot tikai angļu valodu, kas ir samērā internacionāla valoda, taču arī tas nenodrošina vienmērīgu rezultātu lokācijas sadalījumu.

Valoda	**Tvītu daudzums ar Twitter ģeolokāciju**	**Tvītu daudzums ar lietotāja ievadītu lokāciju**	**Kopējais tvītu daudzums ar lokāciju**
Angļu	2.64%	59.2%	59.8%
Latviešu	4.6%	71.6%	72.6%

4.3. tabula **Ar lokācijas informāciju pieejamo tvītu daudzums**[10]

Vēl viens sistēmas datu pilnību ierobežojošs faktors ir datu skaita ierobežojums vienā pieprasījumā. Izmantojot Twitter Search API, vienā pieprasījumā iespējams iegūt maksimums tikai 100 tvītus, kas ir nepietiekošs viedokļu skaits, lai objektīvi atspoguļotu sabiedrības kopējo viedokli. Tomēr, ņemot vērā, ka pieejami tikai pēdējo 7 dienu dati, veicot eksperimentus ar latviešu valodas vaicājumiem tika novērots, ka ne vienmēr izdodas sasniegt tvītu skaita ierobežojumu, jo dotajā laika periodā nav bijuši tik daudz tvītu saistībā ar meklētajiem atslēgas vārdiem. Lai apstrādātu situāciju, kad pieejami vairāk nekā 100 tvīti, var veikt vairāku pieprasījumu veikšanu, kuros tvīti tiek sakārtoti pēc to identifikatora, un, veicot papildus pieprasījumus, uzstādīt papildus vaicājuma operatoru, kas atlasa tikai tos tvītus, kuru identifikatori ir lielāki vai mazāki par attiecīgi lielāko vai mazāko iepriekšējā pieprasījumā saņemto tvītu identifikatoru. Šādu papildus pieprasījumu veikšanu var atkārtot līdz atgriezto tvītu skaits ir mazāks par 100, kas nozīmē, ka attiecīgajā virzienā ir atlasīti visi pieejamie tvīti.

4.6. Datu priekšapstrāde

Kā jau tas tika aprakstīts 1.3. apakšnodaļā, ņemot vērā Twitter noteikto simbolu skaita limitu, kā arī sociālajiem tīkliem raksturīgo neformālo gaisotni, tvītos esošā valoda bieži vien ir neformāla un var saturēt dažādus saīsinājums, slengu, svešvārdus, emocijzīmes u.c. valodas izteiksmes līdzekļus. Līdz ar to, lai iegūtos datus varētu sekmīgi izmantot sentimenta analīzei,

[10] Ar lokācijas informāciju pieejamo tvītu daudzums procentos tika rēķināts balstoties uz attiecīgajā pieprasījumā iegūto kopējo tvītu skaitu

tos nepieciešams apstrādāt pirms padošanas mākslīgajam neironu tīklam. Šim nolūkam darbam uzstādītās problēmas risināšanas gaitā tika izveidots atsevišķs modulis, kas paredzēts datu ielādei un priekšapstrādei, kura detaļas ir aprakstītas šajā apakšnodaļā.

Datu priekšapstrādes process tiek uzsākts ar nepieciešamo resursu ielādi.

- Vārdu vektoru modeļa ielāde – vārdu vektoru modeļi visbiežāk tiek glabāti binārā formātā un to ielādei izmanto tam paredzētu bibliotēku. Šī darba ietvaros vārda vektoru modeļu ielādei tiek izmantota Gensim[11] bibliotēka;
- Emocijzīmju leksikona ielāde – darba ietvaros izstrādātā emocijzīmju vārdnīca tiek glabāta CSV formāta failā. Attiecīgi tās ielādei tiek izmantota tam paredzētā Python standarta bibliotēku kopā esošā bibliotēka. Emocijzīmju vārdnīcas fails ir strukturēts pa rindiņām, kur katrā rindiņā ir divas vērtības, kuras atdalītas ar komatu. Pirmā vērtība ir emocijzīme, bet otrā vērtība ir attiecīgās emocijzīmes vektors;
- Tvītu korpusa ielāde – darba gaitā izstrādātā mākslīgā neironu tīkla apmācīšanai paredzētie dati tika glabāti JSON formāta failos. Šāds datu failu formāts tika izvēlēts, lai datus varētu pārveidot cilvēkiem ērti lasāmā formātā.

Pēc visu nepieciešamo resursu ielādes tika veikta datu priekšapstrāde. Tā tika veikta iterējot cauri tvītu korpusam. Katrs tvīts tika sākumā sadalīts vārdos, pieņemot, ka vārdi tiek atdalīti ar atstarpēm vai ‘\n’ un ‘\t’ simboliem. Pēc tam tika veikta vārdu pārveidošana vektoru formātā ar iepriekš minēto vārda vektoru modeļu palīdzību. Ja vārds netika atpazīts, tad tika veikta vārda attīrīšana. Ja tika konstatēts, ka konkrētais vārds sākas ar ‘@’ vai ar ‘http’, tad tas tika izmests. Ja tika konstatēts, ka konkrētais vārds sākas ar ‘#’, tad tas tika apstrādāts ar vārdu atdalīšanai paredzētu autora implementētu funkciju. Šī funkcija veica ciklisku iterēšanu pār konkrēto simbolu virkni sākot no simbolu virknes beigām. Katrā iterācijā tika ņemta aiz vien lielāka vārda daļa no beigām. Tik līdz vārdu vektoru modelis atpazina kādu daļu, tā tika pievienota jau pārveidotajiem tvīta vārdiem. Ja iegūtais vārds neatbilda nevienam no iepriekš minētajiem kritērijiem, tad tika veikta simbolu virknes atfiltrēšana no simboliem, kas nav burti vai cipari. Pēc simbolu virknes atfiltrēšanas tika veikts atkārtots mēģinājums vārdu pārveidot vektora formātā. Ja arī tad konkrētais vārds netika atpazīts, tad tas tika atmests. Pārveidotie vārdu vektori tika ielikti fiksēta izmēra vektorā, kas reprezentēja visu tvītu pārveidotu vektoru formātā. Šī vektora garums apzīmēja maksimāli pieļaujamo vārdu skaitu, un pēc darbā izmantoto tvītu korpusu izpētes, šim vektoram tika fiksēts 50 dimensiju garums. Tvītiem,

[11] Pieejams https://radimrehurek.com/gensim/

kuriem bija mazāks vārdu skaits, tika veikts to papildinājums ar 0 aizpildītiem vektoriem, līdz tik sasniegts 50 vārdu skaits.

4.7. Datu vizualizēšana

Tīkla veiktspējas rādītāju vizualizēšanai tika izmantots TensorBoard[12] vizualizēšanas rīks. Vizualizēšanai paredzētie dati tika iegūti tīkla trenēšanas un testēšanas procesā ar kopsavilkumu operāciju palīdzību. Tīkla rezultātu pārpratumu matricas vizualizēšanai tika izmantota matplotlib[13] bibliotēka.

[12] Pieejams https://www.tensorflow.org/get_started/summaries_and_tensorboard
[13] Pieejams https://matplotlib.org/

5. PROBLĒMAS RISINĀJUMS

Šajā nodaļā ir aprakstīts bakalaura darbam uzstādītās problēmas risinājums un risinājumā izmantotās tehnoloģijas.

5.1. Mākslīgā neironu tīkla implementēšana

Bakalaura darbam uzstādītās problēmas risināšanai ir izmantota iepriekš aprakstītā ilgās īstermiņa atmiņas rekurentā mākslīgo neironu tīklu struktūra, kuras realizējuma detaļas ir sīkāk aprakstītas šajā apakšnodaļā. Mākslīgais neironu tīkls tika implementēts Python programmēšanas valodā ar TenosrFlow[14] bibliotēkas palīdzību.

```
96  def main(n_hidden, learning_rate):
97      tf.reset_default_graph()    # Tiek veikta grafa atiestatīšana, gadījumā, ja tiek veikta atkārtota trenēšana
98      # Tīkla parametri
99      n_input = dp.v_size         # Vārdu un emocijzīmju apvienotā vektora dimensiju skaits
100     n_steps = dp.max_twt_len    # Maksimālais vārdu skaits - fiksēts uz 50
101     n_classes = 3               # Sentimenta klašu skaits [1, 0, 0], [0, 1, 0], [0, 0, 1]
102     gpu_mem_limit = 0.6         # Limitē cik % no GPU atmiņas drīkst izmantot
103     # Ievades slāņa definīcija
104     with tf.name_scope('input_layer'):
105         x = tf.placeholder(tf.float32, [None, n_steps, n_input], name='x')
106     # Pareizo izvaddatu viettura definīcija
107     with tf.name_scope('correct_labels'):
108         y = tf.placeholder(tf.float32, [None, n_classes], name='labels')
109     # Slēptā slāņa definīcija
110     with tf.name_scope('hidden_layer'):
111         lstm_cell = tf.contrib.rnn.LSTMCell(n_hidden)
112         output, state = tf.nn.dynamic_rnn(lstm_cell, x, dtype=tf.float32)
113     # Izvades slāņa definīcija
114     with tf.name_scope('output_layer'):
115         outputs = tf.transpose(output, perm=[1, 0, 2])
116         last = tf.gather(outputs, int(outputs.get_shape()[0]) - 1)
117         W = tf.Variable(tf.truncated_normal([n_hidden, int(y.get_shape()[1])]))
118         b = tf.Variable(tf.constant(0.1, shape=[y.get_shape()[1]]))
119         variable_summaries(W, 'weight_summary')
120         variable_summaries(b, 'bias_summary')
121         logits = tf.nn.xw_plus_b(last, W, b)
122         prediction = tf.nn.softmax(logits)
```

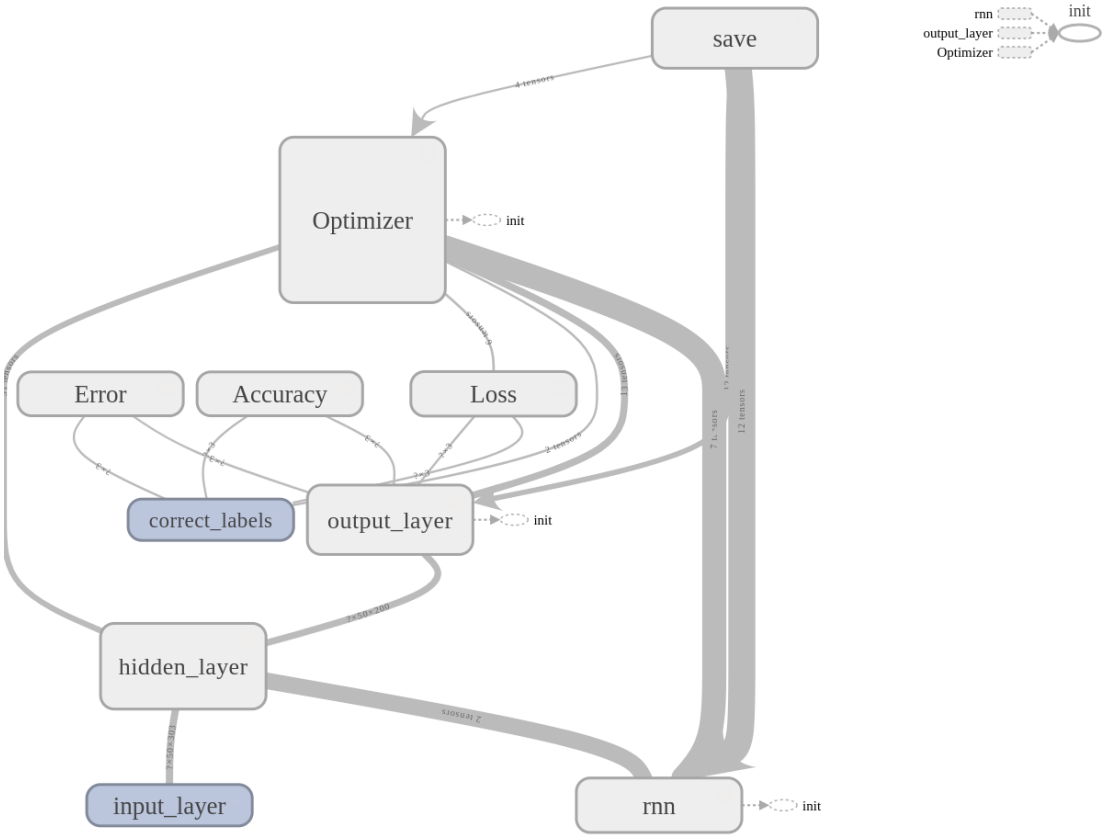
5.1. att. **Modeļa izveides pirmkoda paraugs**

Kā redzams tīkla modeļa izveides pirmkoda paraugā (skat. 5.1. att.), tīkla modeļa definīcijas sākumā tiek izveidoti divi vietturu veida tenzori – ieejas datu vietturis un pareizo izejas datu vietturis. Ieejas datu vietturis kalpos kā tīkla modeļa ievades slānis, bet pareizo izejas datu vietturis tiks izmantots pareizo izejas datu padošanai kļūdas un precizitātes funkcijas definīcijā. Attiecīgi ieejas datu viettura tenzoram tiek definētā forma, kura atbilst treniņa ieejas datu struktūrai, un izejas datu viettura tenzoram tiek definēta forma, kas atbilst treniņu izejas datu struktūrai. Pēc ieejas un izejas datu vietturu definēšanas, tiek veikta slēptā LSTM slāņa definēšana. Šī slāņa definīcijas pamatā ir LSTM šūna, kura tiek inicializēta ar slēptā slāņa

[14] Pieejams https://www.tensorflow.org/

paredzētajam izmēram atbilstošu vienību skaitu. Pēc LSTM šūnas definēšanas, tiek izveidots slēptais slānis, kura konstruktorā tiek padota iepriekš definētā LSTM šūna un ieejas datu vietturis. LSTM slāņa definēšanā izmantotā metode nodrošina dinamisku ievaddatu tenzora attīšanu, kas ļauj izmantot vienu ievaddatu tenzoru nevis Python sarakstu ar tenzoriem, atbilstoši slēptā slāņa izmēram. Pēc slēptā slāņa definēšanas nepieciešams definēt modeļa izejas slāni. Izejas slānis tiek izveidots transponējot slēptā LSTM slāņa izvadu, kas nepieciešams, lai iegūtu tikai pēdējo vērtību, kas saturēs visu kopējo LSTM slāņa informāciju. Papildus iegūtajai slēptā slāņa vērtībai, tiek definēti divi mainīgo veida tenzori. Vienā no tiem tiks glabāti izejas slānim paredzētie svari, bet otrā tiks glabāti izejas slānim paredzētās noslieces. Pēc nepieciešamo tenzoru iegūšanas, tiek veikta izejas slāņa izveidošana – slēptā slāņa vērtības tenzora un svaru tenzora matricas reizinājumam pieskata klāt noslieču tenzoru un iegūto rezultātu padod softmax funkcijai, kas veic izejas vektora vērtību reducēšanu, tā rezultātā iegūstot vektoru, kura vērtību summa ir 1.

Kā redzams TensorBoard veidotajā mākslīgo neironu tīkla struktūras vizualizācijā (skat. 5.2. att.), iepriekš aprakstītā modeļa definīcija atbilst paredzētajai modeļa struktūrai.



5.2. att. **Mākslīgā neironu tīkla struktūras vizualizācija**

5.2. Apmācības process

Lai varētu veikt izveidotā modeļa apmācību, nepieciešams definēt modeļa kļūdas noteikšanai un modeļa parametru optimizēšanai nepieciešamās funkcijas.

```
123        # Kļūdas funkcijas definīcija
124        with tf.name_scope('Loss'):
125            cross_entropy = -tf.reduce_sum(y * tf.log(tf.clip_by_value(prediction, 1e-10, 1.0)))
126        # Tīkla trenēšanas funkcijas definīcija
127        with tf.name_scope('Optimizer'):
128            optimizer = tf.train.AdamOptimizer(learning_rate)
129            minimize = optimizer.minimize(cross_entropy)
130        # Kļūdas intensitātes noteikšanas funkcijas definīcija
131        with tf.name_scope('Error'):
132            mistakes = tf.not_equal(tf.argmax(y, 1), tf.argmax(prediction, 1))
133            error = tf.reduce_mean(tf.cast(mistakes, tf.float32))
134        # Precizitātes noteikšanas funckijas definīcija
135        with tf.name_scope('Accuracy'):
136            correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(prediction, 1))
137            accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

5.3. att. **Modeļa kļūdas noteikšanas un parametru optimizēšanas pirmkoda paraugs**

Kā redzams pirmkoda paraugā (skat. 5.3. att.), modeļa kļūdas noteikšanai tiek izmantota krosentropijas funkcija, kura tika implementēta, balstoties uz daļu no darba praktiskajā daļā apskatītās literatūras [35, 36, 37]. Tālāk, aprēķinātā kļūdas funkcija tiek padota TensorFlow implementētajam Adam optimizatoram, kas atbilstoši uzstādītajai mācīšanās intensitātei veic tīkla parametru koriģēšanu, par pamatu ņemot optimizatoram padoto kļūdas funkciju. Ar šo abu funkciju definīcijām jau ir iespējams veikt izveidotā tīkla trenēšanu. Lai vizualizētu izveidotā tīkla apmācības procesu, tiek definētas arī kļūdas intensitātes un precizitātes funkcijas. To darbības pamatā ir argmax funkcija, kas nosaka padotā vektora augstākās vērtības indeksu, kas attiecīgi ļauj salīdzināt vai tīkla izvadē iegūtais rezultāts sakrīt ar pareizo izejas rezultātu. Pēc tam, tīkla trenēšanas procesā, ar šo funkciju palīdzību tiks veikta tīkla precizitātes un kļūdas intensitātes novērtēšana, kas tiks vizualizēta iekš TensorBoard kopsavilkuma.

Pēc mākslīgā neironu tīkla un visu vajadzīgo funkciju definēšanas, nākošais solis ir tīkla apmācīšana uz nepieciešamajiem treniņa datiem. Tīkla apmācības process tiek veikts ar trīs ciklu palīdzību. Apmācības procesa pamatā ir cikls, kurš iterē atbilstoši izvēlēto epohu skaitam. Epohu cikla iekšienē ir iekļauts otrais cikls, kurš iterē atbilstoši izmantoto treniņam paredzēto datu failu skaitam. Datu failu cikla iekšienē ir iekļauts trešais cikls, kurš iterē atbilstoši ielādētajā failā esošo treniņa datu skaitam. Datu failu ciklā esošā funkcionalitāte tiek nedaudz izmainīta atkarībā no treniņam paredzētu datu failu skaita. Ja treniņam tiek izmantots tikai viens fails, tad attiecīgais fails tiek sadalīts treniņa un testa datu kopā, kur treniņam ir paredzēti 80% no datu kopas un testam ir paredzēti 20% no datu kopas, un iterējot cauri izvēlētajam epohu skaitam, ielādētais datu fails netiek pārlādēts. Taču, ja treniņam tiek izmantoti vairāki datu faili,

tad tie tiek katru reizi pārlādēti datu faila cikla sākumā, un šajā gadījumā ielādētie faili netiek dalīti treniņa un testa datu kopās, bet tam nolūkam tiek izmantots atsevišķs testa datu fails, kuru ielādē epohu cikla beigās, lai novērtētu trenēšanas rezultātus. Testa datu fails var tikt izmantots arī, ja tiek izmantots viens treniņu datu fails, bet tādā gadījumā testa datu fails tiek ielādēts tikai apmācības procesa beigās, lai netiktu veikta lieka datu pārlāde apmācības procesa laikā. Šāda apmācības procesa struktūra tika izmantota, lai varētu veikt divu dažādu veidu apmācības procesus:

- Ātrā apmācība – šis apmācības procesa veids balstās uz to, ka tiek izmantots tikai viens datu fails un treniņa dati apmācības procesa laikā netiek pārlādēti. Attiecīgi tas nodrošina salīdzinoši ātrāku tīkla uztrenēšanu, jo starp treniņu cikla iterācijām netiek tērēts laiks datu ielādei un priekšapstrādei. Šīs metodes trūkums ir tas, ka tiek limitēts apmācībai izmantojamo treniņa datu skaits atbilstoši pieejamai operatīvajai atmiņai. Darba praktiskās daļas izstrādes procesā tika novērots, ka ar 16GB operatīvo atmiņu var izmantot aptuveni līdz 100 000 tvītu lielu datu kopu. Šī apmācības veids ir paredzēts, lai trenētu nelielas datu kopas, kā arī, lai veiktu tīkla optimizāciju;
- Lēnā apmācība – šis apmācības procesa veids balstās uz to, ka tiek izmantoti vairāki datu faili. Izmantojot šo apmācības veidu tīkla trenēšana noritēs daudz lēnāk, taču šajā gadījumā netiek limitēts treniņa datu kopas izmērs. Šis apmācības veids paredzēts, lai veiktu tīkla apmācību uz lielām datu kopām, piemēram, iepriekš minēto Sentiment140 1,6 miljoni lielo tvītu korpusu.

5.2.1. Tīkla apmācīšana, izmantojot Sentiment140 datu kopu

Ņemot vērā Sentiment140 Twitter sentimenta korpusa lielo apjomu, tas tika sadalīts 16 sīkākos korpusos, kur katrs no tiem satur 100 000 tvītus. Tīkla trenēšanai tika izmantoti 1,5 miljoni tvītu, bet testēšanai 100 000 tvītu. Attiecīgi tīkla apmācības procesam tika izvēlēts lēnais apmācības veids.

Viens epohas cikls sastāv no 15 iterāciju cikla, kur katrā iterācijā tiek ielādēta un apstrādāta attiecīgā korpusa daļa. Pēc datu ielādes un apstrādes, tiek uzsākts tīkla trenēšanas cikls, kurā tiek veikta tīkla trenēšana ar sagatavotās korpusa daļas datiem. Lai optimizētu resursu izmantošanu, attiecīgā faila dati tīkla trenēšanas operācijai tiek padoti 1000 tvītu lielās partijās, kuras pēc kārtas katrā cikla iterācijā tiek padotas vienlaicīgi. Papildus trenēšanas operācijas izpildei, katrā tīkla trenēšanas cikla iterācijā tiek izpildīta arī apvienotā kopsavilkuma operācija, kas veic tīkla rādītāju apkopošanu un pievienošanu TensorBoard

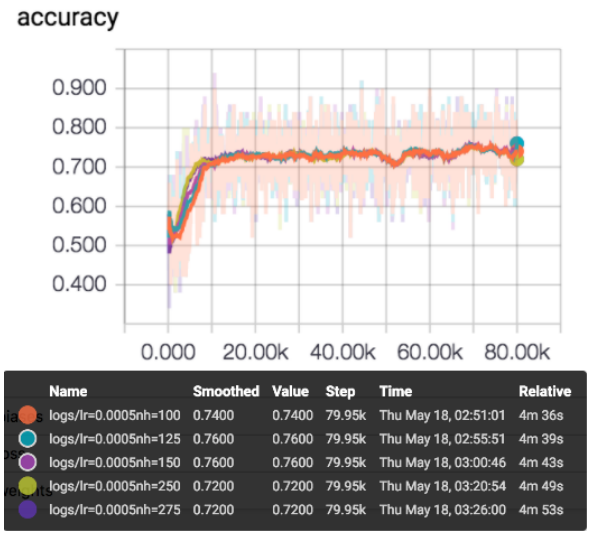
grafikiem. Pēc viena epohas cikla veikšanas, tiek veikta tīkla veiktspējas rādītāju pārbaude ar testēšanai paredzētās korpusa daļas palīdzību. Pēc visu epohu veikšanas, tiek aprēķinātas un izdrukātas 2.1. apakšnodaļā aprakstītās metrikas.

5.2.2. Tīkla apmācīšana izmantojot latviešu valodas tvītu korpusu

Apvienojot darba autora izveidoto latviešu valodas tvītu korpusu ar Jāņa Peisenieka veidoto tvītu korpusu, rezultātā tika iegūts 4308 tvītu liels korpuss. Veicot tīkla apmācību uz iegūtā korpusa datiem, tas tika sadalīts treniņa un testa daļā, atbilstoši iepriekš aprakstītajam procesam.

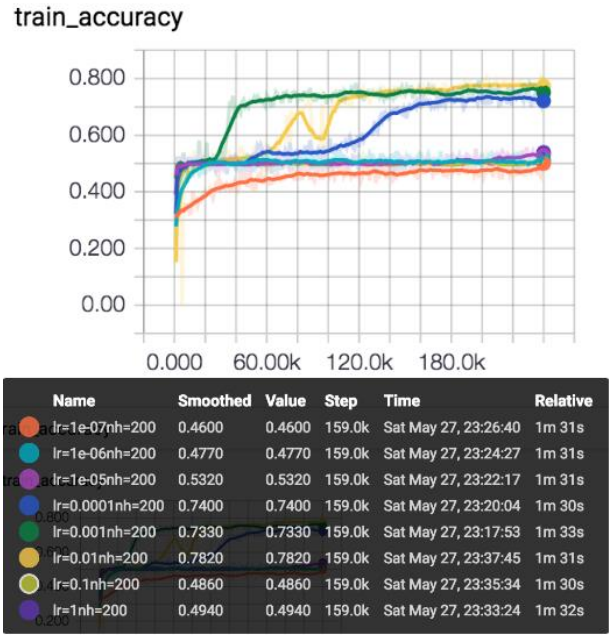
5.3. Tīkla optimizācija

Šajā apakšnodaļā ir aprakstīti dažādi eksperimenti, ar kuru palīdzību tika optimizēti tīkla parametri. Veicot mākslīgā neironu tīkla optimizāciju, pamatā tika koriģēta tikai mācīšanās intensitāte, jo kā tas tika novērots eksperimentu rezultātā, slēptā slāņa izmēra mainīšana neveica būtiskas izmaiņas tīkla precizitātē (skat. 5.4. att.). Grafika apzīmējumos saīsinājums *lr* apzīmē mācīšanās intensitāti un saīsinājums *nh* apzīmē slēptā slāņa izmēru.



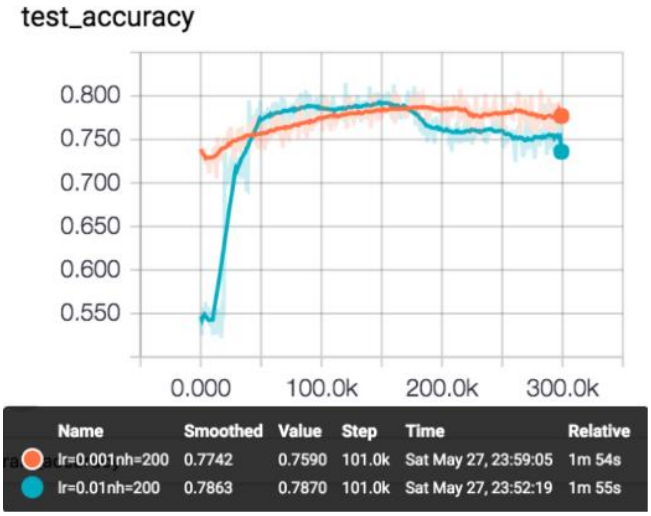
5.4. att. **Tīkla precizitātes izmaiņa atkarībā no slēptā slāņa izmēra**

Tīkla mācīšanās intensitātes optimizācija tika sadalīta vairākās daļās. Sākumā tika testēts liels mācīšanās intensitāšu vērtību intervāls uz nelielu epohu skaitu, lai noteiktu aptuveno optimālo mācīšanās intensitātes reģionu (skat. 5.5. att.).



5.5. att. **Tīkla treniņa precizitātes maiņa atkarībā no mācīšanās intensitātes**

Eksperimenta rezultātā tika secināts, ka tīkls uzrādīja vislabākos rezultātus, ja mācīšanās intensitāte bija 0.01, 0.001 vai 0.0001. Attiecīgi tālāk tika veikta tālāka tīkla rezultātu izpēte, izmantojot divas labākās mācīšanās intensitātes vērtības.

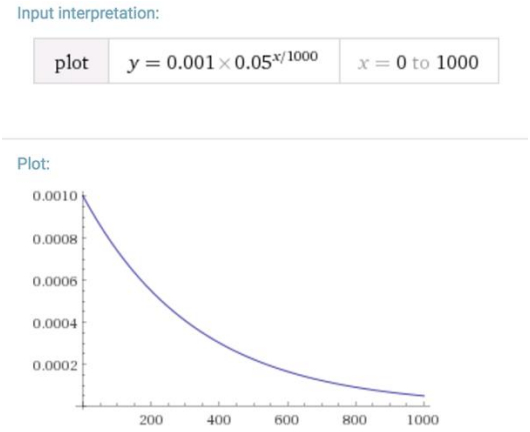


5.6. att. **Tīkla testa precizitātes maiņa atkarībā no mācīšanās intensitātes**

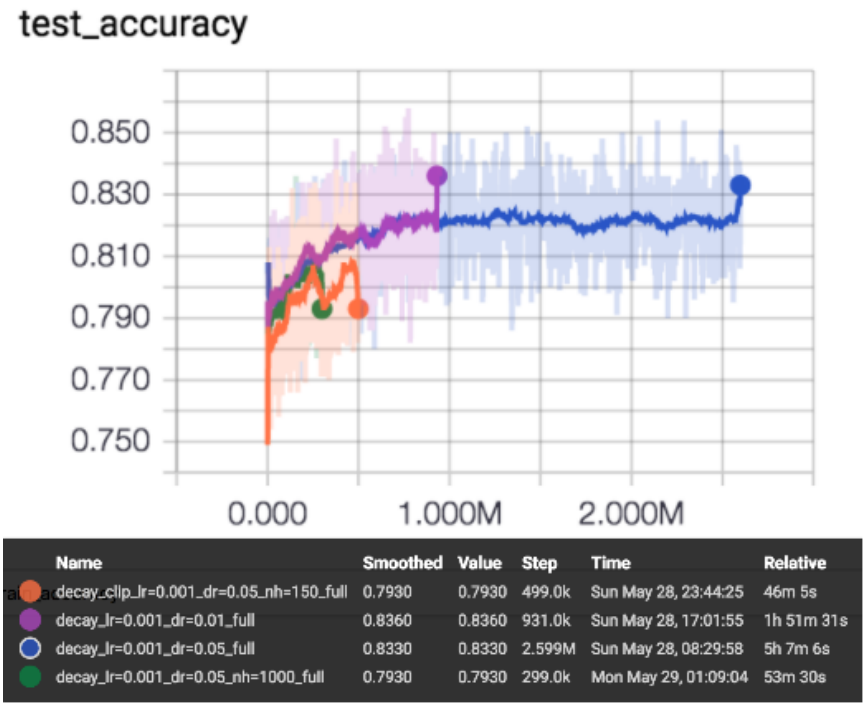
Kā redzams precīzākā mācīšanās intensitātes izpētē (skat. 5.6. att.), tīkls abos gadījumos kopumā sasniedz gandrīz vienādu precizitāti (0.01 sasniedz nedaudz augstāku precizitāti), taču lielākas mācīšanās intensitātes gadījumā konverģēšana notiek daudz ātrāk. Abos gadījumos novērojas, ka pēc noteikta epohu skaita tīkla precizitāte samazinās, kas nozīmē, ka nepieciešams samazināt mācīšanās intensitāti. Tā rezultātā tika nolemts izmantot eksponenciāli dilstošu mācīšanās intensitāti. TensoFlow bibliotēkas piedāvātā eksponenciāli dilstošā mācīšanās intensitāte tiek aprēķināta šādi:

$$y = lr_{init} \times dr^{\frac{step_{curret}}{step_{total}}}$$

Ar lr_{init} apzīmē sākotnējo mācīšanās intensitāti, ar dr apzīmē dilšanas intensitāti un ar $step$ apzīmē treniņa iterāciju skaitītāju. 5.7. attēlā redzams piemērs eksponenciāli dilstošas mācīšanās intensitātes vērtības izmaiņai.



5.7. att. **Eksponenciāli dilstoša mācīšanās intensitātes vērtības izmaiņa atkarībā no treniņa iterāciju skaitītāja**



5.8. att. **Tīkla testa precizitātes izmaiņa atkarībā no eksponenciāli dilstošas mācīšanās intensitātes**

Kā redzams eksponenciāli dilstošas mācīšanās intensitātes eksperimenta paraugā (skat. 5.8. att.), mācīšanās intensitātes pakāpeniska samazināšana dod nelielu tīkla precizitātes uzlabojumu.

5.4. Risinājuma pielietošana

Kā tas tika minēts darba ievadā, izveidoto sentimenta analīzes risinājumu tālāk paredzēs izmantot informācijas izgūšanas sistēmas koncepta izveidē. Attiecīgi šajā apakšnodaļā ir aprakstīta sistēmas koncepta realizācija.

Informācijas ieguves sistēmas koncepts tika realizēts ar NodeJS[15] un ExpressJS[16] bibliotēku palīdzību. Tas tika veidots kā tīmeklī bāzēta lietotne, kura tika mitināta mākoņpakalpojumu platformā DigitalOcean[17]. Lai lietotni padarītu publiski pieejamu un veiktu komunikāciju ar Twitter API, tika piereģistrēts "opinion-mine.com" domēna nosaukums, kā arī tika izveidots SSL sertifikāts ar Let's Encrpyt palīdzību[18].

Izveidotais sistēmas koncepts darba izstrādes gaitā tika izmantots, lai veiktu Twitter API datu iegūšanu un testēšanu. Ņemot vērā mākslīgo neironu tīklu darbināšanai nepieciešamos resursus, darbā izstrādātais mākslīgais neironu tīkls netika sasaistīts ar izstrādāto sistēmas konceptu.

[15] Pieejams https://nodejs.org/en/
[16] Pieejams https://expressjs.com/
[17] Pieejams https://www.digitalocean.com/
[18] Pieejams https://letsencrypt.org/

REZULTĀTI

Lai sasniegtu darbam uzstādīto mērķi, darba izstrādes gaitā tika iegūts 3131 tvītu liels latviešu valodas Twitter ziņu sentimenta korpuss. 1085 jeb 34,7% no ziņām tika klasificētas kā pozitīvas, 1712 jeb 54,7% kā neitrālas un 334 jeb 10,7% kā negatīvas. Pēc iegūto datu priekšapstrādes tikai veikta apstrādāto datu atpazīstamības pārbaude attiecībā pret darbā izmantoto Artūra Znotiņa izveidoto latviešu valodas vārdu vektoru modeli. Pēc datu atpazīšanas pārbaudes tika noskaidrots, ka 336 340 jeb 94,45% no apstrādātajiem vārdiem tika atpazīti un tikai 1976 jeb 5.55% netika atpazīti. Visbiežāk sastopamie atpazītie vārdi bija – "un", "ir", "ka", "es", "ar". Neatpazīto vārdu kopā nebija izteikti bieži sastopamu vārdu.

Darba galvenajā daļā izveidotais ilgās īstermiņa atmiņas rekurentais neironu tīkls pēc tā trenēšanas uz 1,5 miljoniem Sentiment140 korpusa tvītiem ieguva 82,17% procentu precizitāti, kas tika aprēķināta testējot tīkla darbību uz atlikušajiem 100 000 Sentiment140 korpusa tvītiem. Šī precizitāte apzīmē to cik gadījumos tīkla izdotais rezultāts sakrita ar testa datu rezultātu. Aprēķinot tīkla rezultātu precīzākas metrikas, tika noskaidrots, ka tīkls sasniedza 82,37% precizitāti, 81,86% pārklājumu un 81,86% F_1 mēru. Ņemot vērā nelielo latviešu valodas Twitter datu korpusa apjomu, kā arī to, ka korpusā esošie dati ir klasificēti ar 3 klašu sentimentu, izveidotā tīkla apmācības procesā netika iegūti kvalitatīvi rezultāti. Sīkāka informācija par izstrādātā tīkla darbību pieejama pielikumos.

SECINĀJUMI

Darbam uzstādītā mērķa izstrādes procesā tika iegūti vairāki secinājumi saistībā ar darba pētāmo problēmu.

Twitter sociālajā tiklā esošās ziņas ir ļoti vērtīgs informācijas avots. Par to darba autors pārliecinājās darba izstrādes sākuma posmā, kad tika veikta sentimenta analīzes risinājumam nepieciešamo datu vākšana un ievadā aprakstītās informācijas ieguves sistēmas koncepta izveide. Sākotnēji tika paredzēts, ka no Twitter sociālā tīkla ar pieprasījuma palīdzību būs iespējams iegūt vairākus tūkstošus lielu ziņu kopu saistībā ar informācijas ieguves sistēmas lietotāja ievadītajiem atslēgas vārdiem, taču pēc Twitter API izpētēs tika secināts, ka Twitter piedāvā ļoti limitētu ziņu skaitu (maksimums 100 ziņas vienā pieprasījumā), ja tiek lietots bezmaksas Twitter API. Kā alternatīva datu iegūšanai tika piedāvāta komerciāla platforma Gnip, taču sazinoties ar šīs kompānijas pārstāvjiem, tika noskaidrots, ka zemākā līmeņa piedāvājums (1 miljons ziņu 40 dienu periodā) Twitter vēsturisko datu iegūšanai izmaksā 1250 ASV dolārus. Iegūtā informācija apstiprina iepriekš izteikto apgalvojumu par Twitter ziņu vērtību.

Veicot latviešu valodas Twitter ziņu sentimenta korpusa izveidi tika secināts, ka manuāla Twitter ziņu klasificēšana ir lēns un laikietilpīgs process. Ņemot vērā to, ka darbā izveidotā mākslīgā neironu tīkla kvalitatīvu sentimenta analīzes rezultātu iegūšanai nepieciešama salīdzinoši liela treniņa datu kopa (15 000 – 20 000 ziņas), tika secināts, ka šāda veida treniņu datu kopas izveide ir ļoti neefektīva un datu kopas izveides procesu nepieciešams automatizēt.

Neskatoties uz iepriekš aprakstītajām grūtībām, šī darba izstrādes procesā tika sasniegti sekojošie mērķi:

- Sentimenta analīzes risinājums – tika izveidots ilgās īstermiņa atmiņas rekurentais neironu tīkls, kas sasniedza 82,37% precizitāti, 81,86% pārklājumu un 81,86% F_1 mēru. Iegūtie rezultāti bija atbilstoši treniņiem izmantotās datu kopas autoru sasniegtajiem rezultātiem;
- Latviešu valodas Twitter ziņu sentimenta korpuss – darba izstrādes sākuma posmā tika izveidots 3131 ziņu liels manuāli klasificēts ziņu korpuss, kas ir publiski pieejams GitHub repozitorijā tālākiem latviešu valodas sentimenta analīzes pētījumiem;
- Informācijas ieguves sistēmas koncepts – tika izveidota uz NodeJS bāzēta publiksi pieejam tīmekļa lietotne, kas tika izmantota Twitter ziņu iegūšanai un analizēšanai.

Veicot darba gaitā izstrādātā mākslīgā neironu tīkla apmācīšanu un iegūto rezultātu salīdzināšanu, tika secināts, ka ar salīdzinoši vienkāršu mākslīgā neironu tīkla risinājumu var iegūt klasiskajām mašīnmācīšanās metodēm līdzvērtīgu Twitter sentimenta analīzes risinājumu.

Ņemot vērā darba izstrādes gaitā iegūtās atziņas, darbam uzstādītās pētāmās problēmas tālākie iespējamie pētījuma virzieni ir:

- Automatizēti klasificēta latviešu valodas Twitter ziņu korpusa izstrāde;
- Ilgās īstermiņa atmiņas rekurento neironu tīklu optimizēšanas iespēju izpēte;
- Publiski pieejamo sociālo tīklu ziņu automatizētas ieguves sistēmas izveide.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] «Twitter Usage / Company Facts,» Twitter Inc., 30. Jūnijs 2016. [Tiešsaiste]. Available: https://about.twitter.com/company. [Piekļūts 21. Maijs 2017].

[2] Y. Mejova, *Sentiment Analysis: An Overview, Comprehensive Exam Paper,* Iowa, 2009.

[3] S.-M. Kim un E. Hovy, «Determining the sentiment of opinions,» %1 *Proceedings of the 20th international conference on Computational Linguistics*, Ženēva, Šveice, 2004.

[4] B. Liu, Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, 2012, pp. 7-29.

[5] E. Kouloumpis, T. Wilson un J. Moore, «Twitter Sentiment Analysis: The Good the Bad and the OMG!,» %1 *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[6] A. Balahur un M. Turchi, «Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis.,» *Computer Speech & Language*, sēj. 28, nr. 1, pp. 56-75, 2014.

[7] M. Ogneva, «How companies can use sentiment analysis to improve their business,» Mashable, 2010..

[8] A. Go, R. Bhayani un L. Huang, *Twitter sentiment classification using distant supervision*, Stanford: CS224N Project Report, 2009..

[9] V. Vapnik, Statistical learning theory, New York: Wiley, 1998..

[10] S. Raschka, «Naive Bayes and Text Classification I - Introduction and Theory,» arXiv:1410.5329, 2014..

[11] H. Saif, Y. He un H. Alani, «Semantic sentiment analysis of twitter,» %1 *The Semantic Web–ISWC 2012*, 2012..

[12] K. Nigam, J. Lafferty un A. McCallum, «Using maximum entropy for text classification,» %1 *IJCAI-99 workshop on machine learning for information filtering*, 1999..

[13] K. Gediņš, *Automātiskā teksta emocionālās noskaņas noteikšana latviešu valodā,* Rīga, 2013..

[14] C. N. Dos Santos un M. Gatti, «Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts,» %1 *COLING*, 2014.

[15] W. McCulloch un W. Pits, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of mathematical biophysics,* sēj. 5, nr. 4, pp. 115-133, 1943..

[16] D. O. Hebb, The organization of behavior: A neuropsychological theory, Psychology Press (2005.), 1949..

[17] D. E. Rumelhart, G. E. Hinton un R. J. Williams, «Learning representations by back-propagating errors,» *Nature,* sēj. 323, pp. 533-536, 1986..

[18] NVIDIA, «CUDA Parallel Computing Platform,» NVIDIA, [Tiešsaiste]. Available: http://www.nvidia.com/object/cuda_home_new.html. [Piekļūts 27. 05. 2017.].

[19] J. E. Dayhoff un J. M. DeLeo, «Artificial neural networks,» *Cancer,* sēj. 91, nr. S8, pp. 1615-1635, 2001..

[20] A. K. Jain, J. Mao un K. M. Mohiuddin, «Artificial neural networks: A tutorial,» *Computer,* sēj. 29, nr. 3, pp. 31-44, 1998.

[21] M. W. Gardner un S. R. Dorling, «Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,» *Atmospheric environment,* sēj. 32, nr. 14, pp. 2627-2636, 1998..

[22] A. Ng, «Machine Learning by Stanford University,» [Tiešsaiste]. Available: https://www.coursera.org/learn/machine-learning. [Piekļūts 26. 05. 2017.].

[23] A. Ng, «CS229 Lecture notes,» [Tiešsaiste]. Available: http://cs229.stanford.edu/notes/cs229-notes1.pdf. [Piekļūts 26. 05. 2017.].

[24] M. D. Zeiler, «ADADELTA: an adaptive learning rate method,» arXiv:1212.5701, 2012..

[25] «Twitter Developer Documentation,» Twitter Inc., [Tiešsaiste]. Available: https://dev.twitter.com/docs. [Piekļūts 22. Maijs 2017.].

[26] «Web Services Architecture,» World Wide Web Consortium, 11. Februāris 2004.. [Tiešsaiste]. Available: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest. [Piekļūts 22. Maijs 2017.].

[27] «Gnip,» [Tiešsaiste]. Available: https://gnip.com/. [Piekļūts 22. Maijs 2017.].

[28] J. Peisenieks, *Mašīntulkošanas Iespējas Twitter Sīkziņu Sentimenta Analīzē,* Rīga, 2014..

[29] M. Ghiassi, J. Skinner un D. Zimbra, «Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network,» *Expert Systems with Applications,* sēj. 40, nr. 16, p. 6266–6282, 2013..

[30] A. Pak un P. Paroubek, «Twitter as a Corpus for Sentiment Analysis and Opinion Mining,» %1 *LREc (Vol. 10 No. 2010),* 2010..

[31] T. Mikolov, G. Corrado, K. Chen un J. Dean, «Efficient estimation of word representations in vector space,» arXiv, 2013..

[32] T. Mikolov, I. Sutskever, K. Chen, G. Corrado un J. Dean, «Distributed representations of words and phrases and their compositionality,» %1 *Advances in neural information processing systems*, 2013..

[33] A. Znotiņš, «Word Embeddings for Latvian Natural Language Processing Tools,» %1 *Volume 289: Human Language Technologies – The Baltic Perspective*, IOS Press, 2016, pp. 167 - 173.

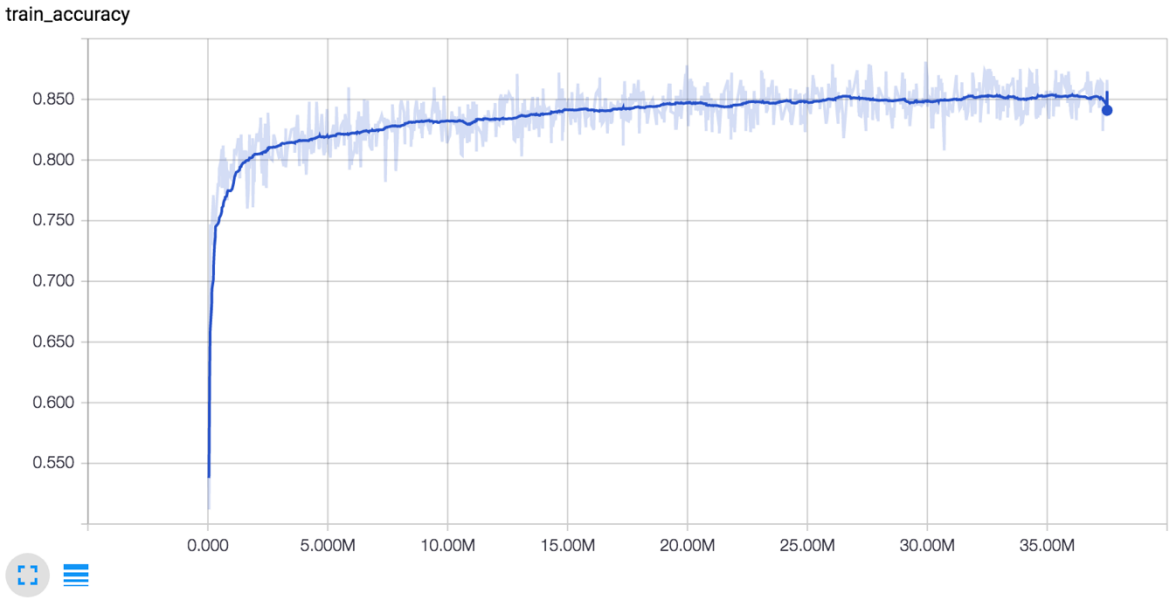
[34] P. Kralj Novak, J. Smailović, B. Sluban un I. Mozetič, «Sentiment of Emojis,» *PloS one e0144296,* sēj. 10, nr. 12, 2015.

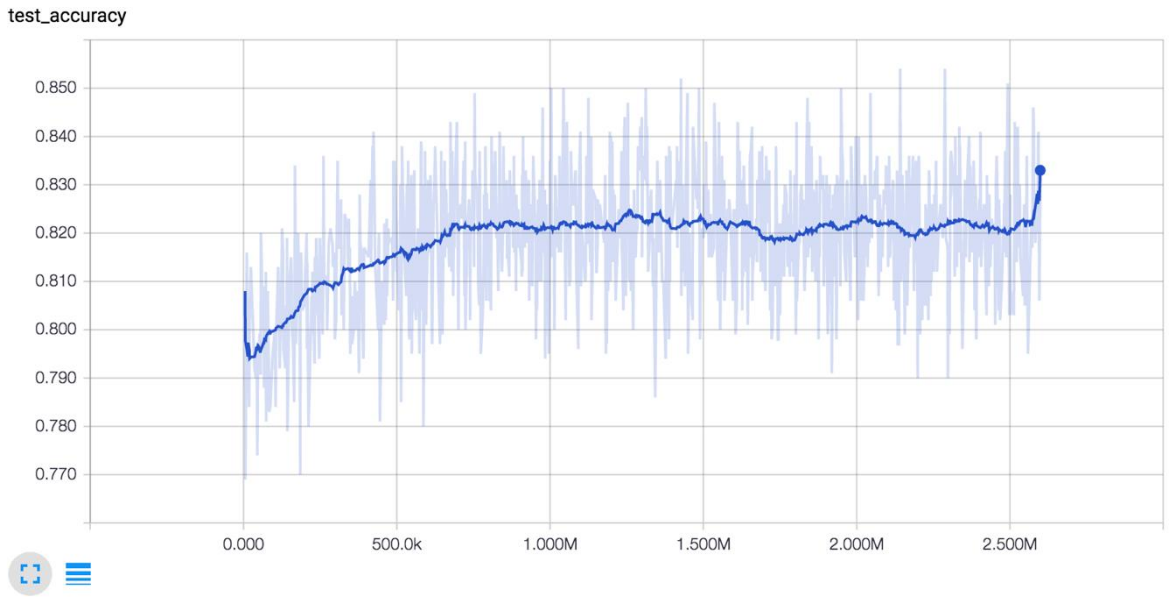
[35] M. Pamecha, «Monik's Blog,» 20. Jūnijs 2016. [Tiešsaiste]. Available: http://monik.in/a-noobs-guide-to-implementing-rnn-lstm-using-tensorflow/. [Piekļūts 23. Marts 2017.].

[36] A. Karpathy, «The Unreasonable Effectiveness of Recurrent Neural Networks,» 21. Maijs 2015.. [Tiešsaiste]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Piekļūts 23. Marts 2017.].

[37] M. Nielsen, «Improving the way neural networks learn,» [Tiešsaiste]. Available: http://neuralnetworksanddeeplearning.com/chap3.html. [Piekļūts 23. Marts 2017.].

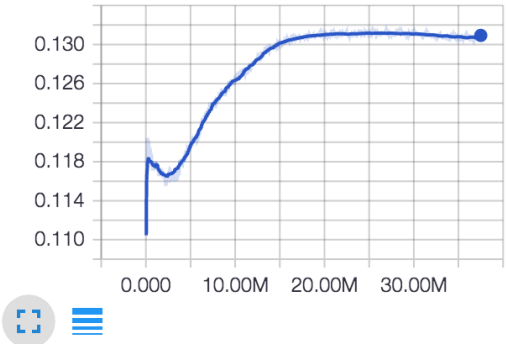
Tīkla treniņa un testa precizitāte pēc Sentiment140 datu apmācības procesa



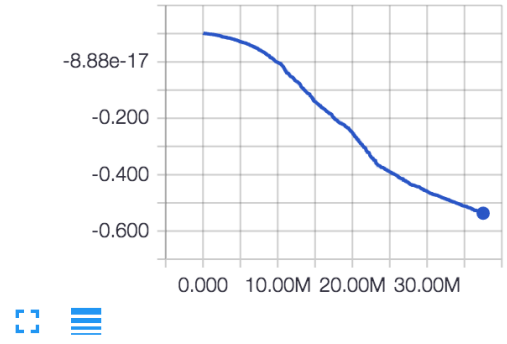


Tīkla izejas slāņa parametru rādītāji pēc Sentiment140 datu apmācības procesa

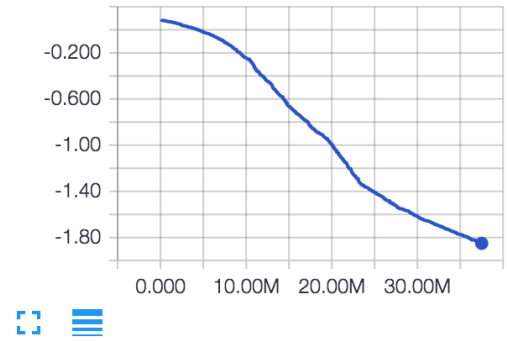
output_layer/bias_summary/max



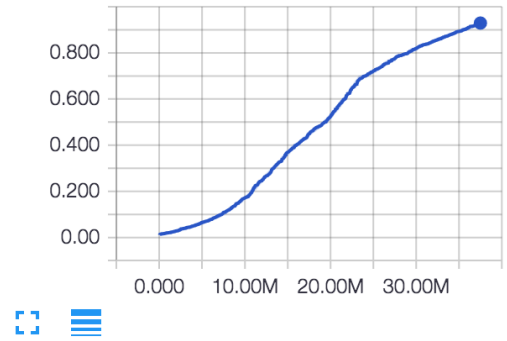
output_layer/bias_summary/mean



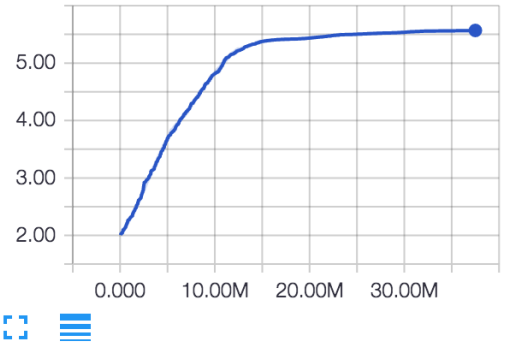
output_layer/bias_summary/min



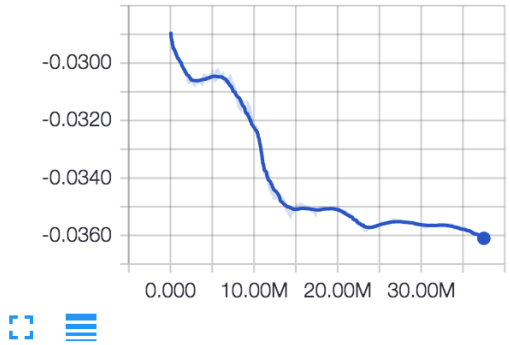
output_layer/bias_summary/stddev_1



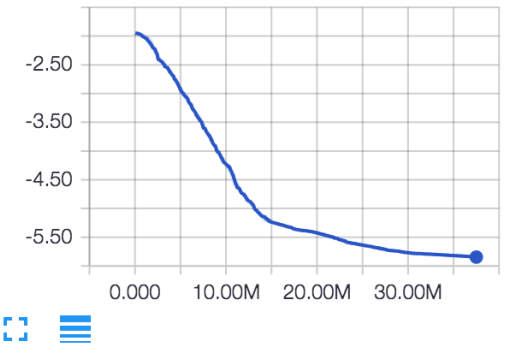
output_layer/weight_summary/max



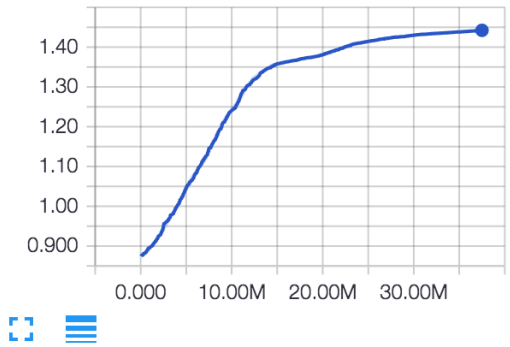
output_layer/weight_summary/mean



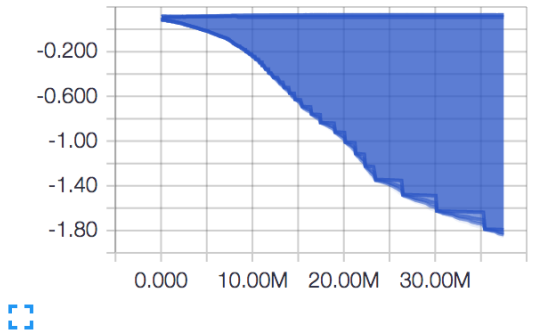
output_layer/weight_summary/min



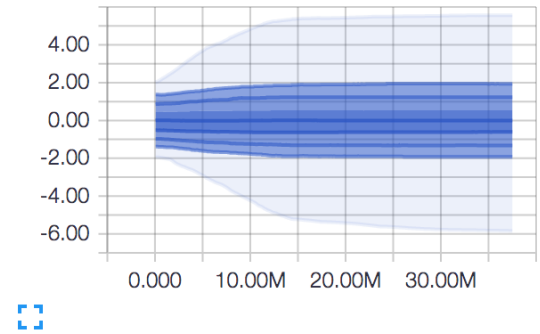
output_layer/weight_summary/stddev_1



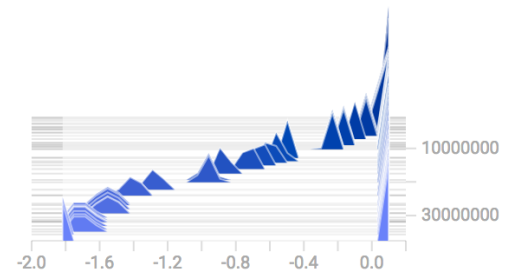
output_layer/bias_summary/histogram
decay_lr=0.001_dr=0.05_full



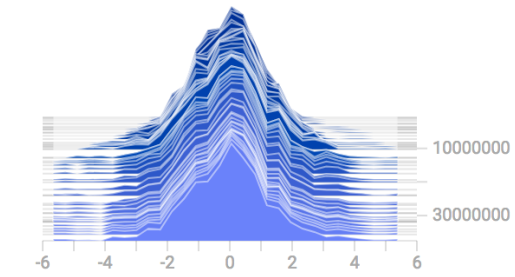
output_layer/weight_summary/histogram
decay_lr=0.001_dr=0.05_full



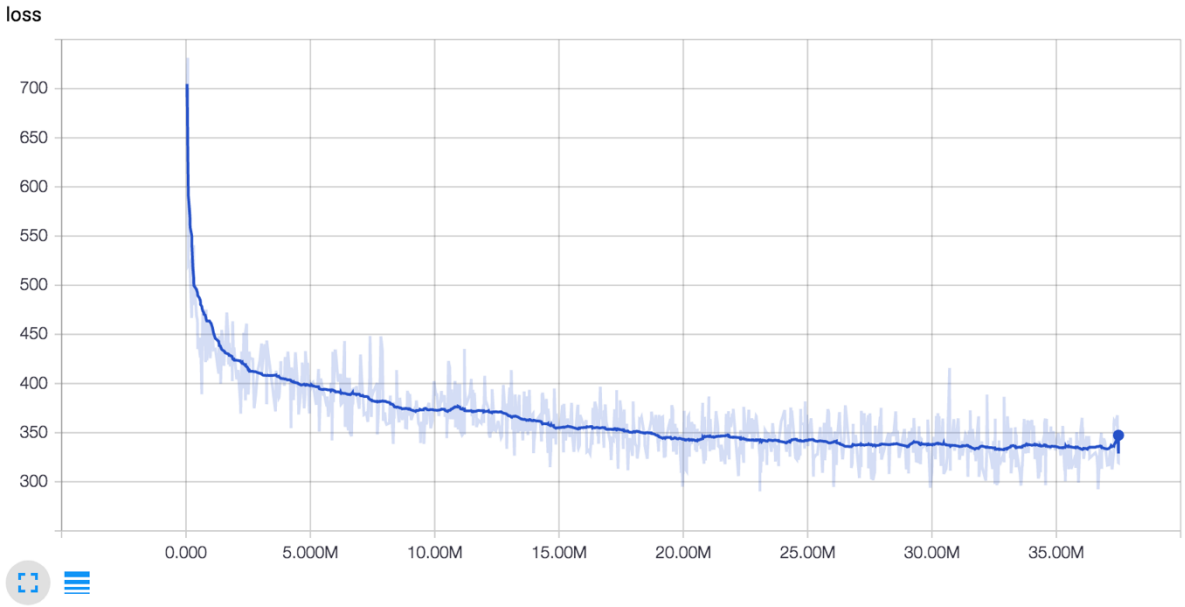
output_layer/bias_summary/histogram
decay_lr=0.001_dr=0.05_full



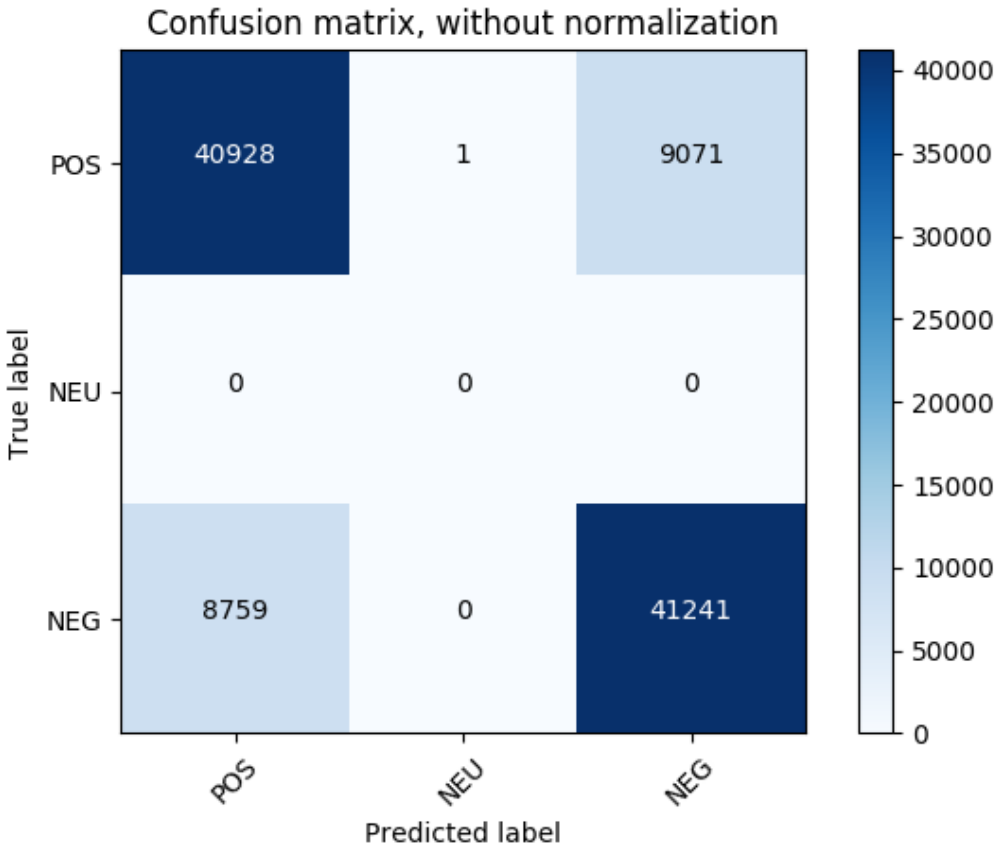
output_layer/weight_summary/histogram
decay_lr=0.001_dr=0.05_full



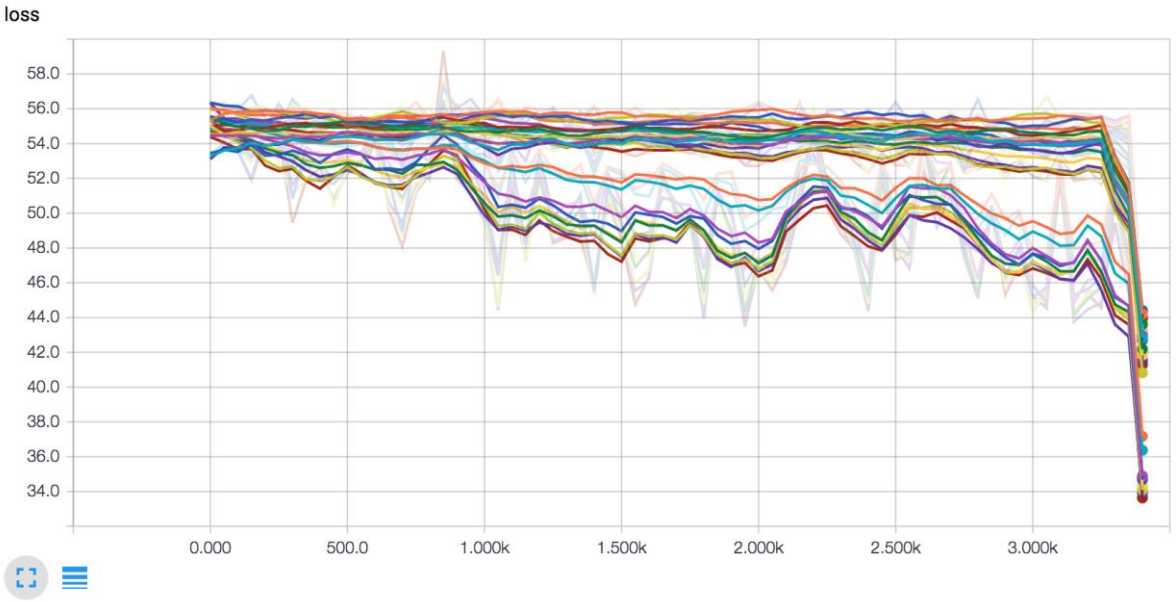
Tīkla kļūdas funkcijas rādītāji pēc Sentiment140 datu apmācības procesa



Tīkla testēšanā iegūtā pārpratumu matrica pēc Sentiment140 datu apmācības procesa



Tīkla kļūdas izmaiņa atkarībā no mācīšanās intensitātes (izmantojot latviešu valodas Twitter ziņu korpusu)



Bakalaura darbs “Sabiedrības attieksmes modelēšana, izmantojot sentimenta analīzi” izstrādāts LU Datorikas fakultātē.

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

Autors: _____ Dāvis Nicmanis

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs: Mg. vad. zin. Pēteris Paikens _____ 29.05.2017.

Recenzents: docents Dr. sc. comp. Normunds Grūzītis

Darbs iesniegts Datorikas fakultātē 29.05.2017.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2017. prot. Nr. ____

Komisijas sekretārs(-e): _____