VENTSPILS AUGSTSKOLA

INFORMĀCIJAS TEHNOLOĢIJU FAKULTĀTE

BAKALAURA DARBS

Ziņu portālu rakstu klasifikācija (ar mašīnmācīšanās algoritmiem)

Autors: Ventspils Augstskolas
Informācijas tehnoloģiju fakultātes
bakalaura studiju programmas
„Datorzinātnes”
3. kursa students
Matīss Kalniņš
Matrikulas Nr. 23020018

_____

(paraksts)

Fakultātes dekāns: doc. Dr.sc.comp. Vairis Caune

_____

(paraksts)

Zinātniskais vadītājs: Mg.sc.comp. Agris Traškovs

_____

(paraksts)

Recenzents:

_____

(Ieņemamais amats, zinātn. nosaukums, vārds, uzvārds)

_____

(paraksts)

Ventspils, 2023

Saturs

ANOTĀCIJA

Darba nosaukums:	Ziņu portālu rakstu klasifikācija (ar mašīnmācīšanās algoritmiem)
Darba autors:	Matīss Kalniņš
Darba vadītājs:	Mg.sc.comp. Agris Traškovs
Darba apjoms:	50. lpp, 6 tabulas, 21 attēli, 12 formulas, 21 bibliogrāfiskās norādes, 2 pielikumi
Atslēgas vārdi:	Dabiskās valodas apstrāde, mašīnmācīšanās, tekstu klasifikācija

Bakalaura darbā ir aprakstīta dabīgās valodas apstrāde un kā mašīnmācīšanās metodes var palīdzēt risināt teksta klasifikācijas problēmu, konkrēti apskatot tieši ziņu klasifikāciju latviešu valodas rakstiem.

Darba ietvaros tiek ievākta rakstu kopa no ziņu portāliem un pārbaudīts kāda pieeja sniedz augstāko precizitāti teksta klasifikācijai latviešu valodā. Tiek izvērtētas un salīdzinātās dažādas pazīmju ģenerēšanas pieejas un apmācības algoritmi (naivā Bajesa metode, loģistiskā regresija, lēmumu koki, atbalsta vektora mašīnas, neironu tīkli).

Papildus apskatītas dažādas atvērtā pirmkoda bibliotēkas mašīnmācīšanās problēmu risināšanai kā scikit-learn un Tensorflow, to praktiskais pielietojums dabīgo valodu apstrādei.

ABSTRACT

The title:	Classification of news articles (with machine learning algorithms)
Author:	Matīss Kalniņš
Academic Advisor:	Mg.sc.comp. Agris Traškovs
The volume of the work:	50. pages, 6 tables, 21 images, 12 equations, 21 literature sources, 2 appendices
Keywords:	Natural language processing, machine learning, text classification

The bachelor thesis describes natural language processing and how machine learning methods can help to resolve text classification problems, focusing specifically the classification of news articles in the Latvian language.

As part of this work a data set of articles is gathered from Latvian news websites and the best approach is researched for achieving the highest accuracy of Latvian text classification. Various feature generation approaches and learning algorithms (e.g. Naïve Bayes, logistic regression, decision trees, support vector machines, neural networks) are evaluated and compared.

In addition, various open source libraries for machine learning as scikit-learn and Tensorflow along with their practical applications for natural language processing are explored as part of this work.

IZMANTOTIE SAĪSINĀJUMI UN TERMINI

DVA - Dabisko valodu apstrāde (angliski - natural language processing)

LSTM - rekurentā mākslīgā neironu tīkla paveids ar atmiņas elementu (angliski - Long Short-Term Memory)

GLUE - standartizēta pieeja valodas apstrādes modeļu veiktspējas novērtēanai, sevī ietver 9 dažādus valodas apstrādes uzdevumus (angliski - General Language Understanding Evaluation)

TF-IDF - Terminu biežums - inversais dokumentu biežums (angliski - term frequency - inverse document frequency)

Epoha - Apmācības periods. Apmācības procesa daļa, kurā neironu tīkls tieši vienu reizi tiek apmācīts uz visiem apmācības piemēriem (no angliskā termina - epoch).

PA - pareiza atbilsme (angliski - true positive)

PA - pareiza atbilsme (angliski - true positive)

PN - pareiza neatbilsme (angliski - true negative)

KA - kļūdaina atbilsme (angliski - false positive)

KN - kļūdaina neatbilsme (angliski - false negative)

IEVADS

Mūsdienās internets ir kļuvis par galveno informācijas avotu lielai daļai cilvēku, kuri ikdienā ar dažādu mediju palīdzību caur to gūst informāciju par jaunākajām aktualitātēm savā rajonā, valstī un pasaulē. Svarīga loma informācijas iegūšanā un izplatīšanā ir arī pareizai teksta klasifikācijai, lai šī informācija sasniegtu vēlamo lasītāju. Pārsvarā problēma tiek atrisināta autoram klasificējot savu darbu jau izveides procesā, tomēr bieži ar to vien nepietiek – tiek pārpublicēti raksti no ārējiem resursiem, mainās kategoriju iedalījums, aktuālas kļūst jaunas tēmas u.t.t. Lai gan arī šādos gadījumos klasifikāciju iespējams darīt manuāli, pie liela informācija apjoma kļūst jēgpilni šo klasifikāciju automatizēt ar mašīnmācīšanās algoritmiem, ietaupot laiku un resursus.

Pirms darba uzsākšanas, veicot izpēti par labākajiem algoritmiem tekstu klasifikācijai, konstatēts, ka viennozīmīgi labākā pieeja problēmas risināšanai neekstistē, optimālā pieeja atkarīga no daudziem faktoriem – gan no dažādām datu kopas īpašībām (valoda, temati, tekstu garumi, valodas stili u.t.t.), gan klašu skaita klasifikācijā, gan pielietotajām priekšapstrādes metodēm, gan cik labi algoritmi mērogojas ar dažādiem datu kopu izmēriem. Līdzīgu ieskatu vispārējā problēmvidē sniedz arī "Papers With Code", kas apkopo ar mašīnmācīšanos saistītās publikācijas un tajos iekļauto modeļu veiktspējas rādītājus, to starpā arī 1091 publikācijas par tekstu klasifikāciju. Šajā resursā iespējams novērot, ka pēdējos gados labākos rezultātus pārsvarā, bet ne vienmēr, sniedz lielo valodu modeļi kā XLNet un BERT. Konkrēti šos iepriekš apmācītos modeļus nav iespējams pielietot latviešu tekstu klasifikācijai - šie modeļi ir apmācīti uz ļoti liela angļu valodas tekstu apjoma - BERT gadījumā 3,3 miljardu vārdu korpusa, XLNet gadījumā - jau krietni lielāka. Salīdzinoši lielākais latviešu valodas korpuss šobrīd (LVK2022) satur tikai 101 miljonu vārdu. Nākošo labāko algoritmu veiktspējas atšķirības kļūst mazāk izteiktas – labus rezultātus sasniedz gan dažādas rekurento neironu tīklu arhitektūras (LSTM balstīti modeļi), gan konvolūcijas neironu tīkli, gan arī vienkāršākas pieejas kā atbalsta vektora mašīnas. Šo secinājumu rezultātā izlemts izpētīt cik labi klasifikāciju iespējams veikt ar plašu spektru ar klasifikācijas metodēm – sākot no vienkāršiem klasifikācijas algoritmiem kā Naivā Bajesa metode, atbalsta vektora mašīnas u.c., turpinot ar dažādām neironu tīklu arhitektūrām, kā arī pielāgojot BERT arhitektūrā balstīto LVBERT modeli.

Darba mērķis ir izveidot mašīnmācīšanās modeli, kas ar augstu precizitāti spētu klasificēt ziņu portālu rakstus latvieu valodā. Lai sasniegtu šo mērķi, tiek izvirzīti sekojoši uzdevumi:

1. Veikt literatūras izpēti par mašīnmācīšanos un tekstu klasifikāciju
2. Izveidot rāpuli ar kura palīdzību izgūt un marķēt ziņu portālu rakstus, pielietojamus modeļu apmācībā
3. Implementēt dažādus mašīnmācīšanās algoritmus tekstu klasifikācijai
4. Izpētīt kā dažādas pazīmju ģenerēšanas pieejas ietekmē klasifikācijas rezultātus
5. Veikt precizitātes novērtējumus un salīdzināt cik labi dažādos algoritmos balstīti modeļi spēj veikt latviešu valodas tekstu klasifikāciju

Lai veiktu algoritmu implementēšanu un analīzi tiks izmantota programmēšanas valoda Python un plaši pielietotas bibliotēkas mašīnmācīšanās problēmu risināšanai (scikit-learn, Tensorflow).

Darbs sastāv no 4 nodaļām. Pirmajā nodaļā tiek apskatīta dabiskās valodas apstrāde un mašīnmācīšanās, pamatjēdzieni uz kuriem balstās nākošo nodaļu saturs.

Otrajā nodaļā uzsvars tiek likts uz pazīmju ģenerēšanas aprakstu un literatūras izpēti par to. Šis ir svarīgs solis pirms mašīnmācīšanās algoritmu pielietošanas, pārvēršot teksta informāciju apstrādei piemērotā formā un ietekmējot to cik veiksmīgu modeli būs iespējams izveidot.

Trešajā nodaļā konkrēti tiek apskatīti dažādi izplatītākie algoritmi ar kuriem veikt teksta klasifikāciju – sākot no vienkāršākām pieejām kā atbalsta vektora mašīnas, turpinot ar neironu tīkliem un lielajiem valodas modeļiem kā BERT.

Ceturtajā nodaļā tiek apskatīta rāpuļa un klasifikācijas modeļu praktiskā izveide, cik labi dažādi modeļi spēja sasniegt vēlamo rezultātu.

1. MAŠĪMĀCĪŠANĀS VALODAS APSTRĀDĒ

1.1. Dabiskās valodas apstrāde

Dabiskās valodas apstrāde (angliski – natural language processing jeb NLP) ir daudznozaru joma, kas apvieno lingvistikas, datorzinātnes un mašīnmācīšanās elementus, lai ļautu datoriem saprast, interpretēt un ģenerēt cilvēka valodu.

DVA sastāv no vairākām pamata komponentēm:

- Tokenizācija: teksta sadalīšanas process vārdos vai frāzēs (tokenos)
- Morfoloģiskā marķēšana: gramatikas marķējumu piešķiršana vārdiem
- Sintakses parsēšana: teikumu gramatiskās struktūras analīze
- Nosaukto entitāšu atpazīšana: nosaukumu atpazīšana un kategorizācija, piemēram, personvārdi, datumi un atrašanās vietas
- Lemmatizācija: vārdu pārveidošana pamatformā

Pielietojot daļu no šīm komponentēm tālāk iespējami sarežģītāki pielietojumi teksta apstrādei – kategorizācijai, noskaņojuma analīzei, mašīntulkošanai, čatbotu izveidei.

Sākotnējās DVA sistēmas balstījās uz manuāli izstrādātām noteikumiem, taču šīs sistēmas ir ierobežotas ar savu nespēju apstrādāt cilvēka valodas daudzveidību un sarežģītību, kā rezultātā DVA sistēmas mūsdienās bieži tiek veidotas tieši ar mašīnmācīšanās iesaisti.

1.2. Mašīnmācīšanās

Mašīnmācīšanās ir mākslīgā intelekta nozare, kas nodarbojas ar datorprogrammu izstrādi, kuras, izmantojot algoritmus un statistikas modeļus, mācās no datiem un uzlabo savu precizitāti. Toms Mičels savukārt apraksta mašīnmācīšanās jomu, izvirzot centrālo jautājumu, ko tā pēta: ”Kā mēs varam izveidot datoru sistēmas, kas automātiski uzlabojas, iegūstot pieredzi, un kādi ir pamatlikumi, kas nosaka visus mācīšanās procesus?” [1].

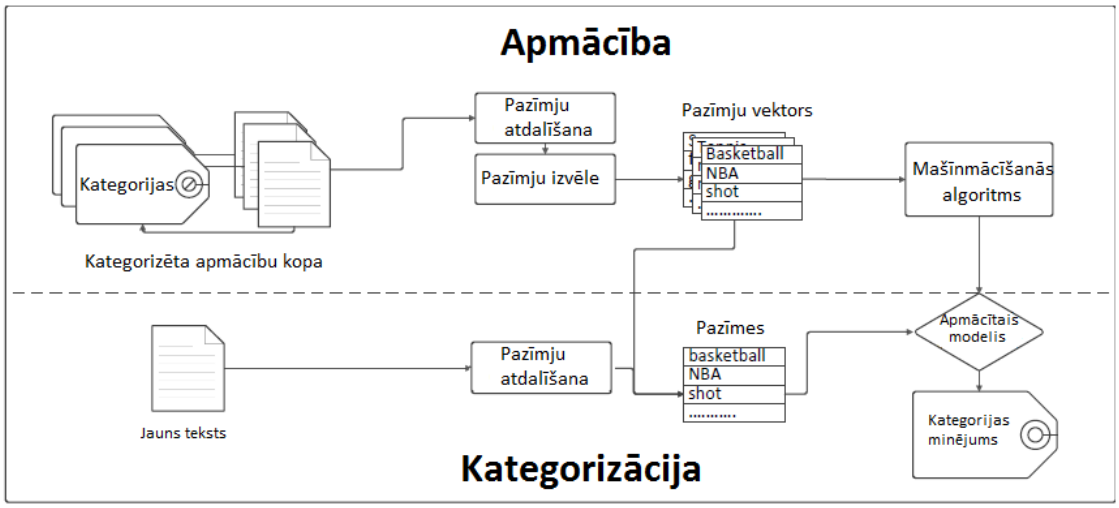
Induktīvā mācīšanās ir mašīnmācīšanās apakšnozare, kas specifiski nodarbojas ar modeļu mācīšanos no novērojumiem. Šajā nozarē mācīšanās uzdevumi bieži tiek raksturoti, pamatojoties uz atgriezenisko saiti, kas tiek sniegta apmācības veicējam [2], šādi:

- Uzraudzīta mācīšanās: Apmācības procesā tiek sniegts vēlams izvads katram novērojumam. Mērķis ir iemācīties funkciju, kas paredz pareizo izvades vērtību brīdī kad tiek sniegts iepriekš neredzēts novērojums.
- Neuzraudzīta mācīšanās: Apmācības laikā netiek sniegts izvads. Mērķis ir atklāt paraugus un regulāras pazīmes datos.
- Stimulētā mācīšanās: Šis ir īpašs uzraudzītās mācīšanās gadījums, kur apmācības posmā tiek sniegta atlīdzība pēc katras darbības.

Uzraudzītas mācīšanās gadījumus, kad uzdevums ir iemācīties diskrēti vērtētu funkciju, sauc par klasifikāciju. Uzdevums, kad jāiemācās nepārtraukti vērtēta funkcija, tiek saukts par regresiju. Savukārt klasterošana ir neuzraudzītās mācīšanās uzdevums, kas atrod līdzīgu objektu grupas datu kopā.

1.3. Tekstu klasifikācija

Teksta klasifikācijas mērķis ir tekstu piesaiste konkrētai kategorijai, balstoties uz teksta saturu. Šāda klasifikācija ir ļoti izplatīta ziņu portālos un citur, kur nepieciešams kategorizēt lielu rakstu daudzumu, piemēram akadēmisko darbu datubāzēs. Mašīnmācīšanās algoritmi ļauj rast risinājumu automātiskai šādu tekstu kategorizēšanai un daudzām citām klasifikācijas problēmām, piemēram, ar augstu precizitāti noteikt vai ienākošais e-pasts ir vai nav mēstule. Iespējams arī veikt sentimentu analīzi un noteikt cilvēku attieksmi par kādu konkrētu tematu, piemēram, M. Kandias ir apskatījis kā ar tekstu klasifikācijas palīdzību noteikt negatīvu attieksmi pret likumsargiem, balstoties uz konkrēta lietotāja ierakstiem vietnē YouTube [3].



1.1. att. **Mašīnmācīšanās tekstu klasifikācijai**

Teksta klasifikācijas piemēru ar mašīnmācīšanās pielietojumu iespējams redzēt attēlā 1.1.. Pieņemsim, ka dota datu kopa ar sporta ziņām, un risināmā problēma ir - kā klasificēt jaunu dokumentu, piešķirot tam atbilstošā sporta veida kategoriju. Sākumā būs nepieciešami apmācības dokumenti (ar klases marķējumiem) no kuriem mācīties. Tālāk katru ziņu mēs pārveidojam par pazīmju kopu, kuru vektorizētā veidā mēs varam padot tālāk mašīnmācīšanās algoritmam. Ar šo informāciju algoritms izveido modeli, kas var paredzēt iepriekš neredzētu tekstu kategoriju.

2. PAZĪMJU ĢENERĒŠANA

2.1. Tekstu priekšapstrāde

Pirms iespējams izveidot klasifikācijas modeli, DVA problēmvidē nepieciešams veikt teksta priekšapstrādi, lai dati būtu pielietojami tālākā apstrādē. Tas sevī ietver gan dažādu teksta fragmentu atmešanu, gan pārveidošanu formā kuru spētu saprast apmācības algoritmi. Tālāk apskatīti konkrēti priekšapstrādes soļi.

Vārdu atdalīšana jeb tokenizācija

Lai tekstu izmantotu klasificēšanā, ir jāspēj atdalīt atsevišķas šī teksta daļas. To iespējams paveikt dažādos veidos – tekstu iespējams sadalīt pa individuāliem vārdiem vai arī secīgu vārdu grupām. Vārdu atdalīšanai var izmantot dažādus atdalošos simbolus, piemēram atstarpi un jaunas līnijas sākuma simbolu (\n'). Tomēr ne visi atdalošie simboli ir vienno-zīmīgi, piemēram punkts var būt gan kā teikuma beigas, gan kā daļa no konkrētas vērtības (skaitlis 10.5). Vārdu atdalīšana secīgās vārdu grupās izmanto n-grammas, kas ir n secīgu vārdu un/vai simbolu kopa tekstā. Izmantojot n-grammas, iespējams iegūt plašāku teksta kontekstu no atdalītajiem vārdiem.

Sakņu atdalīšana un lemmatizācija

Apstrādājot tekstus, svarīgi ir arī ņemt vērā faktu, ka dažādos tekstos viens un tas pats vārds bieži tiek lietots dažādos locījumos un visbiežāk locījumam nav ietekme uz to vai teksts pieder konkrētai kategorijai vai nē. Svarīgi arī ņemt vērā, ka liels individuālo vārdu atkārtojums dažādos locījumos palielina klasifikācijai nepieciešamo laiku un resursus. Sakņu atdalīšana un lemmatizācija risina šo problēmu, pārvēršot vārdus to pamatformā.

Stopvārdu dzēšana

Bieži vien noderīga ir arī tā sauktā stopvārdu (angliski - stopwords) dzēšana no apstrādāmajiem datiem. Tie ir vārdi kuri neietekmē teksta saturu un to biežais lietojums var atstāt negatīvu ietekmi uz klasifikācijas akurātumu. Latviešu valodā piemērs šādiem vārdiem būtu

palīgvārdi, kuri tiek saprasti kā saikļi (un, bet, vai u.c.), prievārdi (uz, no u.c.), partikulas (arī, diezin, gan u.c.).

2.2. Vektorizācija

Teksta vektorizācija ir process, kurā teksta informācija tiek pārveidota skaitļu formā, ko tālāk savukārt var izmantot mašīnmācīšanai. Šis solis ir būtisks, jo lielākā daļa mašīnmācīšanās algoritmu strādā ar skaitļiem, bet teksta dati ir paši par sevi neskaitliski. Teksta vektorizācijai ir vairākas metodes, populārākās no tām uzskaitītas zemāk.

Vārdu maiss

Vārdu maiss – tā ir nesakārtota vārdu kopa, kur vārdu secība tiek ignorēta, saglabājot tikai vārdu biežumu dokumentā [4]. Teksts tiek pārveidots vektorā, kur katram unikālam vārdam dokumentā tiek piešķirts indekss, kā vērtību indeksā norādot konkrētā vārda biežumu. Ar šo pieeju mēs veicam šādus pieņēmumus:

- Tekstu iespējams analizēt, ignorējot vārdu / tekstvienību secību.
- Nepieciešams zināt tikai kuri vārdi / tekstvienības atrodas tekstā un cik bieži tie atkārtojas.

TF-IDF

Terminu biežums - inversais dokumentu biežums (angliski, saīsināti - TF-IDF) ir vektorizācijas paveids ko izmanto, lai novērtētu termina (vārda) svarīgumu dokumentā attiecībā uz dokumentu kopu (korpusu) [5].

Termina biežums (TF):

Termina biežums nosaka, cik bieži konkrēts termins parādās konkrētā dokumentā. To aprēķina kā attiecību starp termina parādīšanos dokumentā un kopējo terminu skaitu šajā dokumentā. Termina biežuma (TF) formula ir šāda:

$$TF(t,d) = \frac{f(t,d)}{|d|} \tag{2.1}$$

Kur:

- $TF(t,d)$ ir termina biežums terminam t dokumentā d.

- $f(t, d)$ ir termina t biežums dokumentā d.
- $|d|$ ir kopējais terminu skaits dokumentā d.

Inversais dokumenta biežums (IDF): Inversais dokumenta biežums nosaka cik unikāls vai svarīgs ir termins visā dokumentu kopā (korpusā). To aprēķina kā logaritmisku attiecību starp visu dokumentu kopā esošo dokumentu skaitu un dokumentu skaitu, kuros šis termins parādās. Inversās dokumentu biežuma (IDF) formula ir šāda:

$$IDF(t, D) = \log\left(\frac{|D|}{|d \in D : t \in d|}\right) \tag{2.2}$$

Kur:

- $IDF(t, D)$ ir termina t inversais dokumenta biežums kopā D.
- $|D|$ ir kopējais dokumentu skaits korpusā.
- $|d \in D : t \in d|$ ir dokumentu skaits, kas satur terminu t.

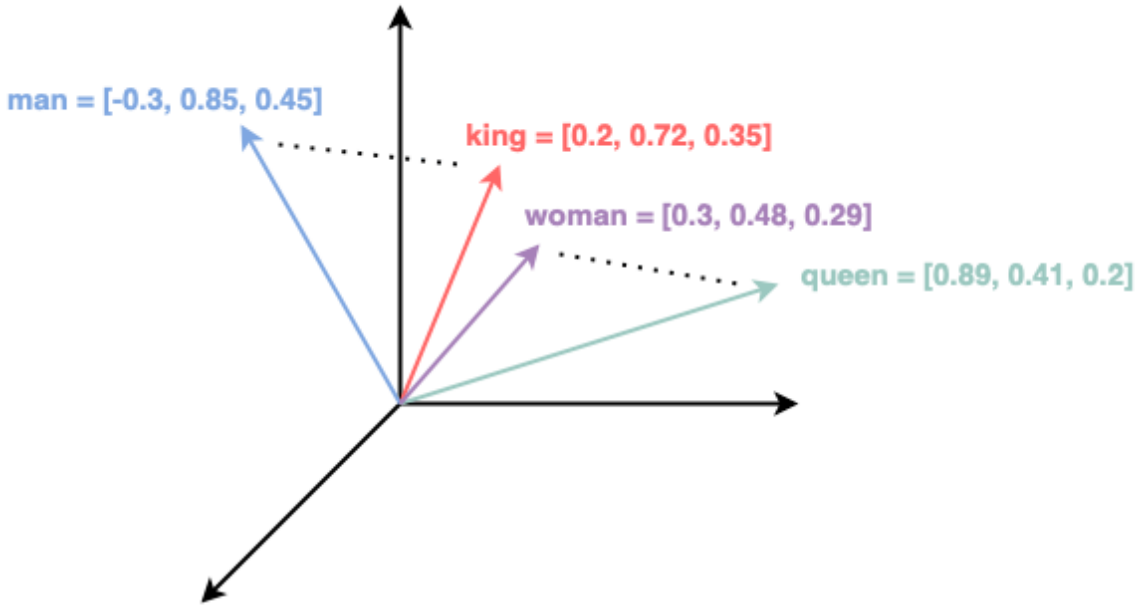
TF-IDF beigu rezultāts ir termina biežuma (TF) un inversā dokumenta biežuma (IDF) reizinājums:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D) \tag{2.3}$$

TF-IDF rezultāts atspoguļo, cik svarīgs ir vārds konkrētajā dokumentā, ņemot vērā visu tekstu kopu. Augstāki TF-IDF vērtējumi tiek piešķirti terminiem, kas bieži parādās dokumentā, bet reti visā kopā. Tas palīdz uzsvērt terminu svarīgumu, kuri ir raksturīgi tieši konkrētiem dokumentiem, vienlaikus samazinot kopīgu terminu nozīmi, kas parādās daudzos dokumentos.

Vārdlietojuma kartējums

Vārdlietojuma kartējums (angliski - word embedding) ir jaunāka metode, kur vārdi tiek attēloti kā skaitliski vektori daudzdimensiju telpā. Šie vektori spēj saglabāt informāciju par vārdu kontekstu / nozīmi / saikni ar citiem vārdiem, respektīvi - vārdi ar līdzīgu nozīmi vai pielietojumu ir attēloti ar vektoriem, kas ģeometriski tuvi viens otram vektoru telpā. Piemēram, labi apmācītā vārdu iegulšanas modelī vārdi ”karalis” un ”karaliene” tiek attēloti kā vektori, kas ir tuvu viens otram, norādot to semantisko līdzību [6].



2.1. att. **Vārdlietojuma kartējums vektora telpā [6]**

Vārdlietojuma kartējums ļauj arī veikt dažādas operācijas ar vārdiem vektoru telpā, to skaitā arī saskaitīšanu un atņemšanu. Piemēram, ”karalis - vīrietis + sieviete” varētu izveidot vektoru, kas vektora telpā ir tuvu vārdam ”karaliene”.

Pazīmju izvēle

Iepriekš tika apskatīts kā atlasīt pazīmes no dokumentu kopas. Atkarībā no apskatīto tekstu daudzuma un sarežģītības, rezultātā var tikt iegūts liels pazīmju skaits, kas var apgrūtināt mašīnmācīšanās algoritmu pielietošanu. Pārāk plaša vai pārāk maza pazīmju kopa var atstāt negatīvu iespaidu uz modeļa veiktspēju. Lai risinātu šo problēmu tiek apskatīta pazīmju izvēle.

Viena no izplatītākajām metodēm, kas samazina pazīmju skaitu ir retu vārdu izņemšana. Dēļ to retuma, tās visdrīzāk nav pazīmes, kas ir raksturīgas visiem kategoriju tekstiem, un nepalīdzēs izveidot precīzāku modeli.

3. MAŠĪNMĀCĪŠANĀS ALGORITMI

3.1. Klasiskie algoritmi tekstu klasifikācijai

Naivā Bejesa metode

Naivā Bejesa mašīnmācīšanās algoritms bieži tiek lietots tieši klasifikācijas uzdevumos tā veiktspējas un efektivitātes dēļ. Tas balstās uz Bejesa teorēmu, kas ir viena no pamatteorēmām varbūtību teorijā. Šī teorija apraksta notikuma varbūtību, pamatojoties uz iepriekš zināmiem datiem. Teksta klasifikācijas kontekstā teorēma palīdz mums aprēķināt varbūtību dokumenta piederībai noteiktai klasei.

Bejesa teorēmu var izteikt šādi:

$$P(klase|dokuments) = \frac{P(klase) \cdot P(dokuments|klase)}{P(dokuments)} \tag{3.1}$$

Kur formulā 3.1.:

- $P(klase|dokuments)$ ir varbūtība, ka dokuments pieder norādītajai klasei.
- $P(klase)$ ir apriorā klases varbūtība.
- $P(dokuments|klase)$ ir varbūtība novērot dokumentu, zinot klasi.
- $P(dokuments)$ ir varbūtība, ka dokuments parādās datu kopā.

”Naivais” aspekts Naivajā Bejesā nāk no pieņēmuma, ka pazīmes (vārdi tekstā) ir neatkarīgas. Citiem vārdiem sakot, mēs pieņemam, ka katra vārda klātbūtne vai neesamība dokumentā ir neatkarīga no citu vārdu klātbūtnes vai neesamības. Šis ir vienkāršs pieņēmums, bet tāds kurš bieži darbojas praksē.

Loģistiskā regresija

Loģistiskā regresija modelē varbūtību, ka dokuments pieder konkrētai klasei, izmantojot loģistisko (sigmoidālo) funkciju [7], kas nodrošina, ka izvades varbūtība ir starp 0 un 1.

Loģistiskā funkcija tiek definēta šādi:

$$P(y = 1|x) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

Kur formulā 3.2. $P(y = 1|x)$ ir varbūtība, ka dokuments pieder klasei 1 un z ir lineāra kombinācija no ievades pazīmēm un modeļa parametriem. Lineāra kombinācija tiek aprēķināta šādi:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \tag{3.3}$$

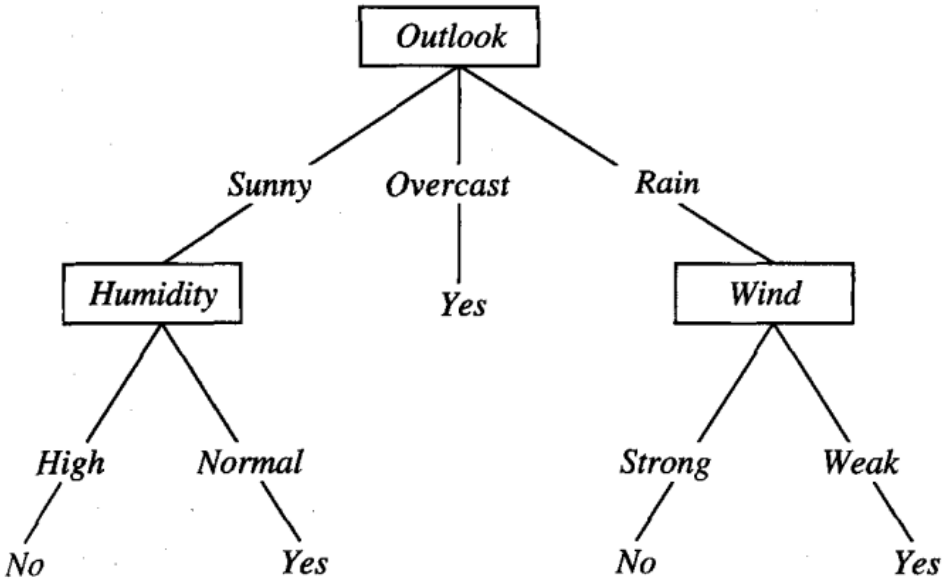
Kur formulā 3.3. $x_1, x_2, \ldots, x_n$ ir skaitliskās pazīmes, kas izgūtas no teksta dokumenta un $\theta_0, \theta_1, \ldots, \theta_n$ ir modeļa parametri, arī saukti par svariem vai koeficientiem.

Apmācības fāzē loģistiskās regresijas modelis mācās optimizēt savus parametrus (θ) no marķētajiem datiem, lai iegūtu pēc iespējas pareizākus minējumus.

Lēmumu koki

Lēmumu koka klasifikators izmanto koka modeli, lai prognozētu teksta klasi. Koks sastāv no viena saknes mezgla, kas ir uzskatāms par klasifikatora sākuma punktu. Pārējie mezgli ir lapu mezgli, ja tiem nav zaru, vai iekšējie mezgli. Iekšējie mezgli un saknes mezgls ir pazīmes un pārbaude, kas jāveic šai pazīmei. Katrs iespējamais testa rezultāts ir mezgla atzars, kas ved uz nākamo mezglu. Šādi veicot pārbaudes uz katra mezgla, tiek iziets caur visiem mezgliem līdz pirmajam lapu mezglam. Lapu mezgli galu galā norāda uz klasi, kurai šis teksts pieder. Citiem vārdiem – klase tiek paredzēta, sekojot ceļam no koka saknes mezgla, līdz tas saskaras ar lapas mezglu [8].

Apmācības algoritma mērķis šajā gadījumā ir izveidot lēmumu koku, pamatojoties uz apmācību datu piemēriem. Tomēr algoritmam ir jāizvairās veidot koku, kas pārmērīgi atbilst apmācības datiem. Tāpēc optimālais koks ir mazākais lēmumu koks, kas vislabāk atšķir klases.

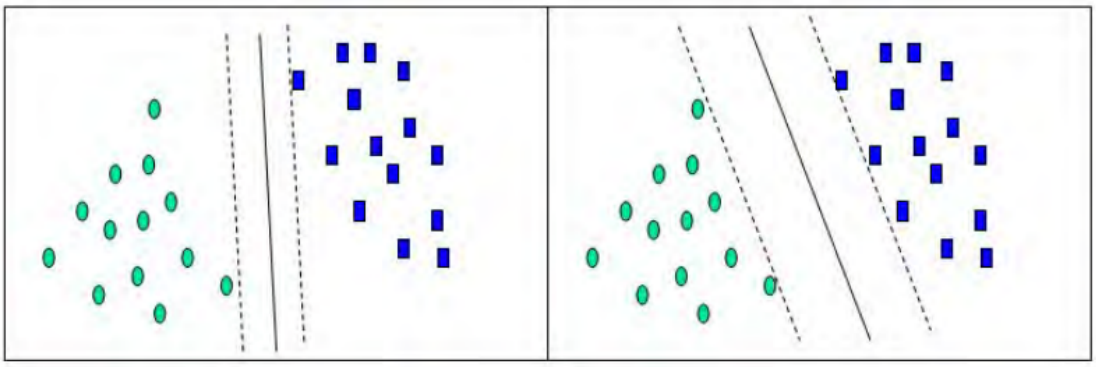


3.1. att. **Lēmumu koka ilustrācija [8]**

Piemērā 3.1. apskatām vienkāršu šāda koka reprezentāciju, kur veicam klasifikāciju par to vai šis ir piemērots laiks tenisa spēlei ārpus telpām. Ar ieejas datiem kā laikapstākļi (outlook) – saulaini (sunny), mitrums (humidity) – augsts (high), mēs virzītos pa mezgliem "Laikapstākļi", "Mitrums" līdz lapas mezglam kurš klasificētu laiku kā nepiemērotu tenisa spēlei.

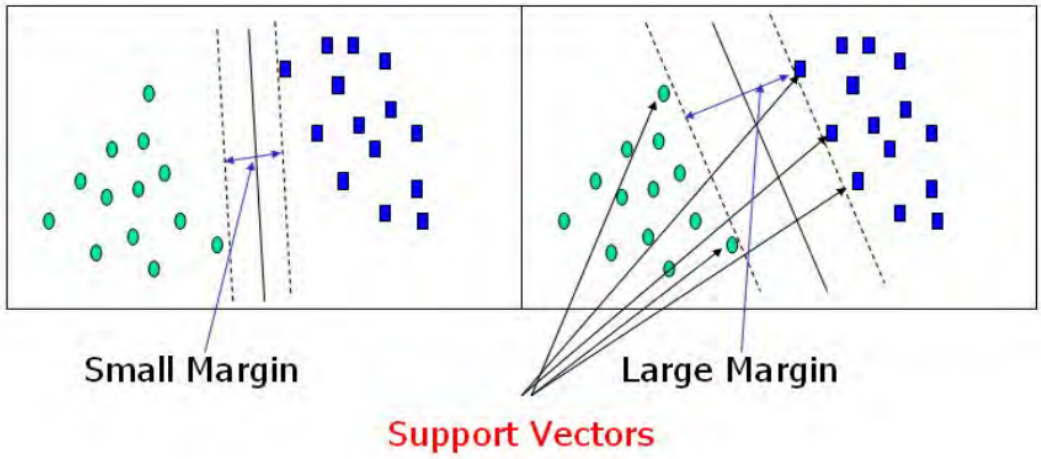
Atbalsta vektoru mašīnas (SVM)

Atbalsta vektora mašīnas [9] (angliski - support vector machines jeb SVM) ir pārraudzītās mācīšanās algoritms kurš ir diezgan populārs tieši klasificēšanas problēmu risināšanā. Šis algoritms veic klasifikāciju konstruējot n-dimensiju hiperplakni kura optimāli ierobežo datus divās nošķirtās kategorijās. Lai vieglāk ilustrētu algoritma darbību, varam apskatīt divdimensiju piemēru.



3.2. att. **Atbalsta vektora mašīnas algoritma ilustrācija [9]**

Šajā piemērā 3.2. gadījumi ar vienu kategoriju atrodas pa kreisi (apzīmēti ar zaļiem apļiem) un ar otru kategoriju – pa labi (apzīmēti ar ziliem kvadrātiem). Atbalstu vektora mašīnas analīze mēģinās atrast 1-dimensijas hiperplakni (līniju) kura atdala datus balstoties uz kategoriju kurai tie pieder. Ir praktiski neierobežots līniju skaits, kas spētu veikt šādu nodalījumu, attēlā norādīti 2 piemēri un atliek gūt atbildi uz jautājumu – kura līnija ir labāka kategorizācijas veikšanai vispārīgā gadījumā. Raustītās līnijas, kuras zīmētas paralēli atdalošajai līnijai, iezīmē attālumu starp atdalošo līniju un tai tuvāko vektoru. Šo attālumu sauc par robežu (angliski – margin). Vektori kas atrodas pie šīs robežas ir atbalsta vektori.



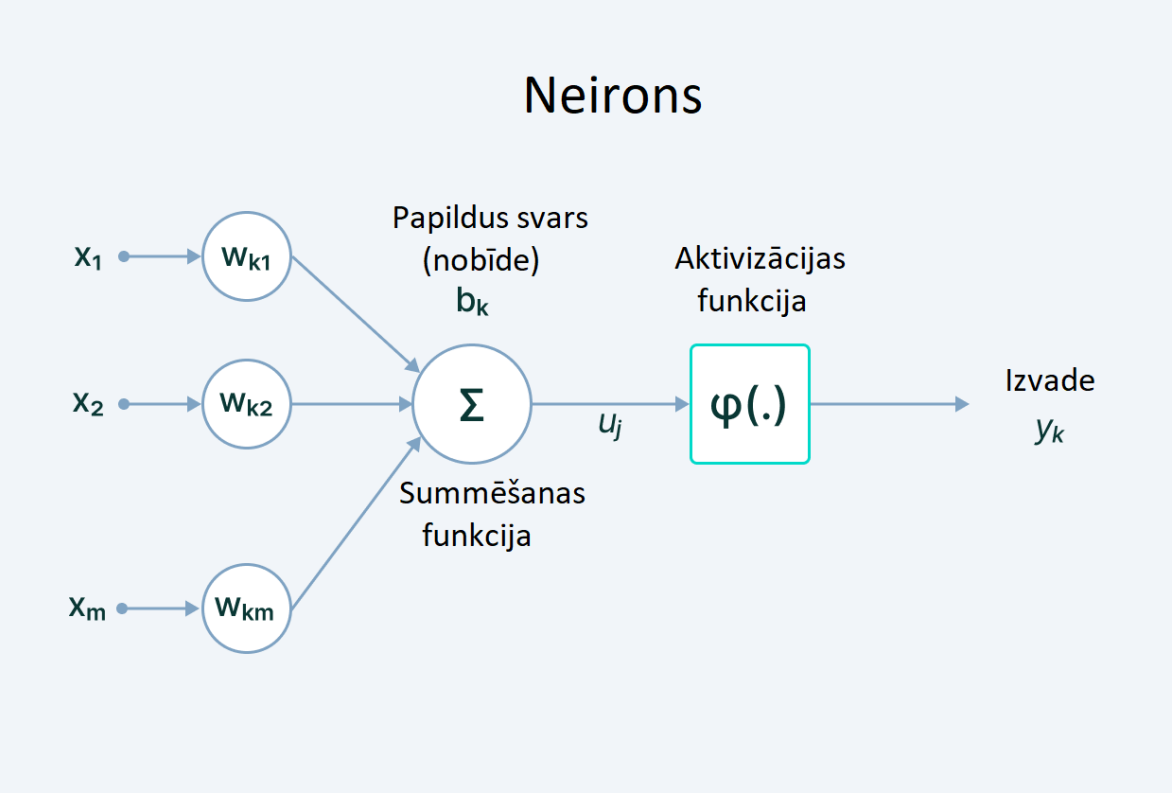
3.3. att. **Robežas un atbalsta vektoru ilustrācija [9]**

Atbalstu vektora mašīnas analīze atradīs līniju (vispārīgi – hiperplakni), kas novietota pēc iespējas lielāku robežu starp atbalsta vektoriem.

3.2. Neironu tīkli

Cilvēka smadzenes ir neironu tīkla arhitektūras iedvesmas avots. Cilvēka smadzeņu šūnas, ko sauc par neironiem, veido sarežģītu, savstarpēji cieši saistītu tīklu, kurā neironi sūta viens otram elektriskus signālus ar mērķi palīdzēt cilvēkiem apstrādāt informāciju. Līdzīgi mākslīgais neironu tīkls ir veidots no mākslīgiem neironiem, kas strādā kopā, lai atrisinātu problēmu [10].

Sākumā jāapskata māklīgā neironu tīkla pamatvienība - neirons.



3.4. att. **Mākslīgā neirona attēlojums**

Kā redzams attēlā attēlā 3.4., šī neirona uzbūvi raksturo tā 4 pamatelementi - ieejas signāls, svars, summēšanas funkcija, aktivizācijas funkcija un nobīde.

Ieejas signāls - tas ir neironā ienākošais signāls. Izcelsme tam var būt ārēja vai arī tas var būt cita neirona izejas signāls. Šādi ieejas signāli neironam var būt vairāki.

Svars - tā galvenā funkcija ir piešķirt lielāku nozīmi tiem ieejas signāliem, kas ir svarīgi pareizam problēmas risinājumam. Piemēram, negatīvs vārds ietekmētu noskaņojuma analīzes modeļa lēmumu vairāk nekā neitrālu vārdu pāris. Svara vērtības tiek noteiktas apmācības procesā.

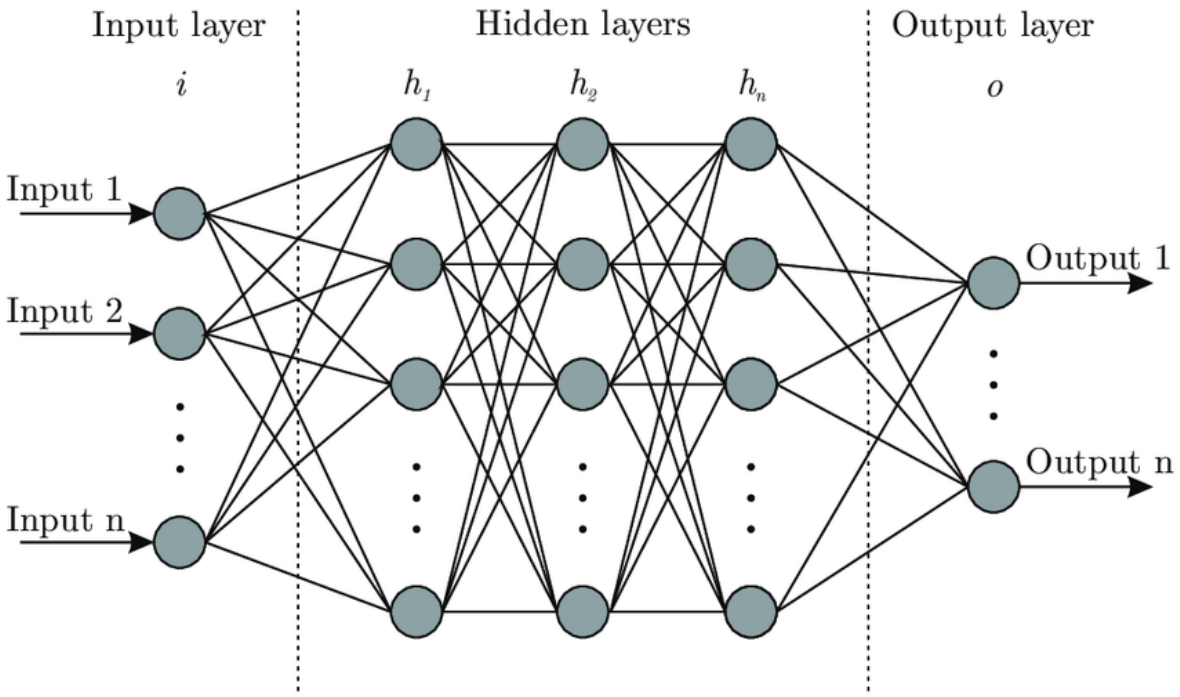
Summēšanas funkcija - šīs funkcijas mērķis ir apvienot vairākus ieejas signālus un to

svarus vienā vērtībā, ko tālāk padot aktivizācijas funkcijai.

Aktivizācijas funkcija - šī funkcija izrēķina aktivitātes stāvokli, kas bieži ir arī neirona izejas vērtība.

Papildus svars (novirze) - novirzes uzdevums ir sniegt svaru aktivizācijas funkcijas radītajai vērtībai. Tās loma ir līdzīga konstantes lomai lineārā funkcijā un, gluži kā svars, novirzes vērtība tiek noteikta apmācības procesā.

Kad vairāki neironi ir salikti kopā pēc kārtas, tie veido slāni. Vairākus slāņus apvienojot varam iegūt daudzslāņu neironu tīklu. Vienkārša daudzslāņu neironu tīkla arhitektūra aprakstāma kā savstarpēji savienoti mākslīgie neironi trīs slāņos.



3.5. att. **Mākslīgā neirona tīkla uzbūves vispārinājums [11]**

Ievades slānis

Informācija no ārpasaules caur ievades slāni nonāk mākslīgajā neironu tīklā. Ievades mezgli apstrādā datus, analizē vai klasificē tos un nodod tos nākamajam slānim.

Slēptais slānis

Slēptie slāņi izmanto ievadi no ievades slāņa vai citiem slēptiem slāņiem. Mākslīgajiem neironu tīkliem var būt liels skaits slēpto slāņu. Katrs slēptais slānis analizē iepriekšējā slāņa izvadi, apstrādā to tālāk un nodod nākamajam slānim.

Izvades slānis

Izvades slānis sniedz visu mākslīgā neironu tīkla veiktās datu apstrādes gala rezultātu.

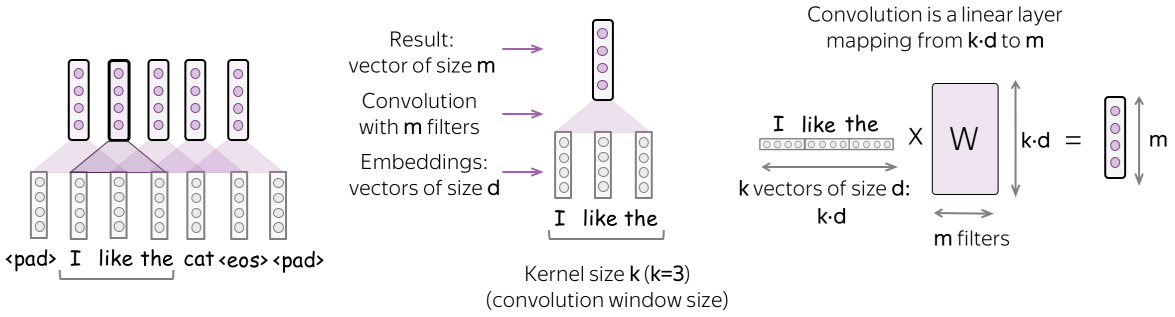
Tam var būt viens vai vairāki mezgli. Piemēram, ja mums ir binära (jā/nē) klasifikācijas problēma, izvades slānim būs viens izvades mezgls, kas dos rezultātu 1 vai 0. Tomēr, ja mums ir vairāku klašu klasifikācijas problēma, izvades slānis var sastāvēt no vairāk nekā viena izvades mezgla.

3.2.1. Konvolūcijas neironu tīkli

Konvolūcijas neironu tīklu sakotnējais mērķis ir bijis datorredzes problēmu risināšana ar dziļās mašīnmācīšanās problēmu, tomēr laika gaitā šāda neironu tīklu arhitektūra ir guvusi plašu pielietojumu arī dabīgās valodas apstrādē. Kim Yoon 2014. gadā ir pierādījis ka teikumu klasifikācijā tieši konvolūcijas neironu tīklu spēj sniegt labākus rezultātus par citiem neironu tīklu modeļiem [12]. Arī šī darba ietvaros tādēļ šāda pieeja tiks pārbaudīta. Lai gūtu priekšstatu par konvolūcijas neironu tīklu darbību sākumā tiek apskatīti visizplatītākie slāņi šāda tīkla izveidei.

Konvolūcijas slānis

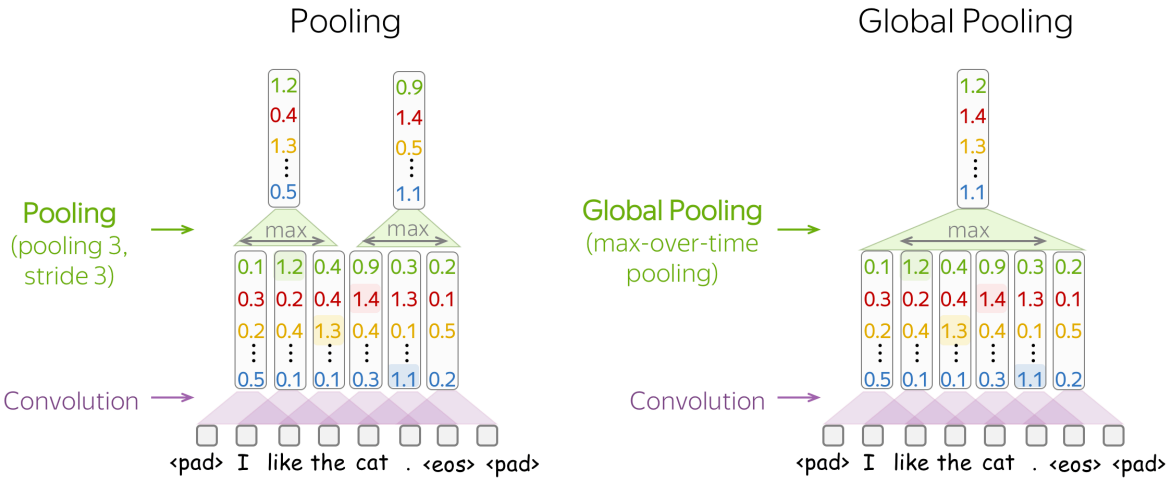
Konvolūcijas darbības princips ir filtru pielietošana, secīgi ejot cauri ievadei un meklējot dažādas pazīmes. Atkarībā no apskatāmās problēmas šie filtri var ieņemt dažādas formas – pielietojot konvolūciju uz attēliem parasti tiks pielietoti trīs dimensiju filtri. Teksta apstrādes gadījumā savukārt tiek lietoti viendimensionāli filtri ar noteiktu platumu, ko parasti sauc par kodola (angliski – kernel) izmēru. Piemēram, kodola izmērs kā 3 nozīmētu to, ka vienlaikus filtrs apskatīs trīs secīgus vārdus. Filtrs tiek pārvietots pār ievades virkni un kovolūcija tiek veikta veicot skalāro reizinājumu starp filtru un apskatīto virknes posmu. Rezultātā katrā posmā tiek iegūta vērtība, kas norāda meklētās pazīmes klātbūtni. Konvolūcijas slānis parasti satur vairākus šādus filtrus un katrs filtrs ir atbildīgs par dažādu pazīmju pārbaudi ievades datos. Apmācības laikā neironu tīkls nosaka optimālos svarus katram no šiem filtriem, norādot dažādu pazīmju nozīmi pareizas izvades iegūšanai.



3.6. att. **Konvolūcijas slānis [13]**

Apvienošanas slānis

Apvienošanas slānis veic ievades datu samazināšanas procedūru, atmetot mazsvarīgāku informāciju un saglabājot svarīgākos datu punktus, rezultātā nodrošinot efektīvāku tīkla darbību (mazāks apstrādājamo parametru skaits). Slāņa darbības principu apskatam iekš 3.7. attēla.



3.7. att. **Apvienšanas slānis [13]**

Galvenā ideja ar maksimālo apvienošanu ir atrast lielāko vērtību katrā dimensijā jeb atrast lielāko vērtību katrai pazīmei, rezultātā iegūstot vektoru, kurš norāda kādas pazīmes apskatāmajā ievades tekstā ir atrastas. Alternatīvi var izmantot arī vidējās vērtības apvienošanu, kur gala vektora vērtība tiek veidota nevis no maksimālās, bet vidējās vērtības apskatāmajā dimensijā. Kā redzams attēlā 3.7. - šādas apvienošanas operācijas varam veikt ar noteiktu soļa izmēru (no angliskā stride) vai pielietojot uz visu ievades virkni.

Atmešanas slānis

Lai izvairītos no pārmērīgas pielāgošanas apmācības laikā bieži tiek lietots arī atmešanas slānis (no angļu val. dropout). Slāņa princips ir nejaušības kārtā "atmest" daļu no iegūtajām vērtībām jeb konkrētāk - iestatīt daļu no tām kā 0, kas palīdz vispārināt modeli un labāk reaģēt uz neredzētiem ievades datiem. Šādam slānim parasti arī definējam atmešanas rādītāju (no angļu val. droput rate), kas norāda kādu daļu no neironiem mēs atmetam apmācības laikā, vērtība visbiežāk tiek izvēlēta robežās no 0.2 līdz 0.5.

Pilnīgi savienotais slānis

Konvolūcijas neironu tīklā kā pēdējais slānis ir pilnīgi savienotais slānis, kas apvieno tīkla iepriekšējo slāņu rezultātus, lai veiktu minējumu par ievades piederību kategorizācijas klasēm. Pārsvarā šajā slānī izmanto softmax aktivizācijas funkciju ar kuras palīdzību pārvēršam slānim padotās skaitliskās vērības par varbūtības vērtībām katrai klasei (robežās no 0 līdz 1) un kur visu klašu varbūtību summa būs kā 1. Ja dots ievades vektors ar rezultātiem katrai klasei kā $z = (z_1, z_2, ..., z_k)$, tad softmax funkciju i-tajam elementam varam definēt kā:

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{3.4}$$

kur:

- e ir Eilera skaitlis.
- z_i ir skaitliskais rezultāts i-tajai klasei.
- Dalītājs ir eksponentfunkciju summa visiem skaitliskajiem rezultātiem.

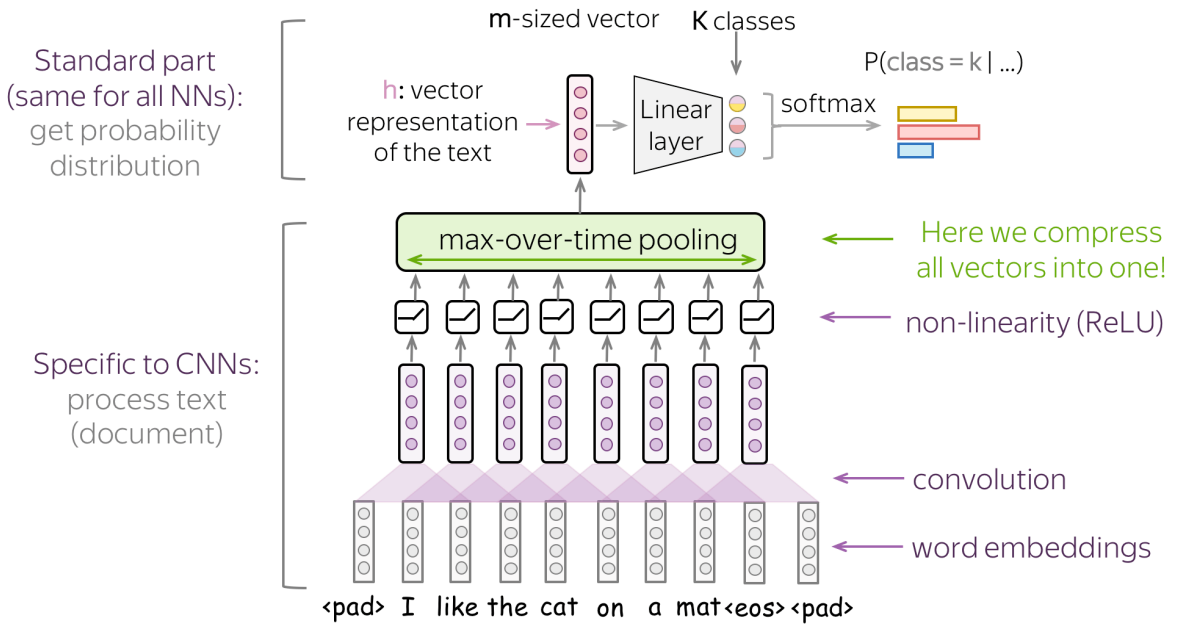
Ilustrējot ar piemēru, ja dots izvades slānis ar vektoru $z = (2.0, 1.0, 0.1)$ trīs klašu klasifikācijai, iegūtās varbūtības katrai klasei būtu:

$$\text{softmax}(z)_1 = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.65900114$$

$$\text{softmax}(z)_2 = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.24243297$$

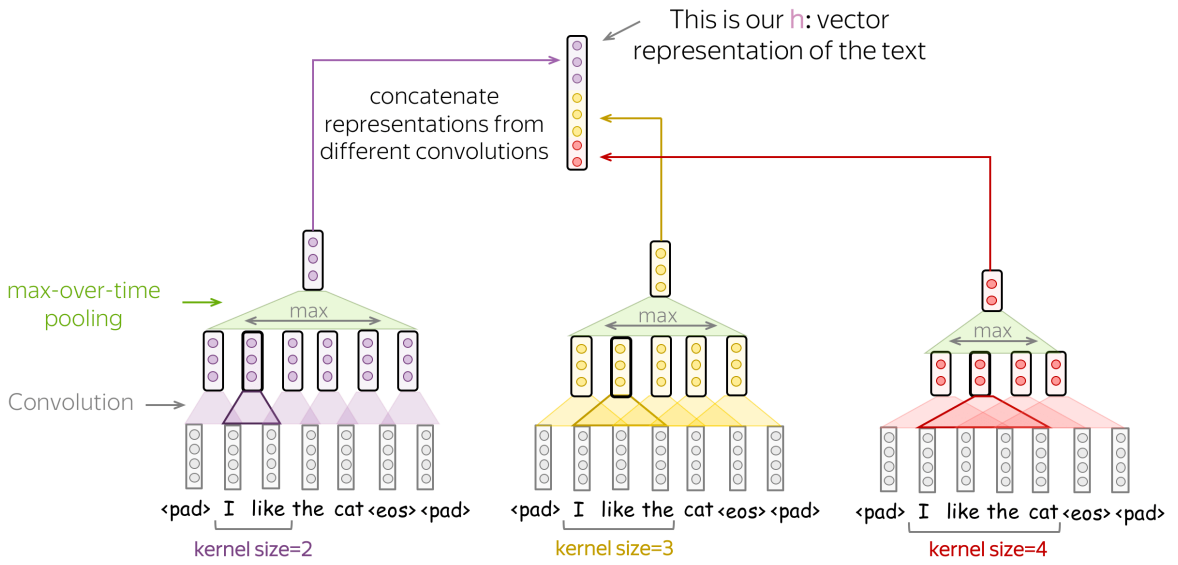
$$\text{softmax}(z)_3 = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.09856589$$

Klasi ar lielāko varbūtību pieņemam kā gala minējumu. Slāņa darbība un mijiedarbība ar citiem slāņiem apskatāma 3.8. attēlā.



3.8. att. **Konvolūcijas neironu tīkls tekstu klasifikācijai [13]**

Standarta arhitektūrai iespējams arī veikt dažādus uzlabojumus, viena no literatūrā biežāk minētajām pieejām, ko piedāvā Kim Yoon [12], ir pielietot paralēli vairākus konvolūcijas un apvienošanas slāņus ar dažādiem filtra izmēriem (pētījumā filtri izvēlēti ar izmēriem kā 2,3 un 4), vēlāk šos slāņus apvienojot pirms padošanas tālāk.



3.9. att. **Arhitektūra ar dažādu izmēru konvolūcijas slāņiem [13]**

3.2.2. Transformatori un lielie valodu modeļi

Lielie valodu modeļi (angliski - Large language models jeb LLM) kā ChatGPT pēdejos gados guvuši lielu popularitāti un tiek plaši pielietoti arī dabīgās valdoas apstrādē. Formāli lielās valodas modeļus mēs varam aprakstīt kā uz plašu valodas korpusu apmācītus modeļus,

kuri veidoti ar transformatora arhitektūras pamatiem. Transformatora arhitektūras sākums ir 2017 gadā ar publikāciju "Attention Is All You Need" [14] jeb "Uzmanība ir viss kas ir nepieciešams". Pirms tālāk tiek apskatīta transformatora arhitektūru - svarīgi ir arī apskatīt to kas tieši ir šī "uzmanība", kas parādās publikācijas virsrakstā.

Uzmanības mehānisms

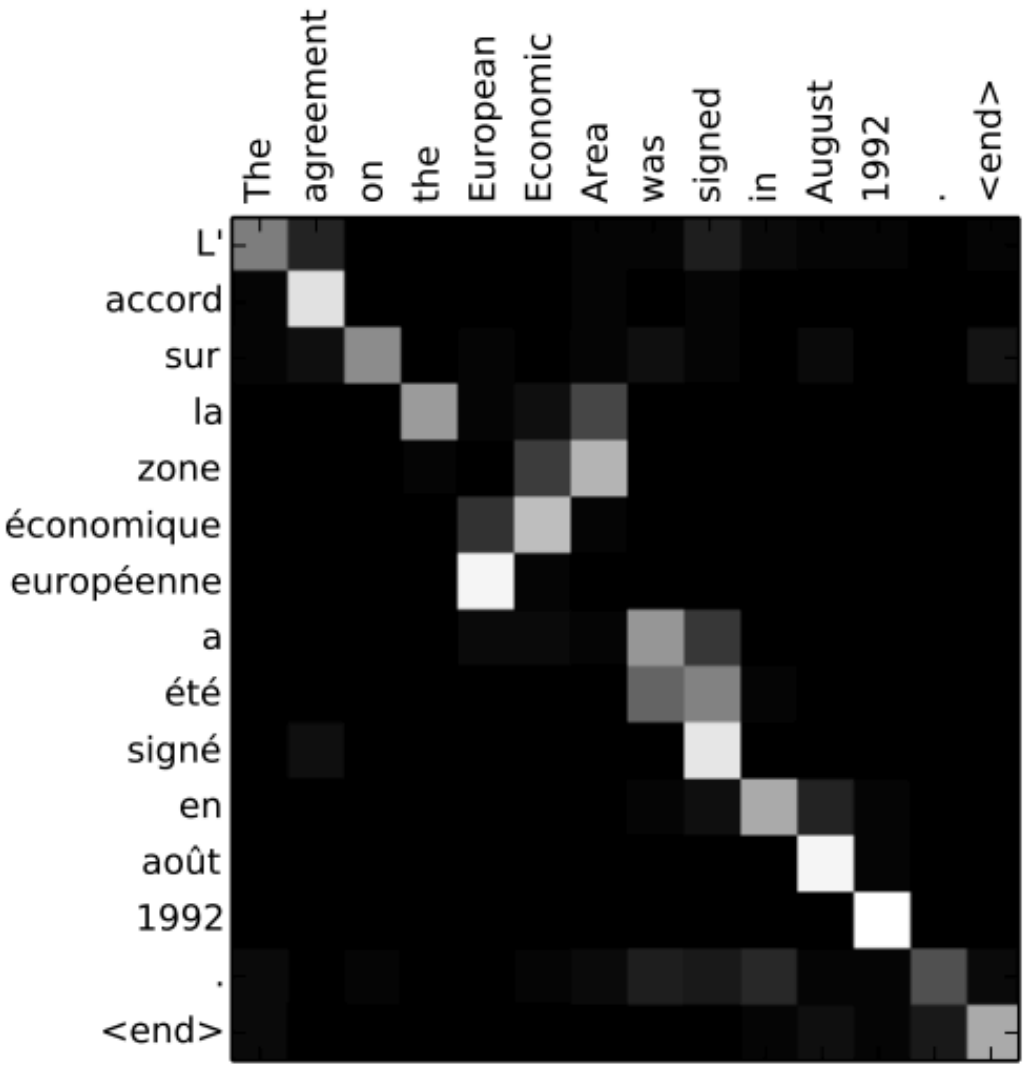
Sākotnēji uzmanības mehānisma ideja tika apskatīta tieši tulkošanas kontekstā, 2015. gada publikācijā "Neural Machine Translation by Jointly Learning to Align and Translate" [15]. Kā tieši šis uzmanības mehānisms izpaužas viegli saprast no pētījumā iekļautās problēmas apskatīšanas, kur tiek veikts tulkojums no angļu uz franču valodu.

Angļu valodas teikums: *The agreement on the European Economic Area was signed in August 1992.*

Franču valodas ekvivalentais teikums: *L'accord sur la zone économique européenne a été signé en août 1992.*

Apskatot pieejas šī teikuma tulkošanai - viens slikts veids, kā mēģināt tulkot šo teikumu, būtu vārdu pa vārdam tulkot teikumu no angļu uz franču valodu. Tas nenovestu pie kvalitatīva tulkojuma dažādu iemeslu dēļ, pirmkārt - daži vārdi franču tulkojumā ir citā secībā, no piemēra - angļu valodā Eiropas Ekonomikas zona ir "European Economic Area", bet franču valodā - "la zone économique européenne". Otrkārt - franču valoda, līdzīgi kā latviešu valoda, ir valoda ar vārdiem, kas sadalīti pēc dzimtes. Vārdiem "économique" un "européenne" ir jābūt sieviešu dzimtes formā, lai tie atbilstu sieviešu dzimtes objektam "la zone".

Uzmanība ir mehānisms, kas ļauj teksta modelim apskatīt katru vārdu oriģinālajā teikumā, kad tiek pieņemts lēmums kā veikt tulkojumu uz izvades teikumu. Zemāk apskatāma vizualizācija šim piemēram no sākotnējā uzmanības pētījuma:

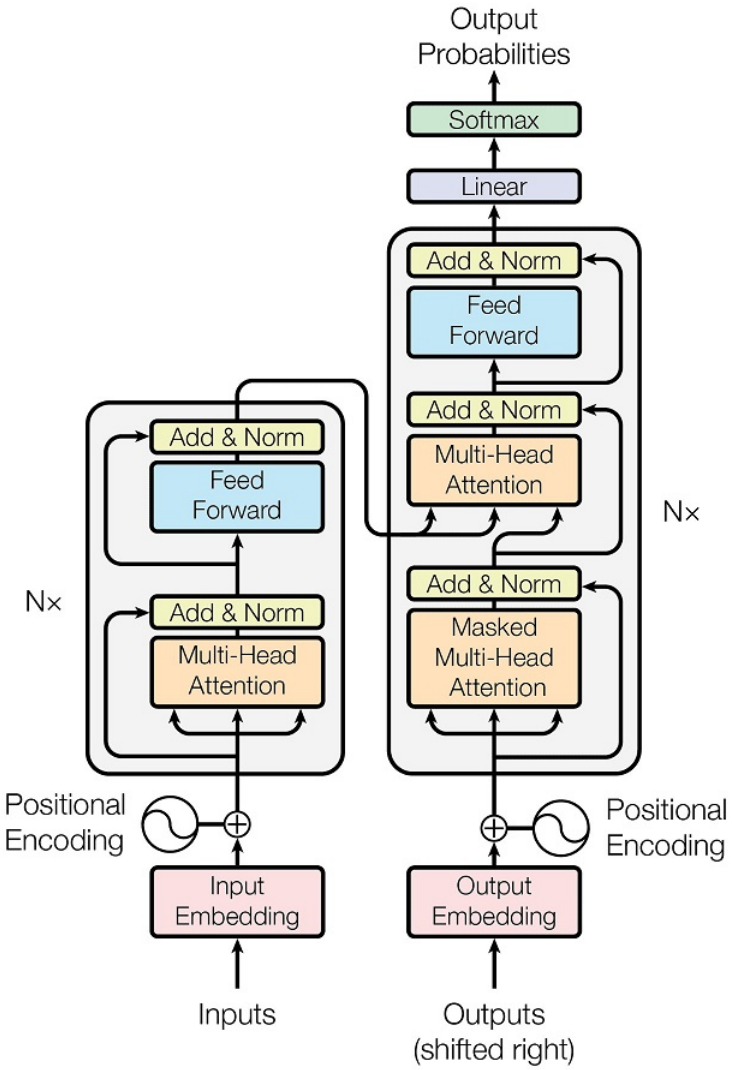


3.10. att. **Uzmanības mehānisma ilustrācija [15]**

Tā ir sava veida intensitātes karte, kas norāda kur modelis "pievērš uzmanību", kad tas izvada katru vārdu franču teikumā. Kā var redzēt, tulkojot Eiropas Ekonomiskās zonas terminu – viss termins tiek apskatīts kā saistīts kopums. Uz ko tieši modelim vērst uzmanību tiek apgūts no apmācības datiem. Veicot apmācību uz tūkstošiem franču un angļu valodas teikumu piemēriem, modelis iemācās par vārdu savstarpējo atkarību, dzimtēm, vārda formām un citiem morfoloģijas un sintakses likumiem.

Uzmanības mehānisms ir bijis ārkārtīgi noderīgs dabiskās valodas apstrādes rīks kopš tā atklāšanas 2015. gadā, taču sākotnējā formā tas tika izmantots kopā ar rekurentiem neironu tīkliem (RNT).

Transformatora arhitektūra



3.11. att. **Transformatora arhitektūra [14]**

Transformatora arhitektūra tiek apskatīta 3.11. attēlā. Ievade sākumā tiek padota iegulšanas slānim, kas strādā pēc principa kas jau apskatīts pie konvolūcijas neironu tīkliem - pārvēršot ievades tekstu vārdlietojuma kartējumā. Tālāk tiek veikta pozicionālā kodēšana, to savukārt var raksturot kā procesu ar kura palīdzību modelim tiek nodrošināta informācija par vārdu secību tekstā. Tas ir svarīgi tādēļ, ka transformatori apstrādā vārdus paralēli, ne secīgi, un citādāk modelim nebūtu iespējams gūt informāciju par vārdu secību/kontekstu kā tas iespējams citām arhitektūrām, piemēram, konvolūcijas neironu tīkliem vai rekurentajiem neironu tīkliem.

Katram vārdam tiek piešķirts unikāls pozīcijas kodēšanas vektors, attēlojot vārda pozīciju tekstā. Šie pozicionālās kodēšanas vektori tiek pievienoti ievades vārdlietojuma kar-

tējumam, papildinot katra vārda sākotnējo kartējumu, lai iekļautu informāciju par vārdus atrašanās vietu. Tas ļauj modelim arī atšķirt vārdus ar vienādu saturu, bet dažādām pozīcijām. Šo informāciju tālāk nodod pašuzmanības slānim.

Pašuzmanības slānis ir transformatora arhitektūras galvenā sastāvdaļa, nosakot kādas attiecības un atkarības starp dažādiem vārdiem ir sastopamas tekstā. Šī slāņa darbību iespējams iedalīt 4 daļās -

- Vaicājumu, atslēgu un vērtību izveide
- Uzmanības rādītāju noteikšana
- Vairāku "uzmanības galvu" pielietojums
- Izvades ģenerēšana

Vaicājumu, atslēgu un vērtību izveide

Lai saprastu, kā vārdi ir saistīti viens ar otru, pašuzmanības slānis katram vārdam no ievades izveido trīs vektorus

Vaicājums (Q): apzīmē vārdu, uz kuru pašlaik koncentrējamies. Katram vārdam ir atbilstošs vaicājuma vektors.

Atslēga (K): apzīmē visus vārdus, no kuriem vēlamies iegūt informāciju, lai palīdzētu noteikt katra vārda atbilstību vaicājumam.

Vērtība (V): satur informāciju par vārdiem, kurus mēs izgūsim, ja tiks atrasta atbilstība starp vaicājumu un atslēgu.

Uzmanības rādītāju noteikšana

Pašuzmanības mehānisms aprēķina uzmanības rādītājus veicot skalāro reizinājumu starp vaicājuma vektoriem (Q) un atslēgas vektoriem (K), rezultātu reizinot to ar $\frac{1}{\sqrt{d}}$, kur d ir vaicājuma vektora garums. Papildus tam - normalizešanai tiek arī pielietota softmax funkcija. Formulu tad var definēt sekojoši.

$$Uzmaniba(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d}}\right) \cdot \mathbf{V} \tag{3.5}$$

Šie rādītāji norāda, cik daudz katram vārdam vajadzētu pievērst uzmanību citiem vārdiem. Augstāki rādītāji nozīmē lielāku uzmanību, kas liecina, ka vārds atbilst vaicājumam.

Zemāki rādītāji liecina par zemāku atbilstību vai nozīmi vaicājumam.

Vairāku "uzmanības galvu" pielietojums

Lai uzlabotu konteksta izpratni, transformators bieži izmanto vairākas vaicājumu, atslēgu un vērtību kopas, kas pazīstamas kā "uzmanības galvas". Tas ļauj modelim vienlaikus koncentrēties uz dažādiem teksta aspektiem.

Izvades ģenerēšana no pašuzmanības slāņa

Izvade satur kontekstualizētus vārdu attēlojumus, ņemot vērā to attiecības ar citiem vārdiem. Šī izvade tagad kļūst par ievadi uz priekšu padeves slānim (no angliskā feed-forward).

Padeves slānis

Padeves slānis tiek izmantots uz pašuzmanības slāņu izvadi un tas tiek pielietots katrai pozīcijai atsevišķi. Šis slānis palīdz modelēt attālas vārdu savstarpējās atkarības. Pirmais solis padeves slānī ir lineāras transformācijas pielietošana ievadei. Tas ietver ievades reizināšanu ar apgūstamo svaru un nobīdes pievienošanu. Šī lineārā transformācija projicē ievadi citā (bieži vien augstākas dimensijas) telpā, radot pamatu sarežģītai mijiedarbībai un transformācijām. Pēc lineārās transformācijas pārveidotajai ievadei tiek piemērota nelineāra aktivizācijas funkcija, piemēram, Rectified Linear Unit (ReLU). Aktivizācijas funkcija ievieš nelinearitāti, ļaujot modelim tvert sarežģītas saiknes datos. Pēc aktivizācijas funkcijas rezultātiem tiek piemērota cita lineāra transformācija. Šī transformācija izmanto atšķirīgus svarus un nobīdes, lai vēl vairāk pielāgotu datus.

3.2.3. BERT

BERT modelis pirmo reizi publicēts 2019. gadā un autoru vārdiem BERT ir pirmā neparraudzītās apmācības pieeja ar dziļu divvirziena konkteksta izveidi dabīgās valodas priekšapstrādei [16]. Viena no BERT priekšrocībām ir tā, ka visu BERT arhitektūru un jau apmācītos modeļus Google komanda ir publiskojusi caur GitHub [https://github.com/google-research/bert], atšķirībā no citiem izplatītiem lielajiem valodas modeļiem kā GPT no OpenAI, kur kods nav publiski pieejams un kurus nevaram izmantot darbā apskatītās problēmas risināšanai.

Ar BERT mēs apmācam "valodas izpratnes" modeli uz lielapjoma teksta korpusa, piemēram, visiem Wikipedia rakstiem konkrētajā valodā, pēc tam to pielāgojot konkrētu dabīgās valodas apstrādes uzdevumu risināšanai kā tekstu klasifikācija. Nepārraudzītā apmācība ļauj mums pielietot to uz plašiem teksta korpusiem no jebkura interneta resursa. Svarīga loma ir arī kā valodas dati tiek vektorizēti, jo valodas attēlojumi var būt ar kontekstu un bez tā. Citas iepriekš populārākās vektorizācijas pieejas kā word2vec ģenerē vārdlietojuma kartējumu bez konteksta katram vārdam apmācības vārdnīcā, respketīvi, daudznozīmīgi vārdi (piemēram, komanda - ko varam uztvert kā pavēli vai cilvēku kopu sportā) pazaudē savu atšķirīgo nozīmi un patieso kontekstu. Kontekstu veidojoši modeļi savukārt ģenerē attēlojumu katram vārdam atkarībā no konteksta konkrētajā teikumā. BERT gadījumā, atšķirībā no citiem iepriekšējiem modeļiem, arī šo kontekstu mēs veidojam arī divos virzienos – apskatot vārdus gan pirms, gan pēc konkrētā vārda. Šādai priekšapstrādei BERT izmanto sekojošu pieeju - tiek maskēti 15% ievades vārdu, tālāk šādu ievades kopu apstrādājam caur dziļu divvirzienu transformatora kodētāju, pēc tam paredzam tikai maskētos vārdus. Piemēram:

Ievade: vīrietis devās uz [MASK1]. viņš nopirka [MASK2] piena.

Etiķetes: [MASK1] = veikalu; [MASK2] = litru

Papildus, lai apmācītu arī attiecības starp teikumiem, veicam apmācību uz vienkāršu uzdevumu, kurš pielietojams jebkuram valodas korpusam. Šis uzdevums ir - ja doti divi teikumi A un B, vai B ir nākamais teikums, kas seko aiz A, vai arī vienkārši nejaušs teikums no valodas korpusa? Piemērs varētu izskatīties sekojoi.

Teikums A: Vīrietis devās uz veikalu.

Teikums B: Viņš nopirka litru piena.

Pazīme: IsNextSentence

Teikums A: Vīrietis devās uz veikalu.

Teikums B: Pingvīni nelido.

Pazīme: NotNextSentence

3.3. Modeļu novērtēšana un validācija

Pēc apmācīta modeļa iegūšanas ir svarīgi novērtēt izveidotā modeļa veiktspēju un to, cik labi tas spēs veikt tekstu klasifikāciju ar jauniem datiem. Viena no izplatītākajām novērtēšanas metodēm ir metode ar noturēšanu (holdout method), kur paraugu kopa tiek sadalīta divās daļās – apmācības kopa un testa kopa. Klasifikators tad tiek apmācīts ar apmācību ko-

pu un validēts ar testa kopu. Šīs metodes mīnuss ir tas, ka apmācībai pieejama mazāka datu kopa. Cita negatīvā īpašība šai metodei ir arī tā, ka rezultāti ir atkarīgi no nevienlīdzīgā datu sadalījuma starp šīm abām kopām.

Cita metode ir nejauša paraugu atlase (Random Subsampling). Šī metode atkārto metodi ar noturēšanu vairākas reizes, lai labāk noteiktu modeļa veiktspēju, tomēr arī šai metodei piemīt pirmās metodes negatīvās īpašības. Modeļa novērtēšana ar apmācību datiem nav ieteicama, jo tādējādi notiks pārmērīga pielāgošana (overfitting) un mašīnmācīšanās modelis iegaumēs apmācības datus, bet nespēs pareizi klasificēt jaunus datus.

Visbeidzot var izmantot arī šķērsvalidāciju (cross-validation). Šī metode sadala paraugu kopu k vienādās daļās un izmanto katru no šīm daļām tieši vienu reizi priekš testēšanas. Citas, k-1, reizes katra daļa tiek izmantota apmācībai.

Klasifikācijas mēri

Klasifikācijas problēma ir bieži apskatīts mašīnmācīšanās temats un visizplatītākie mēri, ar kuriem novērtēt modeļa precizitāti ir aprakstīti zemāk.

Pārpratumu matrica

Pārpratumu matrica ļauj pārredzami attēlot modeļa precizitāti modelim ar 2 un vairāk klasēm. Matrica sastāv no 2 asīm, kur uz x ass tiek attēlotas visas klases un uz y ass tiek attēloti klases minējumi. Katra matricas šūna satur minējumu skaitu attiecīgajai klases un minētās klases kombinācijai.

Bināras klasifikācijas gadījumā pārpratumu matrica varētu būt attēlojama ar šādu tabulu:

3.1. tabula

Pārpratuma matrica

	+	-
+	PA	KA
-	KN	PN

Kur tabulas 3.1. vērtības raksturojamas šādi:

- PA - pareiza atbilsme (tabulā - gadījumi, kad '+' tiek pareizi klasificēts)
- PN - pareiza neatbilsme (tabulā - gadījumi, kad '-' tiek pareizi klasificēts)

- KA - kļūdaina atbilsme (tabulā - gadījumi, kad '-' tiek nepareizi klasificēts kā '+')
- KN - kļūdaina neatbilsme (tabulā - gadījumi, kad '+' tiek nepareizi klasificēts kā '-')

Akurātums

Akurātums ir mērs, kurš norāda cik no minējumiem ir bijuši pareizi. Tas ir pareizo minējumu dalījums ar visu minējumu skaitu.

$$Akurātums = \frac{PA + PN}{PA + PN + KA + KN} \tag{2.6}$$

Precizitāte

Precizitāte ir mērs, kas parāda pareizo pozitīvo vērtību proporciju starp modeļa kopējiem pozitīvajiem minējumiem. Tas atbild uz jautājumu "Cik no visām veiktajām pozitīvajām prognozēm bija patiesas?". Tas ir svarīgs mērs gadījumos, kad svarīgi ir novērst tieši kļūdaini pozitīvus minējumus.

$$Precizitāte = \frac{PA}{PA + KA} \tag{2.7}$$

Pārklājums

Pārklājums norāda uz to, cik labi modelis spēj atrast visas pozitīvās vērtības. Šis mērs atbild uz jautājumu "Cik no visiem datu punktiem, kas būtu jāparedz kā patiesi, tika pareizi prognozēti kā patiesi?". Tas ir svarīgs mērs gadījumos, kad svarīgi ir novērst tieši kļūdaini negatīvus minējumus.

$$Pārklājums = \frac{PA}{PA + KN} \tag{2.8}$$

F1 mērs

F1 mērs ir precizitātes un pārklājuma mēru harmoniskais vidējais.

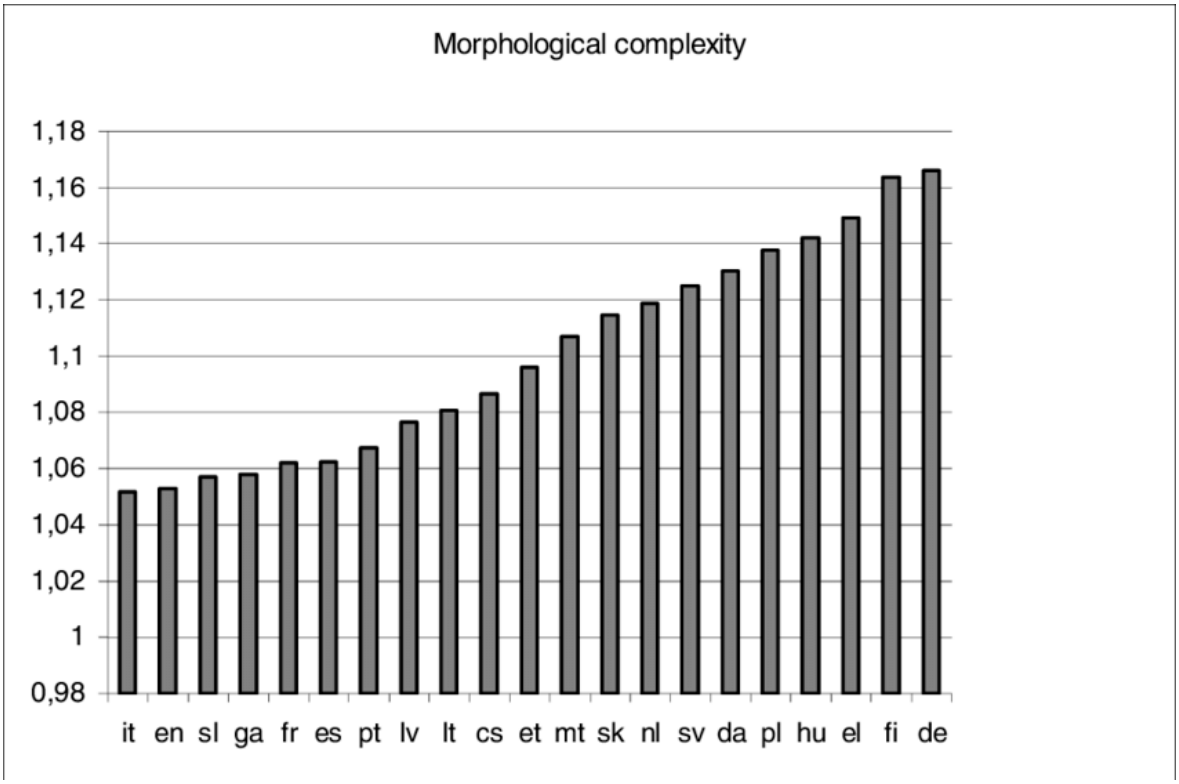
$$F1 = \frac{(2 * precizitāte * pārklājums)}{(precizitāte + pārklājums)} \tag{2.9}$$

Atšķirībā no aritmētiskas vidējās vērtības, harmoniska vidējā vērtība tiecas uz mazāko vērtību no diviem mēriem, no tā izriet, ka F1 mērs būs zems ja precizitāte vai pārklājums ir zems. Īpaši noderīgs šis mērs ir gadījumos, kad vēlamies noteikt modeļa veiktspēju datu kopā ar nevienmērīgu klašu sadalījumu.

3.4. Biežākās problēmas tekstu klasifikācijā

Morfoloģiskā un sintaktiskā sarežģītība

Liela nozīme ir konkrētai valodai, kurai veicam apstrādi. Darbā apskatam latviešu valodu un tā gan morfoloģiski, gan sintaktiski ir sarežģītāka par izplatītākām valodām kā angļu valodu [17], kurai dabīgās valodas apstrāde ir pētīta visplašāk. Morfoloģiskā sarežģītība Eiropā izplatītajām valodām apskatīta attēlā 3.12..



3.12. att. **Morfoloģiskā sarežģītība Eiropas valodām[17]**

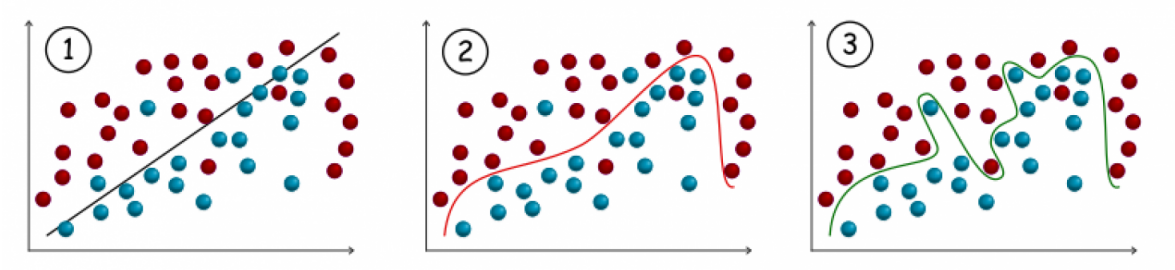
Papildus tam - dabiskā valoda var būt neskaidra, ar vārdiem un frāzēm, kuriem ir vairākas nozīmes atkarībā no konteksta. Šīs neskaidrības precīza risināšana ir izaicinājums teksta klasifikācijas modeļiem.

Tekstu nevienmērība

Teksta dati ir dažādi un atšķiras pēc garuma, struktūras un kvalitātes. Tie var ietvert rakstos raksturīgas kļūdas, slengu, saīsinājumus un plašu rakstīšanas stilu klāstu, kas padara tos grūti standartizējamus. Arī kopumā bieži apmācībai pieejamie dati nav vienmērīgi - dažādas tekstu klases var būt vai nu plašāk vai retāk izplatītas dotā datu kopā.

Pārmērīga pielāgošana

Pārmērīga pielāgošana nozīmē to, ka klasifikators ir pārāk labi modelējis apmācības datus un nedarbojas labi uz iepriekš neredzētiem datiem. Kļūdas, ko klasifikators pieļauj uz apmācības datiem sauc par apmācības kļūdām, savukārt kļūdas, kuras tiek pieļautas uz iepriekš neredzētiem datiem, sauc par vispārināšanas kļūdām. Labam modelim ir gan zems apmācības kļūdu skaits, gan zems vispārināšanas kļūdu skaits. Nepietiekama pielāgošana notiek ja modelim ir gan augsts apmācības kļūdu skaits, gan arī augsts vispārinājuma kļūdu skaits. No otras puses - pārmērīga pielāgošana notiek kad modelim ir zems apmācības kļūdu skaits, bet augsts vispārināšanas kļūdu skaits [18]. Zemāk apskatāmajā attēlā 3.13. attēloti piemēri divdimensiju klasifikācijas scenārijā.



3.13. att. **Pielāgošanas scenāriji (1 - nepietiekama, 2 - optimāla, 3 - pārmērīga)**

Visbiežāk šāda veida kļūdas rodas no apmācību datiem, kuros ir pārāk daudz ar konkrēto klasifikāciju nesaistīti dati (lieks fona "troksnis") vai arī izvēlētais apmācību datu apjoms ir pārāk mazs.

4. RĀPUĻA UN KLASIFIKĀCIJAS MODEĻU IZVEIDE

4.1. Datu izgūšana no ziņu portāliem ar rāpuli

Lai veiktu izpēti, sakumā ir nepieciešams ievākt treniņdatus / valodas korpusu, kas raksturo problēmvidi – ziņu portālu rakstus. Praktiskai rāpuļa implementācijai tika izvēlēts Python ietvars "Scrapy", ar kura palīdzību iespējams izveidot tīmekļa rāpuļus, kas pārmeklē mājaslapas un izvelk no tām datus strukturētā formā. Šis ietvars izvēlēts, jo tas ir viens no populārākajiem rīkiem šajā kategorijā un tas labi spēj apstrādāt un formatēt lielu datu apjomu. Tā kā tīmekļa rāpuļi ir jāpielāgo konkrētai mājaslapas struktūrai, lai iegūtu vēlamos datus, tika izvēlēts konkrēts portāls - delfi.lv, dēļ tā daudzveidīgā kategoriju klāsta un rakstu daudzuma. Lai palielinātu iegūstamo rakstu daudzveidību tika apsvērts pielietot rāpuli arī uz citiem ziņu portāliem, tomēr tiem visiem ir liels pārklājums savā starpā, pārpublicējot rakstus no ziņu aģentūrām kā LETA. Šāda pieeja sekojoši varētu novest pie dublikātiem datu kopā, radot iespēju vienādiem rakstiem parādīties gan apmācības, gan validācijas kopās vienlaicīgi. Darba ietvaros izveidots rāpulis, kas ievāc datus no delfi.lv portāla un saglabā tos JSON formā ar 4 pamatlaukiem – virsraksts, kategorija, saturs, hipersaite. Lai sašaurinātu problēmvidi un ierobežotu nepieciešamos resursus tika izvēlētas 10 apskatāmās kategorijas – mūzika, atpūta, kriminālziņas, finanses, tehnoloģijas, kino, literatūra, politika, sports, auto. Darbības princips rāpulim ir sekojošs:

- Rāpulim sākotnēji apskatāmās saites noradām kā 10 izvēlēto kategoriju lapas (piemēram, politikas ziņām - https://www.delfi.lv/193/politics?page=1).
- No lapām iegūstam hipersaites uz individuāliem rakstiem, ierobežojot tālāk apskatāmās saites tā, lai tās aizvien piederētu apskatāmajām kategorijām un nesaturētu nevēlamas saites (kā komentāru lapas rakstiem).
- Apstrādājam katra raksta lapu, piefiksējot hipersaiti un dažādas komponentes - virsrakstu, kategoriju, saturu, atlasot tos pēc HTML elementu atbilstības konkrētas komponentes kritērijiem.
- Saglabājam katru rakstu JSON formātā un ierakstam failā, atkārtojot procedūru līdz vairs neatrodam unikālus rakstus ko rāpulim apmeklēt.

Scrapy ietvars kopumā dod iespēju diezgan efektīvi realizēt šādu rāpuli – sākot no apskatāmo rakstu ierobežošanas līdz elementu atlasei un apstrādei. Salīdzinoši grūtāk ir tieši formalizēt kādus selektorus izvēlēties elementu atlasei, jo lapu formatējums starp dažādām kategorijām mēdz būt atšķirīgs un pat vienas kategorijas ietvaros tika novērots ka senākiem rakstiem formatējums var atšķirties no jaunāko rakstu formatējuma. Papildus tam arī ne visi raksti ir derīgi datu ieguvei, piemēram sastopami raksti kas satur tikai foto un video galerijas ar minimāliem aprakstiem, sastopami arī maksas raksti ar tikai vienu publiski pieejamu rindkopu.

Izgūta raksta piemēru JSON formātā iespējams apskatīt 1. pielikumā. Rezultātā tika ievākti 13762 raksti ar sadalījumu pa kategorijām kāds redzams tabulā 4.1.

4.1. tabula

Ievākto rakstu sadalījums pa kategorijām

Kategorija	Raksti
Mūzika	1722
Atpūta	1523
Kriminālziņas	1517
Finanses	1363
Tehnoloģijas	1333
Kino	1282
Literatūra	1277
Politika	1263
Sports	1250
Auto	1232

Ar rāpuli izgūtie teksti papildus tika manuāli pārbaudīti un attīrīti no nevēlamiem datiem – iegulti koda fragmenti audio /video atskaņotājiem, hipersaites. Ievācot rakstus novērots, ka bez rakstu satura atšķiras arī vidējie rakstu garumi katrā kategorijā. Piemēram atpūtas ziņām raksturīgi gari raksti ar vidēji vairāk nekā 662 vārdiem, savukārt auto, sporta un kriminālziņām – krietni īsāki raksti (īpaši auto ziņām ar vidējo rakstu garumu ap 227 vārdiem). Šāda atšķirība garumos varētu atstāt ietekmi uz konkrētu kategoriju klasifikāciju akurātumu. Rakstu iedalījumu garumos sīkāk iespējams apskatīt 2. pielikumā.

4.2. Klasisko mašīnmācīšanās algortimu implementācija

Izmantotie rīki

Viena no Python valodas populārākajām mašīnmācīšanās bibliotēkām, kas palīdz risināt problēmas kā klasteru veidošana, regresija, klasifikācija, dimensiju skaita samazināšana, ir 'sckit-learn'. Autors ir izvēlējies lietot šo bibliotēku lai atvieglotu plaši lietotu klasifikācijas algoritmu implementāciju (Naivā Bejesa metode, loģistiskā regresija, lēmumu koki, atbalsta vektoru mašīnas).

Svarīga loma valodas apstrādē arī ir ievades tokenizācijai, tās veikšanai iespējams izmantot Python bibliotēkas kā NLTK un spaCy. Tieši šīs divas bibliotēkas ir populārākās un spēj tikt galā ar biežām tokenizācijas problēmām kā saīsinājumu, pieturzīmju un simbolu atdalīšana. Darba ietvaros tokenizācijas nolūkiem tika izvēlēts pielietot spaCy.

4.2.1. Priekšapstrāde un vektorizācija

Tokenizācija un tekstu attīrīšana

Teksta apstrāde tiek sākta ar teksta tokenizāciju, izmantojot spaCy bibliotēkā iebūvētās metodes. Kad teksts ir sadalīts vārdos – apstrāde tiek turpināta ar visu vārdu pārvēršanu formā ar visiem mazajiem burtiem, tiek izņemtas pieturzīmes un stopvārdi.

Konkrētāk apskatot stopvārdu atmešanu - dažādās Python bibliotēkās ir iekļauti saraksti ar stopvārdiem izplatītām valodām, tomēr latviešu valodai šāds sarakts jādefinē neatkarīgi. Tika veikta izpēte par to vai šāds saraksts jau ir publiski pieejams un kā viens no populārākajiem atrasts 'stopwords-lv' repozitorijs iekš github. Lai gan tas ir izmantojams kā labs pamats un uzskaita palīgvārdus (saikļus, prievārdus, partikulas), trūkst citas svarīgas morfoloģiskās grupas kā vietniekvārdi (attieksmes vietniekvārdi – kurš, kura u.c., norādāmie vietniekvārdi – šis, šī, tas, tā, viņš u.c, kā arī locījumi šiem vārdiem), jo arī šo vārdu esamība neraksturo teksta fragmenta jēgu vai piederību kādai kategorijai. Darba ietvaros izveidots uzlabots stopvārdu saraksts latviešu valodai, kas labāk spētu veikt vārdu filtrēšanas soli teksta priekšasptrādē, un pielietots uz apmācības datiem.

Vektorizācija

Pirms algoritmu pielietošanas nepieciešams datus vektorizēt. Darba ietvaros uz katru no algoritmiem tika pārbaudītas dažādas vektorizācijas pieejas – vārdu maiss, bigrammu

maiss, TF-IDF un vārdlietojuma kartējumi ar FastText, pielietojot katru no tām uz rāpuļa izgūtās datu kopas. Tika arī veikti eksperimenti ar word2vec vārdlietojuma kartējuma pieeju, tomēr Fasttext pielietošana sniedza labākus sākotnējos rezultātus, arī citi pētījumi par vārdlietojuma kartējuma pieejām latviešu valodas tekstiem [19] norāda FastText kā pieeju ar labāko veiktspēju, salīdzinot ar word2vec, ngram2vec, SSG (Structured Skip-Gram).

4.2.2. Algoritmu implementācijas

Datu līdzsvarošana un sadale

Svarīgs priekšnosacījums labai klasifikācijas modeļa apmācībai ir izvairīšanās no nevienmērīga klašu sadalījuma datu kopā. Ar rāpuli tika mēģināts izgūt samērā līdzīgu rakstu skaitu pa kategorijām, tomēr atšķirības eksistē, piemēram, klasei "Mūzika" rezultātā tika izgūti 1722 raksti, bet auto ziņām – 1232 raksti. Pirms apmācības visām kategorijām saglabājam 1200 rakstus, pārējos atmetot. Rezultātā iegūstam 12 000 rakstus, kurus tālāk sadalām – 80% apmācībai (9600 raksti), 20% validācijai (2400 raksti).

Atbalsta vektora mašīnas

Atbalsta vektora mašīnas apmācības algoritms ir implementēts ar scikit-learn komponenti LinearSVC (sklearn.svm.LinearSVC) ar dažādām vektorizācijas metodēm. Novērtēto akurātuma mēru ar dažādām vekorizācijas pieejām varam apskatīt 4.2. tabulā.

4.2. tabula

Akurātuma mēri pielietojot AVM

Vārdu maiss	Bigrammu maiss	TF-IDF	Fasttext
0.9696	0.9679	0.9738	0.9567

Naivā Bajesa metode

Naivā Bajesa apmācības algoritms ir implementēts ar scikit-learn komponenti MultinomialNB (sklearn.naive_bayes.MultinomialNB) ar dažādām vektorizācijas metodēm. Novērtēto akurātuma mēru ar dažādām vekorizācijas pieejām varam apskatīt 4.3. tabulā.

4.3. tabula

Akurātuma mēri pielietojot Naivā Bajesa metodi

Vārdu maiss	Bigrammu maiss	TF-IDF	Fasttext
0.9629	0.9558	0.9617	0.85

Loģistiskā regresija

Loģistiskās regresijas apmācības algoritms ir implementēts ar scikit-learn komponenti LogisticRegression (sklearn.linear_model.LogisticRegression) ar dažādām vektorizācijas metodēm. Novērtēto akurātuma mēru ar dažādām vekorizācijas pieejām varam apskatīt 4.4. tabulā.

4.4. tabula

Akurātuma mēri pielietojot loģistisko regresiju

Vārdu maiss	Bigrammu maiss	TF-IDF	Fasttext
0.9663	0.9663	0.9663	0.9579

Lēmumu koki

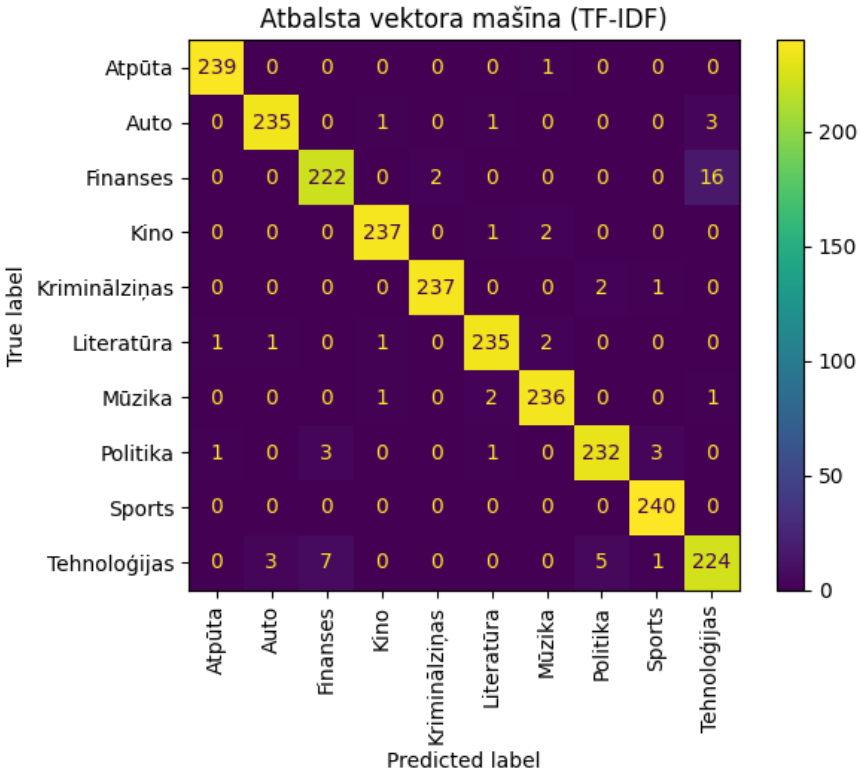
Lēmumu koku pmācības algoritms ir implementēts ar scikit-learn komponenti DecisionTreeClassifier (sklearn.tree.DecisionTreeClassifier) ar dažādām vektorizācijas metodēm. Novērtēto akurātuma mēru ar dažādām vekorizācijas pieejām varam apskatīt 4.5.tabulā.

4.5. tabula

Akurātuma mēri pielietojot lēmumu kokus

Vārdu maiss	Bigrammu maiss	TF-IDF	Fasttext
0.8096	0.8196	0.7975	0.7408

Vislabākais sasniegtais akurātums iegūts ar atbalsta vektora mašīnām un TF-IDF vektorizāciju - **0.9738**, šai pieejai varam vizualizēt kategorizāciju ar pārpratuma matricu. Kā redzams attēlā 4.1. – desmit kategoriju klasifikācija tiek veikta ļoti precīzi un biežākā kļūda ir nepareizi klasificētas finanšu ziņas, klasificējot tās kā tehnoloģiju ziņas.



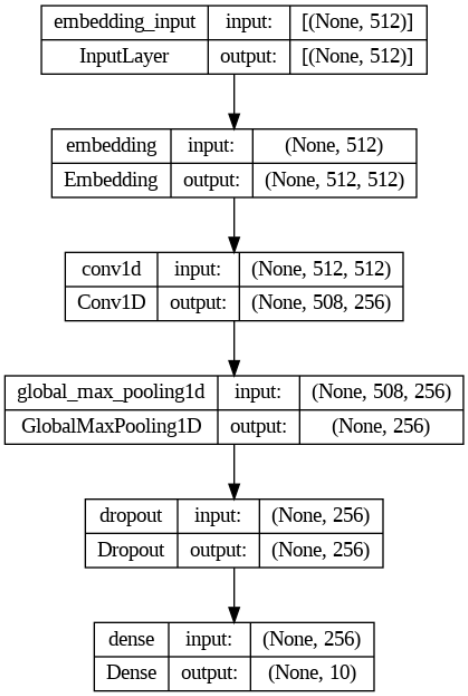
4.1. att. **Atbalsta vektora mašīnas pārpratuma matrica**

4.3. Neironu tīklu implementācija

Darba ietvaros tika izvērtētas un implementētas dažādas neironu tīklu arhitektūras. Tā kā tika nolemts pārbaudīt plašu spektru ar dažādiem apmācības algoritmiem – tika izvēlēts Python ietvars Keras, balstīts uz Tensorflow bāzes, kas palīdz ātri un sintaktiski vienkārši prototipēt dažādus modeļus. Neironu tīklu apmācība, vismaz sākotnēji, tika veikta uz autora datora, izmantojot viduvējas veiktspējas procesoru (AMD Ryzen 5700X) un 16GB RAM. Apmācība vienkāršākiem tīkliem šādi veicama diezgan efektīvi, tomēr sarežģītākas arhitektūras ar BiLSTM slāņiem vai BERT modeļa pielāgošana kļūst gan laikietilpīgāka (12+ stundu apmācība), gan ierobežota operatīvās atmiņas resursu dēļ. Šo iemeslu dēļ darba gaitā modeļu apmācība turpināta ar Google TPU v2, kas izstrādāts tieši neironu tīklu apmācībai un nodrošina krietni ātrakus apmācības laikus.

4.3.1. Konvolūcijas neironu tīkli

Konvolūcijas tīklu implementācijai tiek veidots Keras modelis ar secīgiem slāņiem – iegulšanas, viendimensiju konvolūcijas ar ReLU aktivizācijas funkciju, apvienošanas (globālā maksimuma), atmešanas, pilnīgi savienotais slānis gala klasifikācijas veikšanai. Ilustrēti ar papildus informāciju par izvēlētajām slāņu dimensijām varam apskatīt modeli 4.2. attēlā.



4.2. att. **Konvolūcijas neironu tīkla uzbūve**

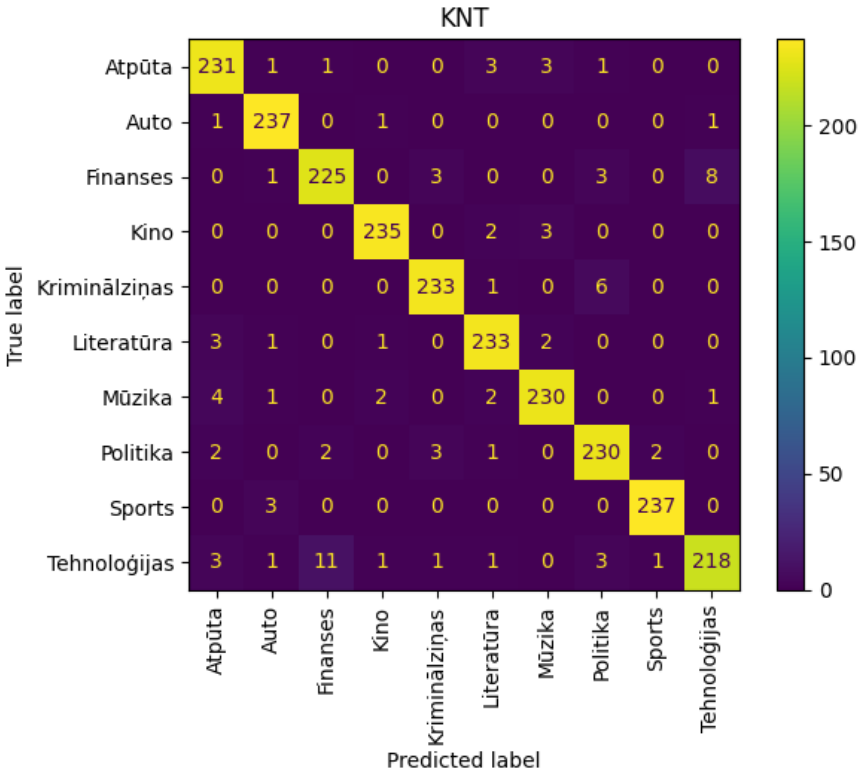
Apmācību tiek veikta ar partijas izmēru kā 32, turpinot apmācību 10 epohos, saglabājot modeli posmos ar mazāko validācijas zuduma (validation loss) vērtību, mēģinot izvairīties no pārmērīgas pielāgošanas. Uz tekstiem pirms apmācības tiek veikta priekapstrāde - simbolu un stopvārdu atmešana. Akurātuma un zuduma evolūciju pa epohiem iespējams apskatīt 4.4. attēlā.

4.6. tabula

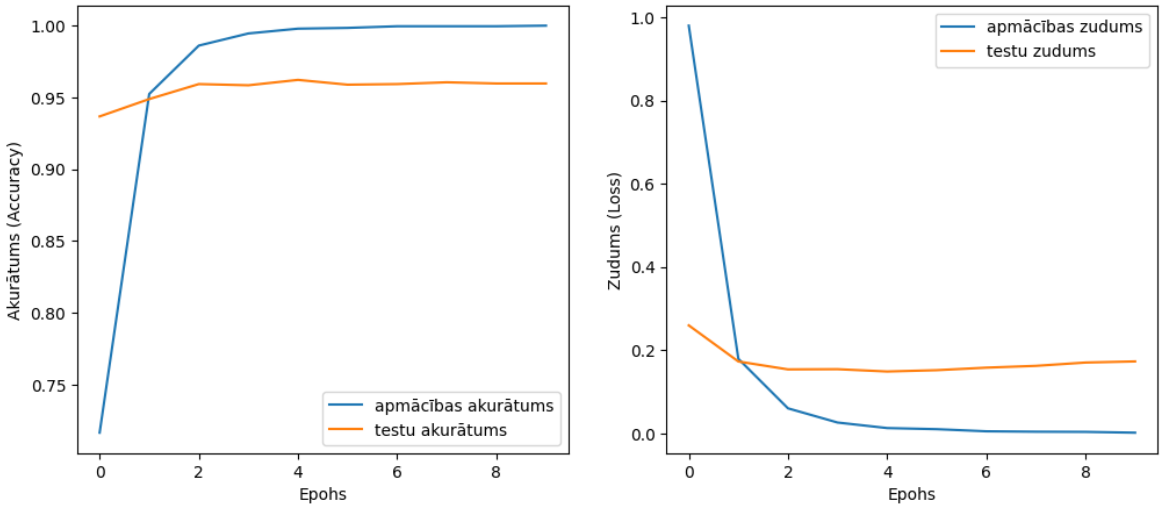
Veiktspējas mēri pielietojot konvolūcijas neironu tīklu

Akurātums	F1
0.9621	0.9620

Pielietojot attēlā 4.2. ilustrēto konvolūcijas neironu tīkla uzbūvi tiek iegūti rezultāti kā 4.6. tabulā. Pārpratuma matricu iespējams apskatīt attēlā 4.3., kur varam novērot līdzīgu kļūdu izplatību pa klasēm kā iepriekš apskatītos modeļos, arī ar šo pieeju visgrūtāk modelim ir atķirt tieši tehnoloģiju un finanšu ziņas.

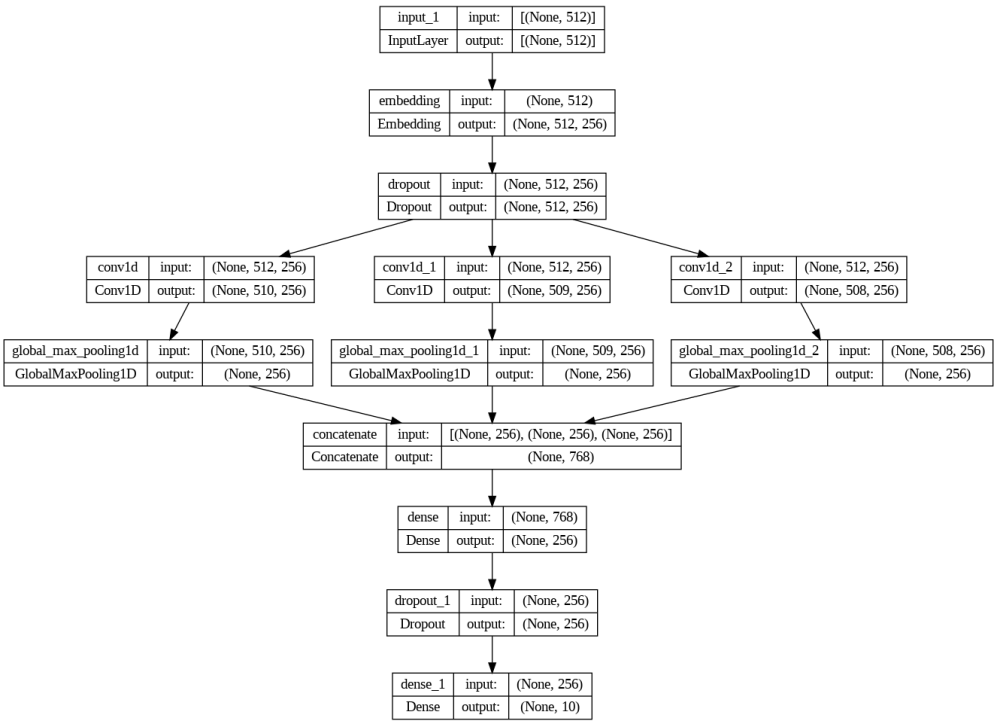


4.3. att. **Konvolūcijas neironu tīkli - pārpratuma matrica**



4.4. att. **Konvolūcijas neironu tīkli - novērtējums pa apmācības posmiem**

Mēģinot uzlabot modeļa veiktspēju tiek pielietota plaši citēta konvolūcijas neironu tīklu arhitektūra tekstu apstrādei no Kim Yoon [12]. Šīs arhitektūras galvenā ideja ir ieviest vairākus paralēlu konvolūcijas un apvienošanas slāņus ar dažādiem filtra izmēriem, vēlāk to rezultātus apkopojot. Galvenais iemesls – palīdzēt neironu tīklam apgūt dažādas iezīmes un likumsakarības tekstos, piemēram, mazāki filtri uztver vārdu kombinācijas, plašāki filtri – sarežģītākas valodas struktūras kā frāzes. Arī šāda pieeja tiek implementēta ar filtra izmēriem kā 4,5 un 6. Šī modeļa uzbūvi var apskatīt 4.5. attēlā.



4.5. att. **Konvolūcijas neironu tīkla uzbūve - paralēla konvolūcija**

Modelis kurš apmācīts at augstākminēto arhitektūru nesiendza veiktspējas uzlabojumus, kā redzams 4.7.tabulā.

4.7. tabula

Veiktspējas mēri - konvolūcijas neironu tīkls ar parelēlu konvolūciju

Akurātums	F1
0.9517	0.9516

4.3.2. BERT

Sākotnējais BERT modelis, kuru Google izstrādātāji jau bija apmācījuši un publicējuši, latviešu valodas ziņu apstrādei nav pielietojams, jo modelis apmācīts tikai uz angļu valodas tekstiem. Lai gan tikusi publicēta arī BERT uzbūves arhitektūra un modeli iespējams pašrocīgi apmācīt uz latviešu valodas tekstiem – šī darba ietvaros tas netiek veikts, jo šādai apmācībai nepieciešami ievērojami apmācības resursi ilgstošā laika posmā, kas autoram nav pieejami. Publiski gan palaik ir pieejams BERT modelis, kur apmācība uz latviešu valodas tekstiem jau ir veikta, to apmācījuši LU Matemātikas un informātikas institūta pētnieki Artūrs Znotiņš un Guntis Barzdiņš, modeli nosaucot par LVBERT. Tas ir apmācīts uz latviešu Wikipedia ierakstiem, tekstiem no valodas korpusa LVK2018, dažādiem ziņu rakstiem un to komentāriem, kas kopā veido 500 miljonu lielu tokenu kopu apmācībai [20]. Salīdzinot ar sākotnējo BERT modeli, LVBERT apmācības tokenu skaits ir vairāk nekā sešas reizes mazāks.

LVBERT, gluži kā citi BERT balstītie modeļi ir vispārināti un nav piemēroti tikai vienai problēmai – tos nepieciešams pielāgot konkrētai problēmai pirms to pielietošanas, šajā gadījumā - papildus tika pievienoti slāņi tieši klasifikācijas risināšanai. Papildus tekstu priekšapstrāde netika pielietota, jo tādējādi varam pazaudēt daļu no konteksta un saiknēm starp vārdiem, kas tieši ir viena no BERT modeļu spēcīgākajām pusēm.

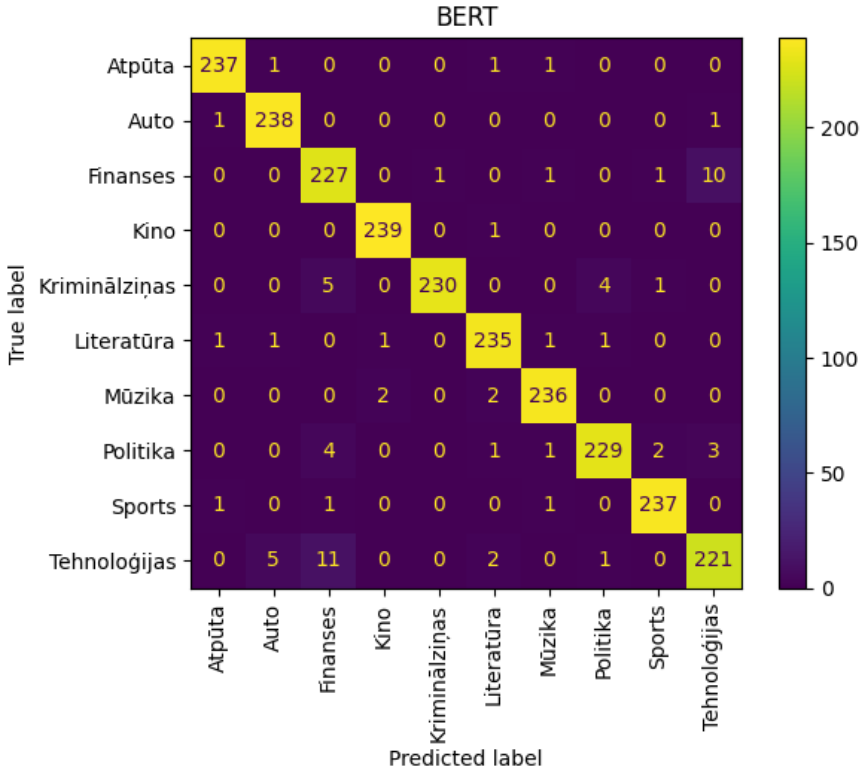
Modeļa pielāgošana veikta 5 epohos ar apmācības ātrumu kā 2e-5 un partijas izmēru kā 32, šai kombinācijai pēc eksperimentiem ar dažādām parametru vērtībām uzrādot labākos rezultātus. Sākotnējā BERT publikācijā [16] pielāgošana dažādiem GLUE uzdevumiem veikta ar partijas izmēru kā 32, 3 epohos un apmācības ātrumiem 5e-5, 4e-5, 3e-5 un 2e-5 (izvēloties labāko rezultējošo modeli katram uzdevumam). Apskatot arī citas publikācijās [21], kur apskatīta BERT pielāgošana teksta klasifikācijai, apmācības ātrums kā 2e-5 uzrāda labākos rezultātus, kamēr lielāki ātrumi kā 4e-4 noved pie "katastrofālas aizmiršanas" problēmas, kur iepriekš apgūtā informācija tiek zaudēta veicot jaunu apmācību.

4.8. tabula

Veiktspējas mēri pielietojot BERT

Akurātums	F1
0.9704	0.9704

Rezultātā iegūtā modeļa raksturojošie mēri skatāmi 4.8. tabulā. Lai gan salīdzinot ar citiem neironu tīklu modeļiem tiek sasniegts rezultātu uzlabojums, tas nepārsniedz atbalsta vektora mašīnu akurātumu. Angļu valodas lielie valodu modeļi pārsvarā gūst krietni labākus rezultātus klasifikācijā, salīdzinot ar vienkāršākām metodēm. Daļējs izskaidrojums LVBERT veiktspējai varētu būt ievērojami mazākais tokenu skaits apmācības posmā, salīdzinot ar angļu valodas modeli.



4.6. att. **BERT - pārpratuma matrica**

Papildus apskatot pārpratuma matricu attēlā 4.6., BERT modelim var novērot līdzīgu klasifikācijas kļūdu sadalījums kā jau iepriekš apskatītajiem modeļiem.

SECINĀJUMI UN PRIEKŠLIKUMI

Darba procesā tika noskaidrots ka ziņu klasifikācijā pielietot mašīnmācīšanās algoritmus ir noderīgi, jo iespējams veikt šo klasifikāciju ļoti precīzi, augstāko akurātuma rādītāju 0.9738 sasniedzot ar atbalsta vektora mašīnas algoritmu un TF-IDF pielietojumu pazīmju ģenerēšanā.

Novērots arī tas, ka ne visas kategorijas ir vienlīdz viegli klasificēt. Piemēram – finanšu un tehnoloģiju ziņas visiem modeļiem bija grūti klasificēt. Tas izskaidrojams ar saturisku pārklājumu starp tēmām (tehnoloģiju jaunumi un tehnoloģijas uzņēmumu finanšu jaunumi par peļņu/investīcijām).

Lai gan izpētīt un implementēt neironu tīklus un dažādas to arhitekrūtas autora ieskatā bija jēgpilni, izveidotie modeļi nespēja sasniegt augstāku precizitāti.

Autora ieskatā publiskas ziņu rakstu datu kopas ir ļoti noderīgas mašīnmācīšanās eksperimentos, piemēram, angļu valodā ziņu kopas kā "20 Newsgroup" tiek plaši pielietotas un pat iekļautas populārās bibliotēkās kā sckit-learn. Latviešu valodā šādas publiskas datu kopas netika atrastas un rakstu kopas izveide ne vienmēr ir triviāls uzdevums. Autora ievākto datu kopu publiskojot iespējama tālāka tās pielietošana citu autoru darbos.

Teksta priekšapstrāde latviešu valodā ir ierobežota morfoloģisko rīku pieejamības dēļ. Zināmus uzlabojumus priekšapstrādē autoram ir izdevies panākt ar paplašināta stopvārdu saraksta izveidi.

Priekšlikumi:

Autora ieskatā noderīgi būtu uzlabot modeļu apmācību ar papildus tekstu morfoloģisko apstrādi (piemēram, lemmatizāciju). Šāda apstrāde angļu un citu izplatītāku valodas tekstiem ir pieejama dažādās Python bibliotēkās, diemžēl latviešu valodas tekstiem nav tāda atbalsta. Nepieciešams veikt papildus darbu šādu rīku izstrādei.

Nepieciešama tālāka izpēte par neironu tīkliem un iespējām sasniegt augstāku precizitāti – iespējams autora izvēlētie slāņi un paramatri tīkla izveidei nebija optimāli.

Lai izvairītos no klasifikācijas problēmām kategorijās ar saturisku pārklājumu – noderīgi būtu implementēt kategorizāciju, kas spētu piešķirt tekstiem vairāk par vienas klases atbilstību.

IZMANTOTĀS LITERATŪRAS UN AVOTU SARAKSTS

[1] Tom Mitchell. *The Discipline of Machine Learning*. Carnegie Mellon University, 2006.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[3] Miltiadis Kandias, Vasilis Stavrou, Nick Bozovic, and Dimitris Gritzalis. Proactive insider threat detection through social media: The youtube case. pages 261 – 266, 11 2013.

[4] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2008.

[5] Prabhakar Manning, Christopher D.and Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[6] Baeldung. Dimensionality of word embeddings. https://www.baeldung.com/cs/dimensionality-word-embeddings, 2023. [Tiešsaistē; Skatīts Okt. 14, 2023].

[7] Ian H.Witten, Eibe Frank, and Mark A.Hall. *Data Mining: Practical Machine Learning Tools and Techniques(Third Edition)*. Morgan Kaufmann, third edition edition, 2011.

[8] Tom Mitchell. *Machine Learning*. McGraw - Hill Education, 1997.

[9] Corinna Cortes and Vladimir Vapnik. Support - vector networks. *Machine Learning*, 20(3):273–297, 1995.

[10] AWS. What is a neural network? https://aws.amazon.com/what-is/neural-network/, 2023. [Tiešsaistē; Skatīts Okt. 24, 2023].

[11] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 11 2017.

[12] Yoon Kim. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 08 2014.

[13] Lena Voita. Text classification. https://lena-voita.github.io/nlp_course/text_classification.html, 2024. [Tiešsaistē; Skatīts Jan. 29, 2024].

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[17] Kimmo Kettunen, Markus Sadeniemi, Tiina Lindh-Knuutila, and Timo Honkela. Analysis of eu languages through text compression. pages 99–109, 01 2006.

[18] Pang Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.

[19] Rolands Laucis and Gints Jēkabsons. Evaluation of word embedding models in latvian nlp tasks based on publicly available corpora. *Applied Computer Systems*, 26(2):132–138, 2021.

[20] Artūrs Znotiņš and Guntis Barzdins. *LVBERT: Transformer-Based Model for Latvian Language Understanding*. 09 2020.

[21] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2020.

GALVOJUMS

Ar šo es, Matīss Kalniņš, galvoju, ka šis bakalaura darbs ir manis paša patstāvīgi izpildīts oriģināls darbs. Visi informācijas avoti, kā arī no tiem ņemtie dati un definējumi ir norādīti darbā. Šis darbs tādā vai citādā veidā nav iesniegts nevienai citai pārbaudījumu komisijai un nav nekur publicēts.

Esmu informēts (-a), ka mans bakalaura darbs tiks ievietots un apstrādāts Vienotajā datorizētajā plaģiāta kontroles sistēmā plaģiāta kontroles nolūkos.

202___.gada _____._____ _____

Es, Matīss Kalniņš, atļauju Ventspils Augstskolai savu bakalaura darbu bez atlīdzības ievietot un uzglabāt Latvijas Nacionālās bibliotēkas pārvaldītā datortīklā Academia (www.academia.lndb.lv), kurā tie ir pieejami gan bibliotēkas lietotājiem, gan globālajā tīmeklī tādā veidā, ka ikviens tiem var piekļūt individuāli izraudzītā laikā, individuāli izraudzītā vietā.

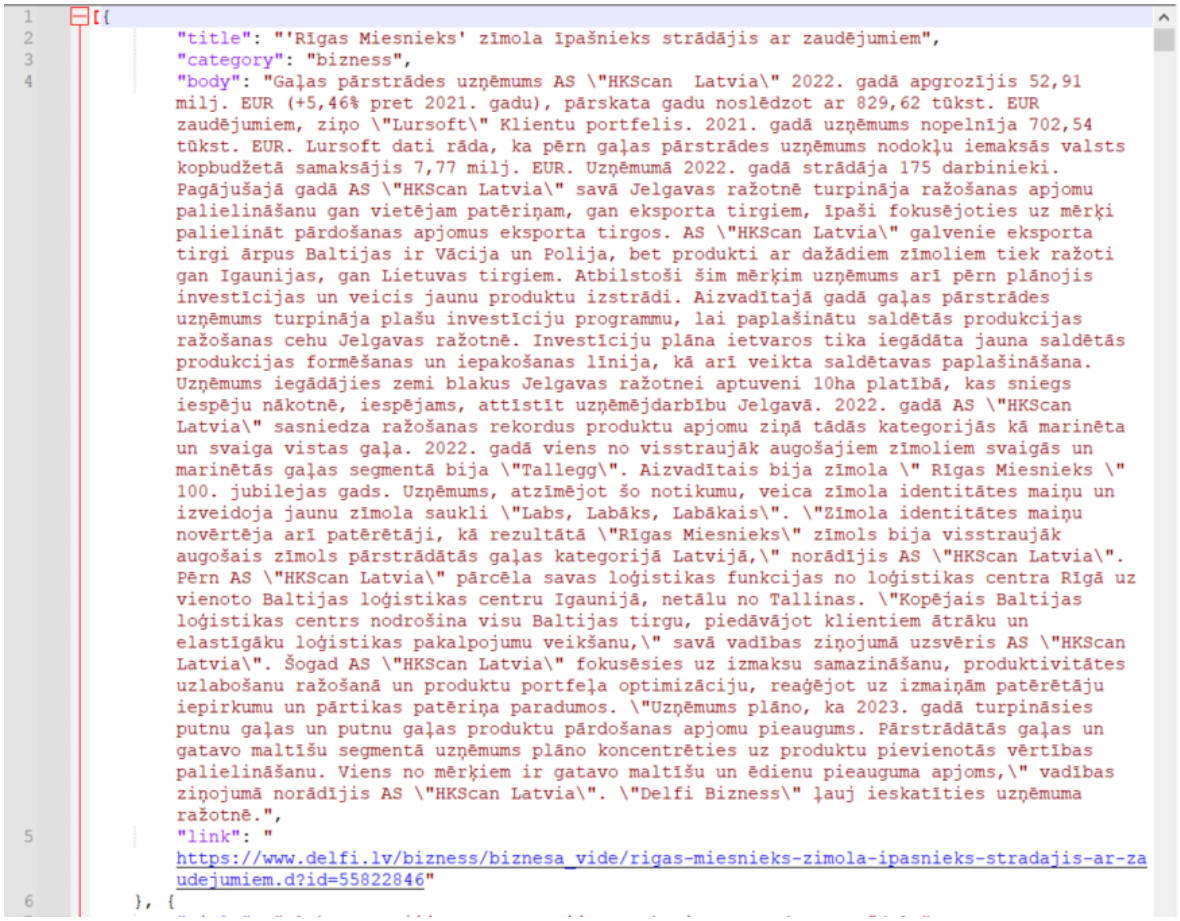
Piekrītu _____

Nepiekrītu _____

202___.gada _____._____

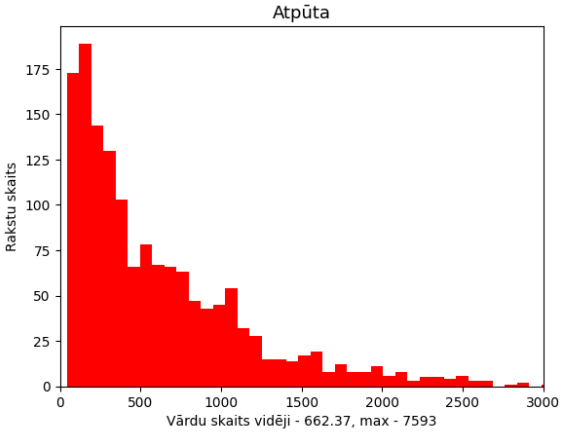
PIELIKUMS

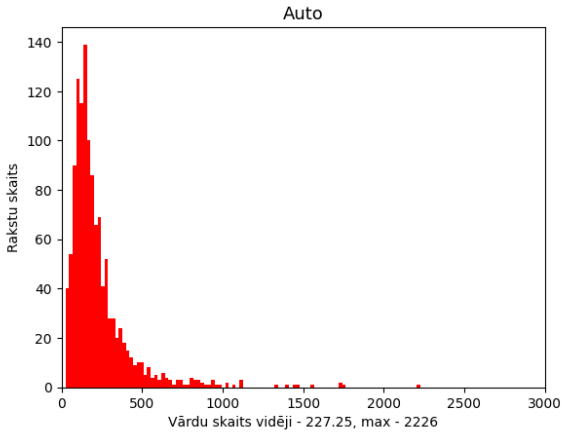
1. pielikums. Ar rāpuļa palīdzību izgūta raksta piemērs

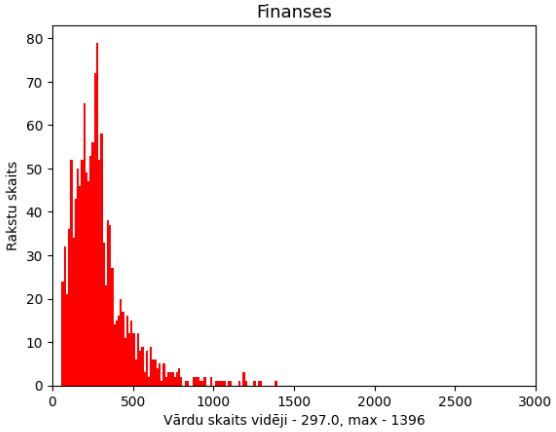


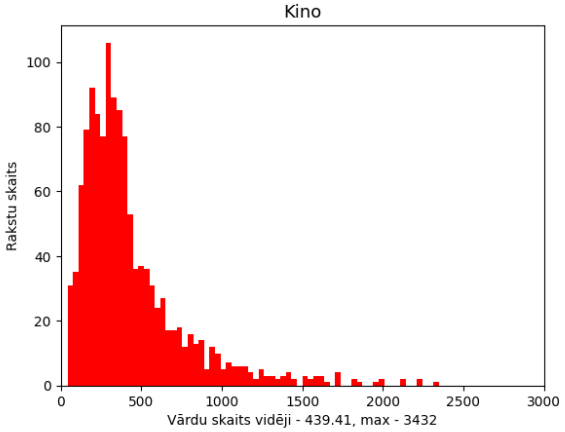

```
[{
    "title": "'Rīgas Miesnieks' zīmola īpašnieks strādājis ar zaudējumiem",
    "category": "bizness",
    "body": "Gaļas pārstrādes uzņēmums AS \"HKScan  Latvia\" 2022. gadā apgrozījis 52,91
    milj. EUR (+5,46% pret 2021. gadu), pārskata gadu noslēdzot ar 829,62 tūkst. EUR
    zaudējumiem, ziņo \"Lursoft\" Klientu portfelis. 2021. gadā uzņēmums nopelnīja 702,54
    tūkst. EUR. Lursoft dati rāda, ka pērn gaļas pārstrādes uzņēmums nodokļu iemaksās valsts
    kopbudžetā samaksājis 7,77 milj. EUR. Uzņēmumā 2022. gadā strādāja 175 darbinieki.
    Pagājušajā gadā AS \"HKScan Latvia\" savā Jelgavas ražotnē turpināja ražošanas apjomu
    palielināšanu gan vietējam patēriņam, gan eksporta tirgiem, īpaši fokusējoties uz mērķi
    palielināt pārdošanas apjomus eksporta tirgos. AS \"HKScan Latvia\" galvenie eksporta
    tirgi ārpus Baltijas ir Vācija un Polija, bet produkti ar dažādiem zīmoliem tiek ražoti
    gan Igaunijas, gan Lietuvas tirgiem. Atbilstoši šim mērķim uzņēmums arī pērn plānojis
    investīcijas un veicis jaunu produktu izstrādi. Aizvadītajā gadā gaļas pārstrādes
    uzņēmums turpināja plašu investīciju programmu, lai paplašinātu saldētās produkcijas
    ražošanas cehu Jelgavas ražotnē. Investīciju plāna ietvaros tika iegādāta jauna saldētās
    produkcijas formēšanas un iepakošanas līnija, kā arī veikta saldētavas paplašināšana.
    Uzņēmums iegādājies zemi blakus Jelgavas ražotnei aptuveni 10ha platībā, kas sniegs
    iespēju nākotnē, iespējams, attīstīt uzņēmējdarbību Jelgavā. 2022. gadā AS \"HKScan
    Latvia\" sasniedza ražošanas rekordus produktu apjomu ziņā tādās kategorijās kā marinēta
    un svaiga vistas gaļa. 2022. gadā viens no visstraujāk augošajiem zīmoliem svaigās un
    marinētās gaļas segmentā bija \"Tallegg\". Aizvadītais bija zīmola \" Rīgas Miesnieks \"
    100. jubilejas gads. Uzņēmums, atzīmējot šo notikumu, veica zīmola identitātes maiņu un
    izveidoja jaunu zīmola saukli \"Labs, Labāks, Labākais\". \"Zīmola identitātes maiņu
    novērtēja arī patērētāji, kā rezultātā \"Rīgas Miesnieks\" zīmols bija visstraujāk
    augošais zīmols pārstrādātās gaļas kategorijā Latvijā,\" norādījis AS \"HKScan Latvia\".
    Pērn AS \"HKScan Latvia\" pārcēla savas loģistikas funkcijas no loģistikas centra Rīgā uz
    vienoto Baltijas loģistikas centru Igaunijā, netālu no Tallinas. \"Kopējais Baltijas
    loģistikas centrs nodrošina visu Baltijas tirgu, piedāvājot klientiem ātrāku un
    elastīgāku loģistikas pakalpojumu veikšanu,\" savā vadības ziņojumā uzsvēris AS \"HKScan
    Latvia\". Šogad AS \"HKScan Latvia\" fokusēsies uz izmaksu samazināšanu, produktivitātes
    uzlabošanu ražošanā un produktu portfeļa optimizāciju, reaģējot uz izmaiņām patērētāju
    iepirkumu un pārtikas patēriņa paradumos. \"Uzņēmums plāno, ka 2023. gadā turpināsies
    putnu gaļas un putnu gaļas produktu pārdošanas apjomu pieaugums. Pārstrādātās gaļas un
    gatavo maltīšu segmentā uzņēmums plāno koncentrēties uz produktu pievienotās vērtības
    palielināšanu. Viens no mērķiem ir gatavo maltīšu un ēdienu pieauguma apjoms,\" vadības
    ziņojumā norādījis AS \"HKScan Latvia\". \"Delfi Bizness\" ļauj ieskatīties uzņēmuma
    ražotnē.",
    "link": "
    https://www.delfi.lv/bizness/biznesa_vide/rigas-miesnieks-zimola-ipasnieks-stradajis-ar-za
    udejumiem.d?id=55822846"
}, {
```

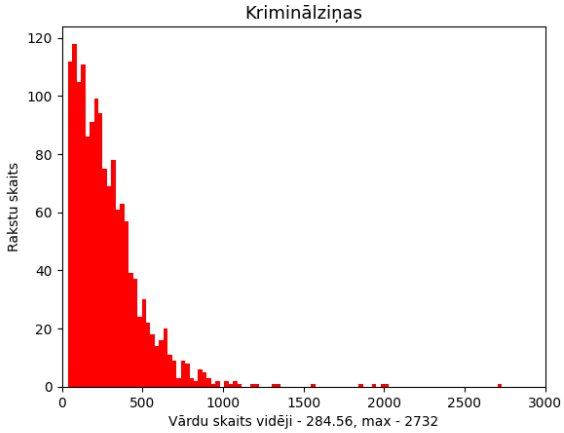
2. pielikums. Klasificējamo kategoriju rakstu garumi

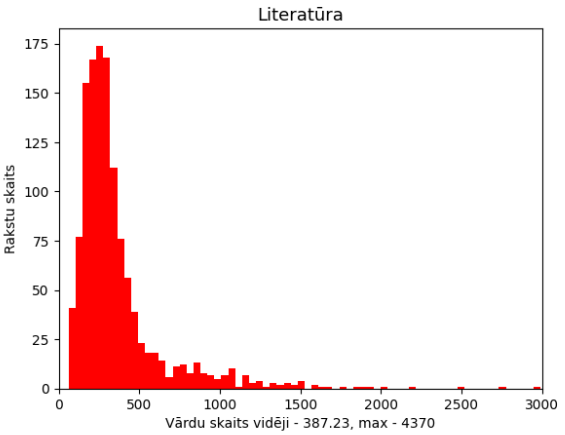


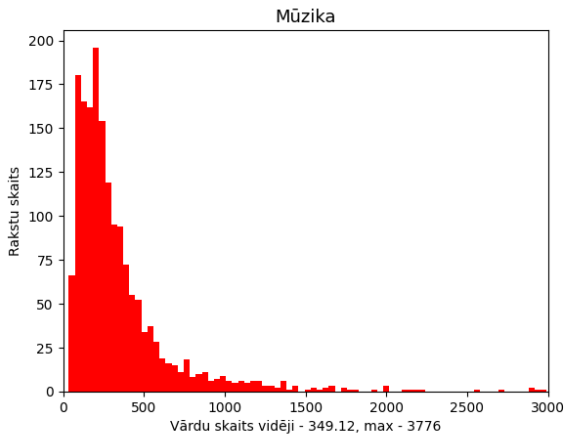


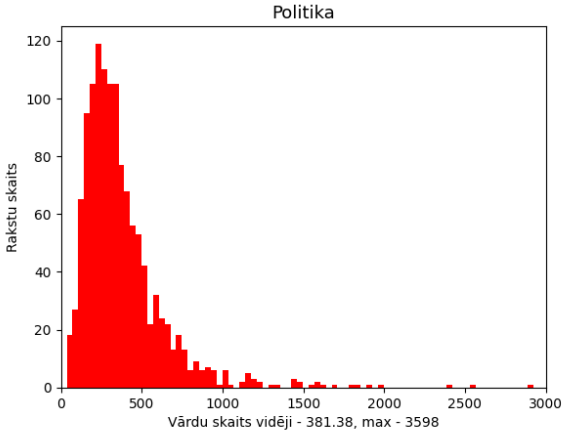


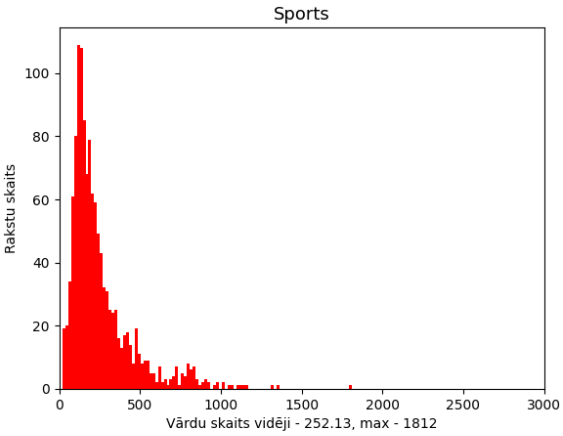


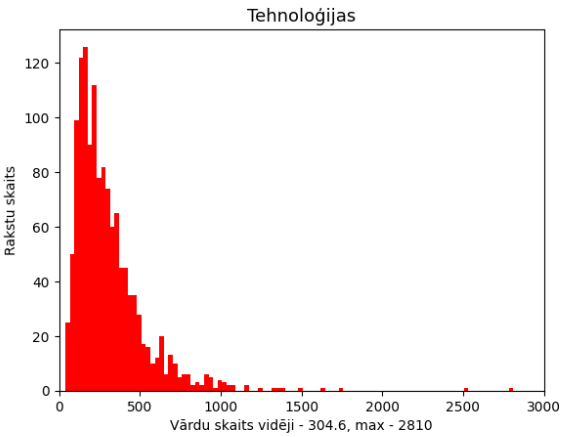





Mūzika
Rakstu skaits
Vārdu skaits vidēji - 349.12, max - 3776


Politika
Rakstu skaits
Vārdu skaits vidēji - 381.38, max - 3598


Sports
Rakstu skaits
Vārdu skaits vidēji - 252.13, max - 1812


Tehnoloģijas
Rakstu skaits
Vārdu skaits vidēji - 304.6, max - 2810