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 Blūms

_____

(paraksts)

Recenzents:

_____

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

_____

(paraksts)

Ventspils, 2024

Saturs

Anotācija **4**

Abstract **5**

Izmantotie saīsinājumi un termini **6**

Ievads **7**

1. Mašīmācīšanās valodas apstrādē **9**

1.1. Dabiskās valodas apstrāde 9

1.2. Mašīnmācīšanās 10

1.3. Tekstu klasifikācija 11

2. Pazīmju ģenerēšana **12**

2.1. Tekstu priekšapstrāde 12

2.2. Vektorizācija 13

3. Mašīnmācīšanās algoritmi **17**

3.1. Klasiskie algoritmi tekstu klasifikācijai 17

3.2. Neironu tīkli 20

3.2.1. Izplatītāko arhitektūru salīdzinājums 23

3.2.2. Konvolūcijas neironu tīkli 24

3.2.3. Rekurentie neironu tīkli 28

3.2.4. Transformatori un lielie valodu modeļi 32

3.2.5. BERT 36

3.3. Modeļu novērtēšana un validācija 38

3.4. Biežākās problēmas tekstu klasifikācijā 40

4. Rāpuļa un klasifikācijas modeļu izveide **42**

4.1. Datu izgūšana no ziņu portāliem ar rāpuli 42

4.2. Klasisko mašīnmācīšanās algortimu implementācija 44
4.2.1. Priekšapstrāde un vektorizācija 45
4.2.2. Algoritmu implementācijas 46
4.3. Neironu tīklu implementācija 50
4.3.1. Konvolūcijas neironu tīkli 50
4.3.2. LSTM neironu tīkli 54
4.3.3. BERT 57

Secinājumi un priekšlikumi 59

Izmantotās literatūras un avotu saraksts 63

Pielikumi 64
1. Darba koda repozitoriji 64
2. Ar rāpuļa palīdzību izgūta raksta piemērs 65
3. Loģistiskās regresijas, KNT un LSTM modeļu salīdzinājums - Xiang Zhang 66
4. Loģistiskās regresijas, KNT un LSTM modeļu salīdzinājums - Mathieu Cliche 67
5. Klasificējamo kategoriju rakstu garumi 68

Galvojums 70

ANOTĀCIJA

Darba nosaukums:	Ziņu portālu rakstu klasifikācija (ar mašīnmācīšanās algoritmiem)
Darba autors:	Matīss Kalniņš
Darba vadītājs:	Mg.sc.comp. Agris Blūms
Darba apjoms:	70. lpp, 16 tabulas, 29 attēli, 17 formulas, 31 bibliogrāfiskās norādes, 5 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, kā scikit-learn un Tensorflow, kas palīdz risināt mašīnmācīšanās problēmas, un 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 Blūms
The volume of the work:	70. pages, 16 tables, 29 images, 17 equations, 31 literature sources, 5 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 (*Natural language processing*) – Dabisko valodu apstrāde

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

KNT (*Convolutional neural network*) – konvolūcijas neironu tīkls

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

TF-IDF (*Term Frequency - Inverse Document Frequency*) – Terminu biežums - inversais dokumentu biežums

Epoha (*epoch*) – Apmācības periods. Apmācības procesa daļa, kurā neironu tīkls tieši vienu reizi tiek apmācīts uz visiem apmācības piemēriem

PA (*true positive*) – pareiza atbilsme

PN (*true negative*) – pareiza neatbilsme

KA (*false positive*) – kļūdaina atbilsme

KN (*false negative*) – kļūdaina neatbilsme

IEVADS

Mūsdienās internets ir kļuvis par galveno informācijas avotu lielai daļai cilvēku, kuri ikdienā ar dažādu mediju palīdzību 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, mainīts kategoriju iedalījums, aktuālas kļūst jaunas tēmas u.t.t. Lai gan arī šādos gadījumos klasifikāciju iespējams veikt 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 neeksistē, 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"[1], 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 - tie 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 [2]. 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 latviešu valodā. Lai sasniegtu šo mērķi, tiek izvirzīti tālāk norādītie 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 nākamo 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

Dabiskā valodas apstrāde jeb NLP apvieno datorlingvistiku - uz likumiem balstītu cilvēka valodas modelēšanu - ar statistikas un mašīnmācīšanās modeļiem, lai datori un digitālās ierīces varētu atpazīt, saprast un ģenerēt tekstu un runu [3].

Cilvēku valodai ir sarežģīta uzbūve, kuru dažkārt ir grūti aprakstīt. Par piemēru ņemot problēmu par nodoma izprašanu - valodas stili kā sarkasms, homonīmi, idiomas, metaforas, dažādas sintaktiskās konstrukcijas, tie visi apgrūtina šādu uzdevumu. Lai palīdzētu datoram izprast cilvēku valodu, tiek veikti dažādi dabiskās valodas apstrādes pamatuzdevumi, ieskaitot arī tālāk norādītos.

- Balss atpazīšana: balss datu pārveide teksta datos. Cilvēku runas veids (dažādas valodas, akcenti, runas temps, intonācijas, nepareiza gramatika, neskaidra izruna) un tā daudzveidība padara šo par īpaši sarežģītu problēmu
- Nosaukto entītiju atpazīšana: nosaukumu atpazīšana un iedalīšana kategorijās, piemēram, personvārdos, datumos, atrašanās vietās
- Semantiskā analīze: vārdiem ar daudzām nozīmēm nepieciešams analizēt kontekstu, lai spētu noteikt vārda nozīmi konkrētajā lietojumā
- Lemmatizācija: vārdu pārveidošana pamatformā
- Tokenizācija: teksta sadalīšanas process vārdos vai frāzēs
- Morfoloģiskā marķēšana: marķējumu piešķiršana vārda daļām (saknes, priedēkļi galotnes u.c. daļas)
- Runas daļu marķēšana: marķējumu piešķiršana vārdiem, atkarībā no to vārdšķiras (lietvārds, darbības vārds, saiklis u.c.)
- Sintakses parsēšana: teikumu gramatiskās struktūras analīze

Pielietojot daļu no šīm komponentēm iespējami sarežģītāki pielietojumi teksta apstrādei – klasifikācijai, tekstu kopsavilkumu veidošanai, noskaņojuma analīzei, mašīntulkošanai, čatbotu izveidei.

Sākotnējās DVA sistēmas balstījās uz manuāli izstrādātiem noteikumiem, taču šīs sistēmas ir ierobežotas ar savu nespēju apstrādāt cilvēku valodu 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?” [4].

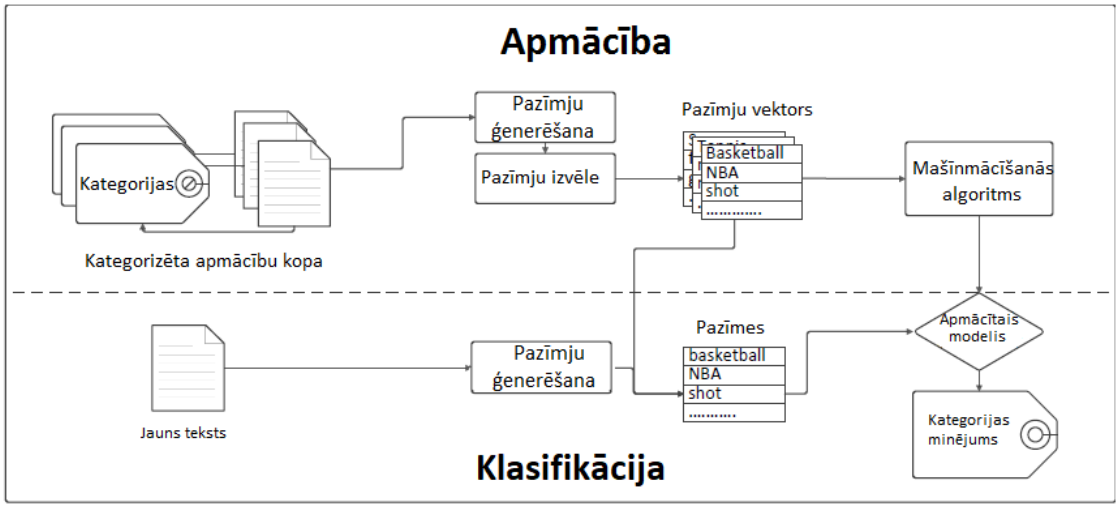
Induktīvā mācīšanās ir mašīnmācīšanās apakšnozare, kas specifiski nodarbojas ar modeļu mācīšanos no novērojumiem. Šajā nozarē mācīšanās uzdevumi bieži tiek raksturoti, pamatojoties uz atgriezenisko saiti, kas tiek sniegta apmācības veicējam [5] tālāk norādītos veidos.

- 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: Īpašs uzraudzītās mācīšanās gadījums, kur apmācības laikā 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. Šī darba ietvaros tiek apskatīta tieši klasifikācija.

1.3. Tekstu klasifikācija

Tekstu klasifikācijas mērķis ir tekstu piesaiste konkrētai klasei, balstoties uz teksta saturu. Šāda klasifikācija ir pielietojama ziņu portālos un citur, kur nepieciešams kategorizēt lielu datu daudzumu. Mašīnmācīšanās algoritmi ļauj rast risinājumu dažādām 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 [6].



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 marķējumu. 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 klases.

2. PAZĪMJU ĢENERĒŠANA

2.1. Tekstu priekšapstrāde

Pirms iespējams izveidot klasifikācijas modeli, nepieciešams veikt teksta priekšapstrādi, lai dati būtu pielietojami tālākajā 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 viennozīmīgi, 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 klasei. Svarīgi arī ņemt vērā, ka liels individuālu vārdu atkārtojums dažādos locījumos palielina klasifikācijai nepieciešamo laiku un resursus. Sakņu atdalīšana un lemmatizācija risina šo problēmu, pārvēršot vārdus dažādos locījumos vienādā formā. Sakņu atdalīšanas gadījumā tiek mēģināts nodzēst vārdu galotnes / piedēkļus (saulei - saul, saulīte - saul), balstoties uz valodas likumsakarībām vārda daļu atpazīšanai. Lemmatizācijas gadījumā vārdi tiek pārveidoti to pamatformā (saulei - saule, saulīte - saule), šādai apstrādes metodei nepieciešama krietni sarežģītāka morfoloģijas izpratne.

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 ietver saikļus (un, bet, vai u.c.), prievārdus (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 var izmantot mašīnmācīšanai. Šis solis ir būtisks, jo mašīnmācīšanās algoritmi strādā ar skaitļiem, bet teksta dati ir paši par sevi neskaitliski. Teksta vektorizācijai ir vairākas metodes, populārākās no tām apskatītas zemāk.

Vārdu maiss

Vārdu maiss – tā ir nesakārtota vārdu kopa, kur vārdu secība tiek ignorēta, saglabājot tikai vārdu biežumu dokumentā [7]. Teksts tiek pārveidots vektorā, kur katram unikālam vārdam dokumentā tiek piešķirts indekss, kā vērtību indeksā norādot konkrētā vārda biežumu.

Lai ilustrētu šādu pieeju, pielietojam to uz diviem teikumiem:

1. Kaķis devās medīt peles
2. Kaķis ātri skrēja, peles ātri bēga

Rezultātā iegūstot vektorus teikumiem kā 2.1. tabulā.

2.1. tabula

Vārdu maisa vektori diviem teikumiem

	kaķis	devās	medīt	peles	ātri	skrēja	bēga
1. teikums	1	1	1	1	0	0	0
2. teikums	1	0	0	1	2	1	1

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.

Plašāka konteksta izgūšanai no teksta, iespējams veidot vektorus, kur katram indeksam atbilst vairāku secīgu vārdu kombinācija jeb n-gramma. Piemēram, izvēloties n-grammas ar garumu 2 (bigrammas), iepriekš apskatīto teikumu vektori izskatītos kā 2.2. tabulā.

2.2. tabula

Bigrammu maisa vektori diviem teikumiem

	kaķis devās	devās medīt	medīt peles	kaķis ātri	ātri skrēja	skrēja peles	peles ātri	ātri bēga
1. teikums	1	1	1	0	0	0	0	0
2. teikums	0	0	0	1	1	1	1	1

TF-IDF

Terminu biežums - inversais dokumentu biežums (angliski, saīsināti - TF-IDF) ir vektorizācijas paveids ko izmanto, lai novērtētu termina (vārda) svarīgumu tekstā attiecībā uz dokumentu kopu (korpusu) [8].

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.

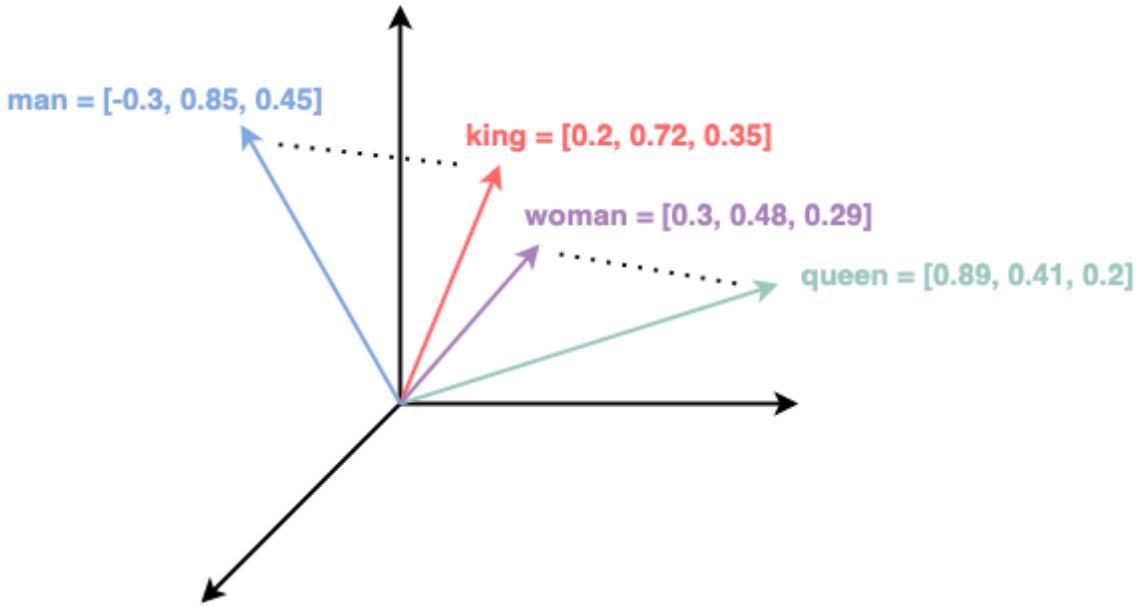
TF-IDF beigu rezultāts ir termina biežuma (TF) un inversā dokumenta biežuma (IDF) reizinājums:

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

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 atrodas tuvu viens otram vektoru telpā. Piemēram, labi apmācītā vārdlietojuma kartējuma modelī vārdi ”karalis” un ”karaliene” tiek attēloti kā vektori, kas ir tuvu viens otram, norādot to semantisko līdzību [9]. Šāda pieeja tiek vizualizēta 2.1. attēlā.



2.1. att. **Vārdlietojuma kartējums vektora telpā [9]**

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 teorēma 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 tas bieži darbojas praksē.

Loģistiskā regresija

Loģistiskā regresija modelē varbūtību, ka dokuments pieder konkrētai klasei, izmantojot loģistisko (sigmoidālo) funkciju [10], 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:

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

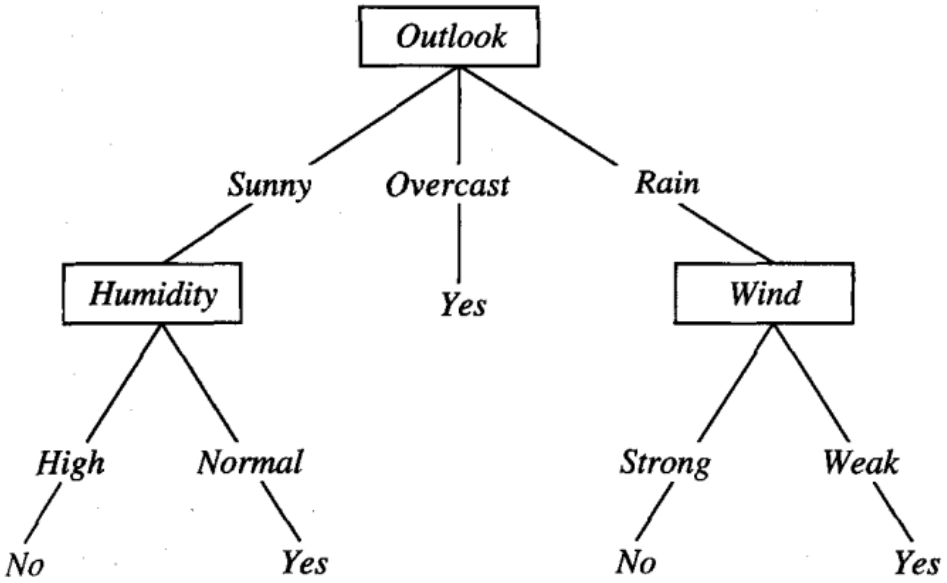
Kur formulā 3.3. $x_1, x_2, \ldots, x_n$ ir skaitliskās pazīmes, kas izgūtas no dokumenta un $\theta_0, \theta_1, \ldots, \theta_n$ ir modeļa parametri, arī saukti par svariem vai koeficientiem.

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 [11].

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.

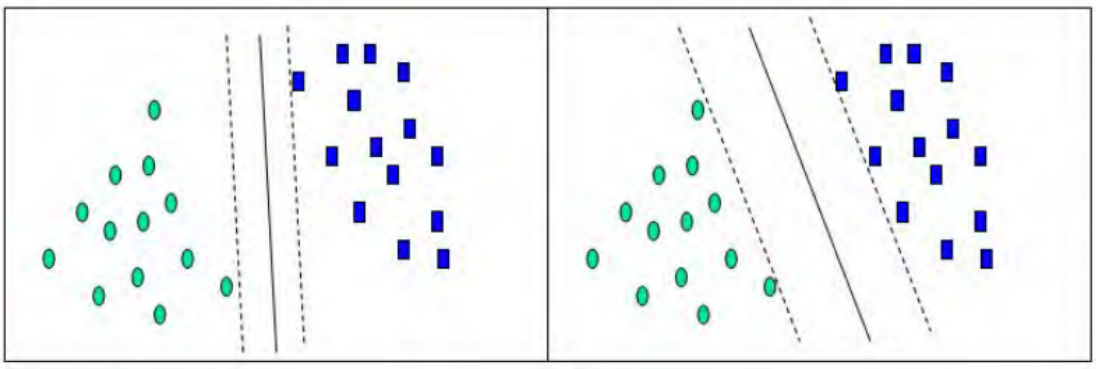


3.1. att. **Lēmumu koka ilustrācija [11]**

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.

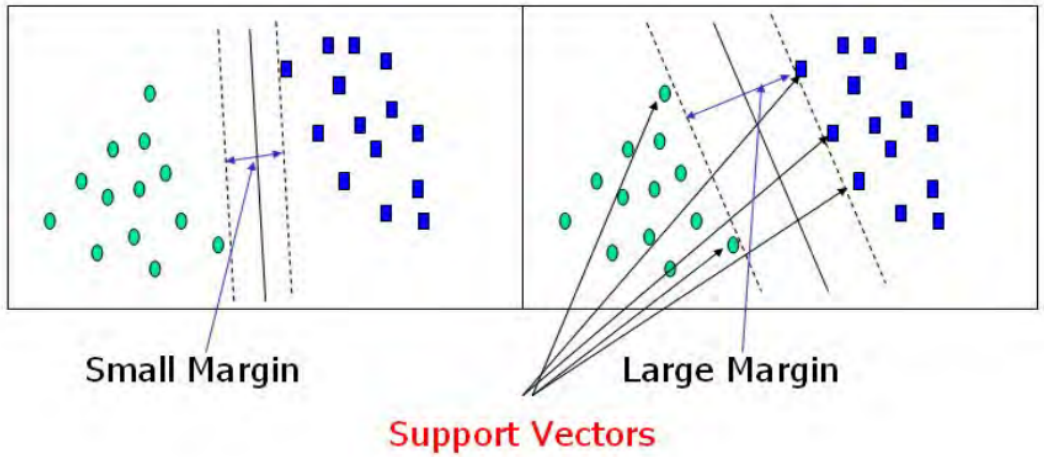
Atbalsta vektoru mašīnas (SVM)

Atbalsta vektora mašīnas [12] (*support vector machines* jeb SVM) ir pārraudzītās mācīšanās algoritms kurš ir populārs tieši klasifikācijas 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 [12]**

Šajā piemērā no 3.2. attēla - 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. Pastāv praktiski neierobežots līniju skaits, kas spētu veikt šādu nodalījumu, attēlā norādīti divi piemēri. 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 (*margin*). Vektori kas atrodas pie šīs robežas ir atbalsta vektori.



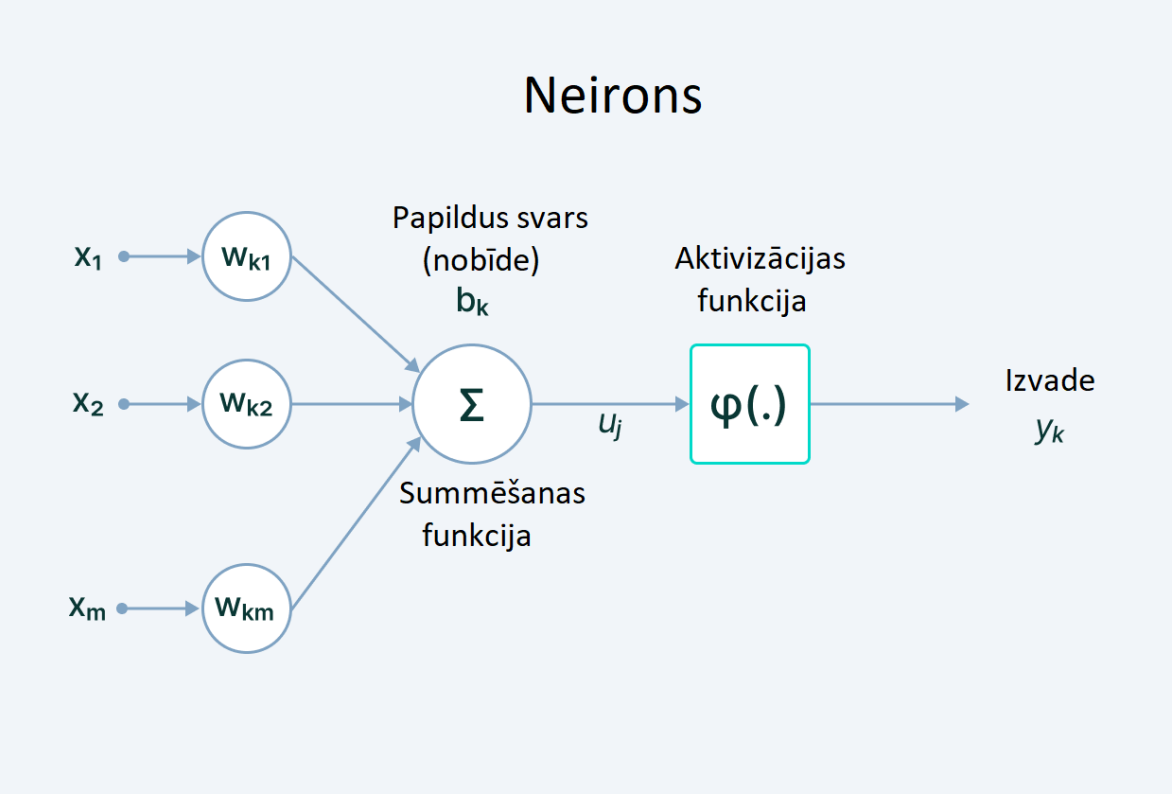
3.3. att. **Robežas un atbalsta vektoru ilustrācija [12]**

Atbalstu vektora mašīnas analīze atradīs līniju (vispārīgi – hiperplakni), kas novietota ar 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 [13].

Sākumā jāapskata māklīgā neironu tīkla pamatvienība - neirons.



3.4. att. **Mākslīgā neirona attēlojums**

Kā redzams attēlā attēlā 3.4., šī neirona uzbūvi raksturo tā pieci 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.

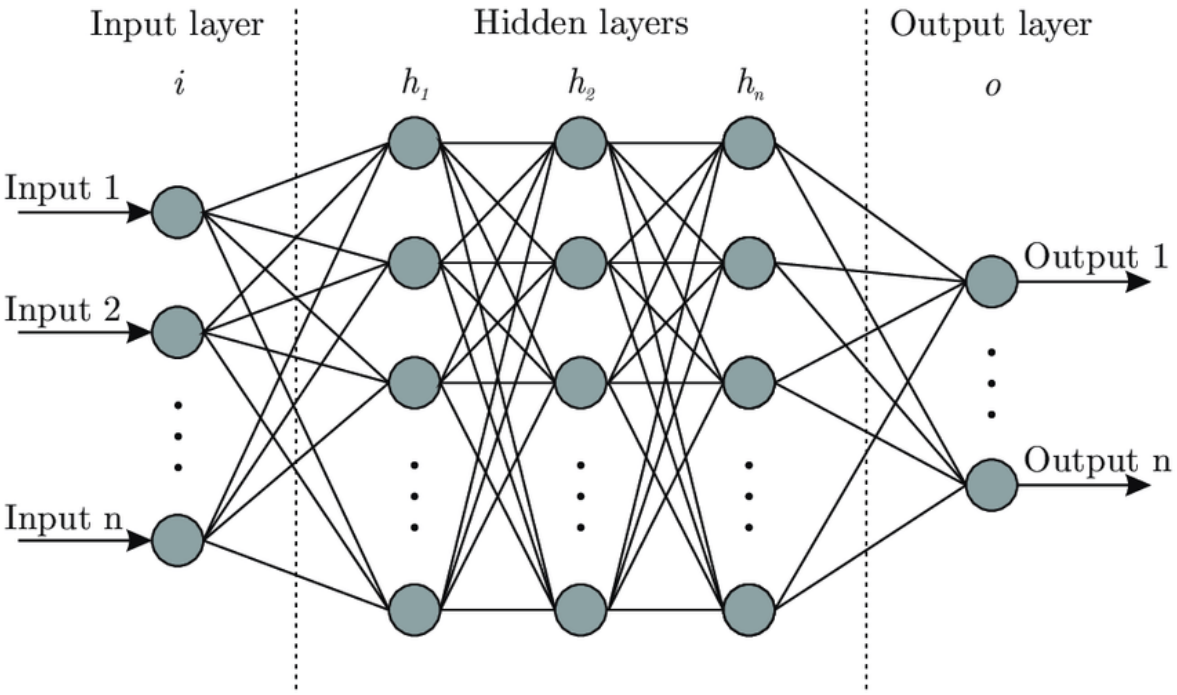
Summēšanas funkcija - šīs funkcijas mērķis ir apvienot vairākus ieejas signālus un to svarus vienā vērtībā, ko tālāk padot aktivizācijas funkcijai.

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 apmācību procesā.

Kad vairāki neironi ir salikti kopā pēc kārtas, tie veido slāni. Vairākus slāņus apvienojot varam iegūt daudzslāņu neironu tīklu. Vispārēja daudzslāņu neironu tīkla arhitektūra,

apskatīta 3.5.attēlā, aprakstāma kā savstarpēji savienoti mākslīgie neironi trijos vai vairāk slāņos.



3.5. att. **Mākslīgā neirona tīkla uzbūves vispārinājums [14]**

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.

Slēptie slāņi

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. Izplatītāko arhitektūru salīdzinājums

Pēdējos gados dabīgās valdoas apstrādē aizvien svarīgāka loma ir lielajiem valodas modeļiem (LLM), kas tiks apskatīti arī šajā darbā. Šādi modeļi, piemēram BERT un XLNet, arī tekstu klasifikācijā uzrādījuši vislabākos rezultātus daudzās angļu valodas tekstu kopās[1], tomēr modeļi ar labāko veiktspēju apmācīti tieši uz angļu valodas tekstiem un nav tieši pielietojami latviešu valodas apstrādei. Darba ietvaros tiks apskatīts uz latviešu valodas tekstiem apmācītais LVBERT modelis. Pozitīvās un negatīvās LLM īpašības tiek apskatītas 3.1. tabulā.

3.1. tabula

Neironu tīklu arhitektūru salīdzinājums - LLM

Pozitīvās īpašības	Plaša valodas morfoloģiskā un sintaktiskā izpratne, ļaujot labi uztvert kontekstu un valodas nianses
	Iepriekš apmācīti uz lieliem valodas korpusiem, pielāgojami konkrētai problēmai
Negatīvās īpašības	Lielāks apmācības laiks / resursu patēriņš apmācībai par LSTM / KNT arhitektūrām
	Var būt pārmērīgi sarežģīta pieeja vienkāršākiem uzdevumiem
	Uz latviešu valodu apmācītie modeļi lieto mazāku apmācības kopu par angļu valodas modeļiem

Tā kā LVBERT spēja veikt klasifikāciju nav tik labi dokumentāta kā citiem angļu valodas modeļiem, tiek apskatīti arī citi neironu tīklu paveidi. Cits populārs neironu tīkla veids, kas ir plaši pielietots valodas apstrādē un bija īpaši populārs pirms transformatoru arhitektūras, ir rekurentie neironu tīkli ar ilgtermiņa īstermiņa atmiņu (LSTM balstīti modeļi) - šāda pieeja ļauj gūt labus rezultātus apstrādē kur liela nozīme ir vārdu secībai un atkarībai no citiem apkārt esošiem vārdiem, piemēram, mašīntulkošanā. Pozitīvās un negatīvās LSTM īpašības tiek apskatītas 3.2. tabulā.

3.2. tabula

Neironu tīklu arhitektūru salīdzinājums - LSTM

Pozitīvās īpašības	Labi uztver attālas vārdu atkarības un kontekstu
	Īpaši efektīvs secīgu datu apstrādē
Negatīvās īpašības	Ilgāks apmācības laiks / resursu patēriņš par KNT

Teksta klasifikācijā valodas izpratnes nozīme, ko spēj sasniegt LSTM modeļi, gan nav tik viennozīmīga kā citās valodas apstrādes problēmās, piemēram, Ņujorkas Universitātes pētnieku salīdzinājums starp dažādiem modeļiem teksta klasifikācijā [15], pielietojot tos uz dažādām datu kopām, skatāms 3. pielikumā. Kā redzams no šiem datiem, labākos rezultātus sasniedz klasiski modeļi ar loģistisko regresiju un KNT modeļi (atkarībā no datu kopas), LSTM modeļiem neuzrādot labākus veiktspējas rādītājus nevienā no apskatītajām datu kopām. Savukārt Mathieu Cliche pētījumā par sentimenta analīzi uz dažādu gadu Twitter datu kopām [16], salīdzinot KNT un LSTM modeļus, iegūtie akurātuma rezultāti dažu gadu datu kopās (skatīt tabulu 4. pielikumā) sasniedz vienādus rezultātus (2013., 2014. un 2016. gada dati), savukārt 2015. gada datiem labāko rezultātu uzrādot KNT modeļiem. Pozitīvās un negatīvās KNT īpašības tiek apskatītas 3.3. tabulā.

3.3. tabula

Neironu tīklu arhitektūru salīdzinājums - KNT

Pozitīvās īpašības	Labi uztver lokālās iezīmes (frāzes, vārdu kombinācijas)
	Mazāks apmācības laiks / resursu patēriņš
	Labāka veiktspēja ar mazu apmācības datu kopu
Negatīvās īpašības	Ierobežots attālu vārdu atkarību un konteksta uztveršanā

Tālāk tiks sīkāk apskatīti augstākminētie neironu tīklu arhitektūras paveidi.

3.2.2. Konvolūcijas neironu tīkli

Konvolūcijas neironu tīklu sākotnējais mērķis ir bijis datorredzes problēmu risināšana ar dziļās mašīnmācīšanās palīdzību, 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 [17]. Sākumā tiek apskatīti pamatslāņi, kurus apvienojot iespējams veidot konvolūcijas neironu tīklus.

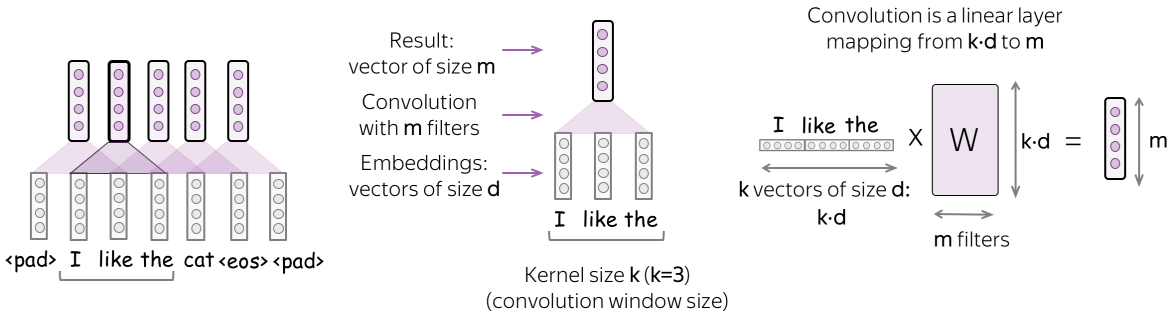
Iegulšanas slānis

Neironu tīklā iegulšanas slānis pārvērš ievades vārdus vektoru attēlojumā, veidojot vārdlietojuma kartējumu (jau iepriekš apskatītu pie vektorizācijas pieejām). Šis ir svarīgs solis, lai modelis varētu efektīvi apstrādāt un saprast teksta datus, it īpaši vārdu kontekstu.

Vārdi ar līdzīgām nozīmēm tiek kartēti kā līdzīgi vektori, ļaujot izprast vārdu saistības un teikuma uzbūves sakarības. Šie vektori ir apmācāmi parametri, respektīvi, apmācības laikā tiek modificēti to svari, ļaujot neironu tīklam apgūt vispiemērotāko vārdu attēlojumu konkrētajam uzdevumam.

Konvolūcijas slānis

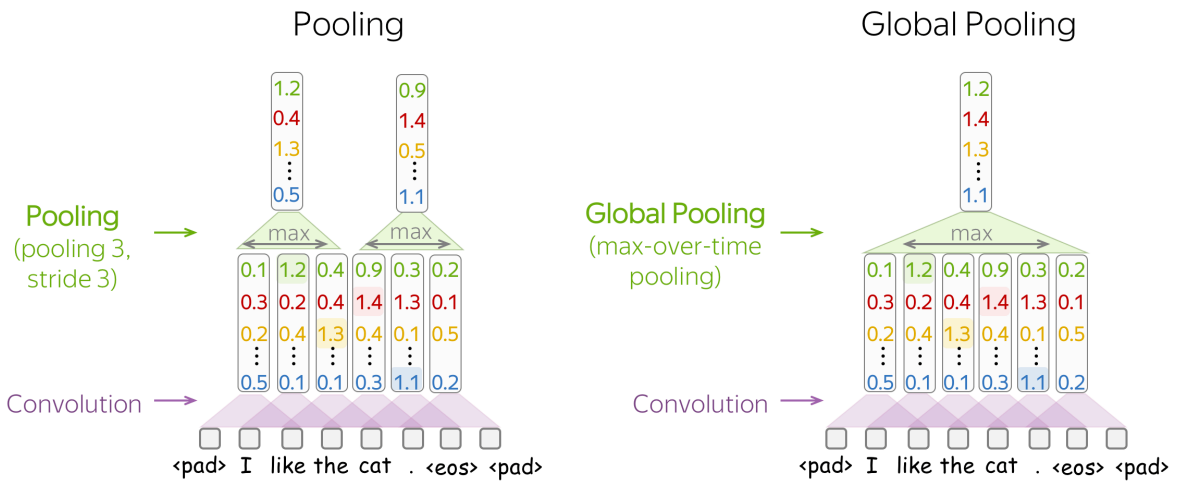
Konvolūcijas darbības princips ir filtru pielietošana, secīgi ejot cauri ievadei un meklējot dažādas pazīmes. Atkarībā no apskatāmās problēmas šie filtri var ieņemt dažādas formas – pielietojot konvolūciju uz attēliem parasti tiks pielietoti trīs dimensiju filtri. Teksta apstrādes gadījumā savukārt tiek lietoti viendimensionāli filtri ar noteiktu platumu, ko parasti sauc par kodola (*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 [18]**

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. **Apvienošanas slānis [18]**

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 3.7. attēlā - šādas apvienošanas operācijas varam veikt ar noteiktu soļa izmēru (*stride*) vai pielietojot uz visu ievades virkni.

Atmešanas slānis

Lai izvairītos no pārmērīgas pielāgošanas apmācības laikā bieži tiek lietots arī atmešanas slānis (*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 (*dropout rate*), kas norāda kādu daļu no neironiem mēs atmetam apmācību 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.

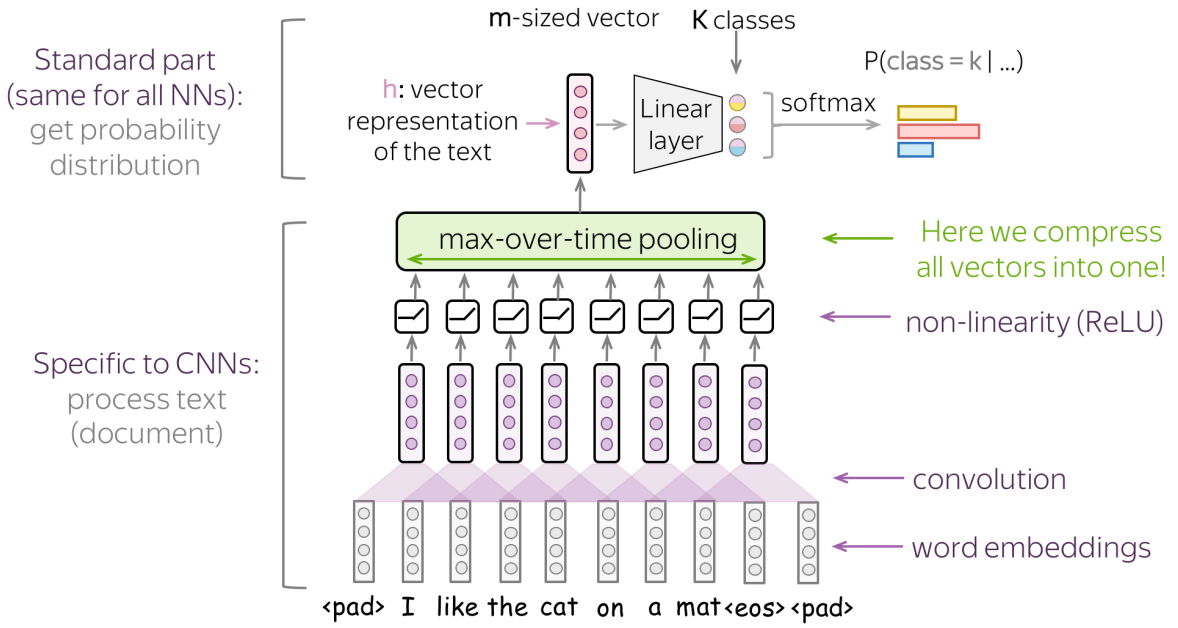
Ilustrējot ar piemēru, ja dots izvades slānis ar vektoru $z = (2.0, 1.0, 0.1)$ trīs klašu klasifikācijai, iegūtās varbūtības katrai klasei būtu:

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

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

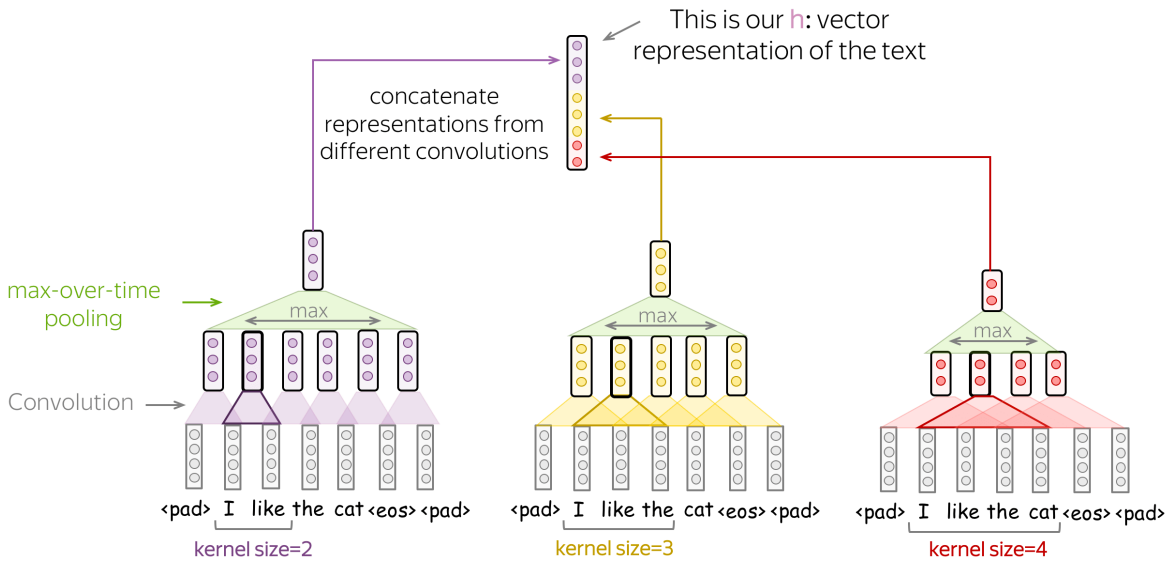
$$\text{softmax}(z)_3 = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.09856589$$

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 [18]**

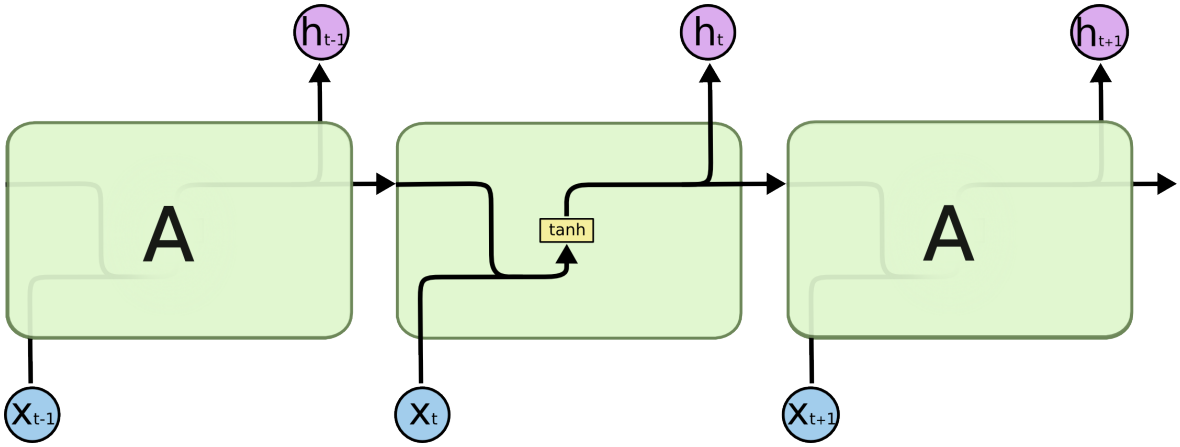
Standarta arhitektūrai iespējams arī veikt dažādus uzlabojumus, viena no literatūrā biežāk minētajām pieejām, ko piedāvā Kim Yoon [17], ir pielietot paralēli vairākus konvolūcijas un apvienošanas slāņus ar dažādiem filtra izmēriem (pētījumā filtri izvēlēti ar izmēriem kā 2,3 un 4), vēlāk šos slāņus apvienojot pirms padošanas tālāk.



3.9. att. **Arhitektūra ar dažādu izmēru konvolūcijas slāņiem [18]**

3.2.3. Rekurentie neironu tīkli

Rekurentie neironu tīkli (RNT) ir neironu tīkla veids, kas palīdz apstrādāt secīgus datus, piemēram, tekstus, ņemot vērā to secību un atkarības starp datiem. Šie tīkli satur rekurentus savienojumus neironu tīklā ar kuru palīdzību iespējams saglabāt slēpto stāvokli ar informāciju no iepriekšējiem apstrādes soļiem.



3.10. att. **RNT ilustrācija [19]**

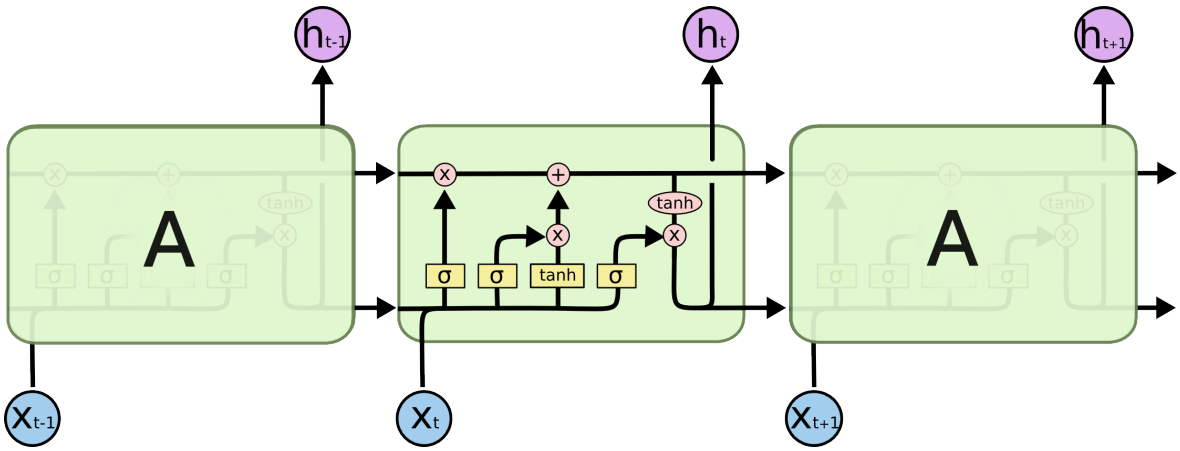
Gradientu zušana

Neironu tīklos gradienti kalpo kā indikatori svara korekcijai, samazinot atšķirību starp prognozētajiem un faktiskajiem rezultātiem. Tomēr gradienti bieži kļūst aizvien mazāki, virzoties caur tīkla slāņiem, potenciāli izraisot mācīšanās palēnināšanos vai pat apstāšanos sākotnējos slāņos. Šo parādību sauc par izzūdošā gradienta problēmu. Rekurento neironu tīklu gadījumā šī parādības a rodas no rekurentajiem savienojumiem un gradientu vērtībām pēc daudziem soļiem, respektīvi, katrā laika posmā RNT šūna izmanto gan pašreizējo ievadi, gan slēpto stāvokli no iepriekšējā laika posma, lai aprēķinātu jaunu slēpto stāvokli un izveidotu izvadi. Šī atkarība no iepriekšējiem laika posmiem izraisa gradienta izplatīšanos, izmantojot atkārtotu savienojumu ķēdi. Veicot aprēķinus par vairākiem iepriekšējiem soļiem – šo sākotnējo soļu gradienti var kļūt ļoti mazi, jo katrā laika posmā tiek atkārtoti aprēķināti gradienti. Tie var samazināties līdz stāvoklim, kur tie faktiski ir ar nulles vērtību. Rezultātā secības agrākie laika soļi apmācības laikā saņem ļoti vājus gradienta signālus. Turklāt to atbilstošie svari tiek atjaunināti lēni vai netiek atjaunināti vispār. Tas noved pie svarīgas informācijas zaudēšanas un lēnas mācīšanās konverģences.

LSTM

LSTM tika izstrādāts lai mēģinātu risināt izzūdošā gradienta problēmu. Lai to izdarītu, LSTM izmanto vārteju mehānismus, lai kontrolētu informācijas plūsmu un gradientus. Tas palīdz novērst izzūdošu gradienta problēmu un ļauj tīklam mācīties un saglabāt informāciju garākās virknēs.

LSTM pamatā ir šūnas stāvoklis, kas tiek nodots no šūnas ievades uz izvadi, veidojot sava veida ilgtermiņa atmiņu (ilustrēts 3.11. attēlā). Pamatā ir 3 vārtejas – aizmiršanas vārti, ieejas vārti, izvades vārti. Informācija plūst pa visu ķēdi tikai ar nelielām lineārām darbībām caur trim vārtiem, kas nosaka informācijas plūsmu un to kāda informācija tiek saglabāta vai ignorēta.



3.11. att. **LSTM ilustrācija [19]**

Aizmiršanas vārti izlemj, cik daudz no ilgtermiņa atmiņas jeb šūnas stāvokļa jāpārnes no iepriekšējā šūnas stāvokļa uz nākošo . Šim nolūkam tiek izmantota sigmoīda funkcija, kas norāda šūnas stāvokļa nozīmi. Izvade f_t svārstās no 0 līdz 1 un norāda, cik daudz informācijas tiek saglabāts, t.i., 0 - nesaglabāt informāciju un 1 - saglabāt visu informāciju par šūnas stāvokli. Aizmiršanas vārtus raksturo formula

$$f_t = \sigma \left(W_{f,x} x_t + W_{f,h} h_{t-1} + b_f \right) \tag{3.5}$$

Kur:

- σ ir sigmoīda aktivācijas funkcija.

- $W_{f,x}$ ir ievades svaru matrica x_t laika solī t.

- x_t ir ievades vektors laika solī t.

- $W_{f,h}$ ir slēptā stāvokļa svara matrica h_{t-1} no iepriekšējā laika soļa.

- h_{t-1} ir slēptais stāvoklis no iepriekšējā laika soļa.

- b_f ir aizmiršanas vārtu novirzes vērtība.

Ievades vārti izlemj, kura informācija jāpievieno šūnas stāvoklim un tādējādi arī ilgtermiņa atmiņai. Izvade i_t svārstās no 0 līdz 1 un norāda, cik daudz informācijas tiek pievienots, t.i., 0 - nepievienot jaunu informāciju un 1 - pievienot visu jauno informāciju. Šo vārteju raksturo zemāk minētā formula.

$$i_t = \sigma \left(W_{i,x} x_t + W_{i,h} h_{t-1} + b_i \right) \tag{3.6}$$

Izvades vārti izlemj kuras šūnas stāvokļa daļas veido izvadi. Tādējādi izejas vārti ir atbildīgi par īstermiņa atmiņu. Izvade o_t svārstās no 0 līdz 1 un norāda, cik daudz no šūnas stāvokļa tiek izvadīts, t.i., 0 - neizvadīt neko par šūnas stāvokli un 1 - izvadīt visušūnas stāvokli. Vārteju raksturo sekojoša formula.

$$o_t = \sigma\left(W_{o,x}x_t + W_{o,h}h_{t-1} + b_o\right) \tag{3.7}$$

Kā redzams, visus trīs vārtus attēlo viena un tā pati funkcija. Atšķiras tikai svari un novirzes. Šūnas stāvoklis tiek atjaunināts, izmantojot aizmiršanas vārtus un ievades vārtus.

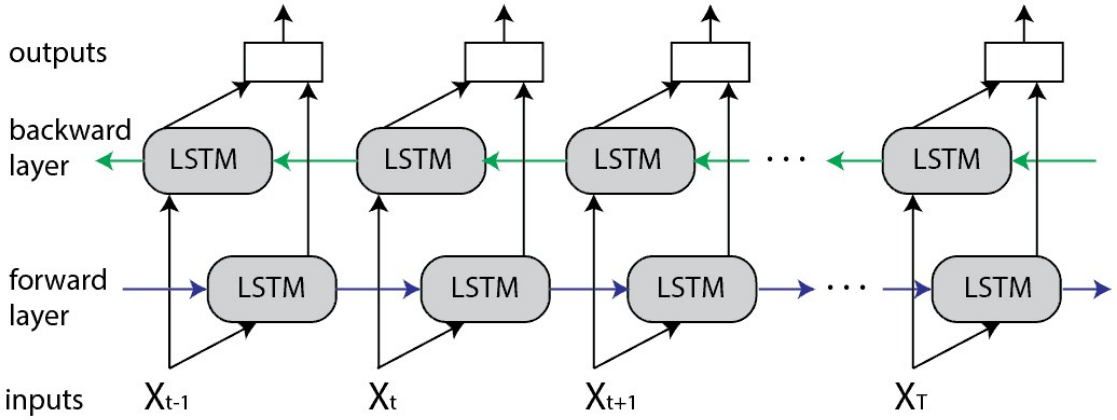
$$c_t = (f_t \cdot c_{t-1}) + (i_t \cdot \tanh(h_{t-1})) \tag{3.8}$$

Pašreizējā laika soļa slēpto stāvokli h_t nosaka izejas vārti un tanh funkcija, kas ierobežo šūnas stāvokli no -1 līdz 1.

$$h_t = o_t \cdot \tanh(c_t) \tag{3.9}$$

Divvirzienu LSTM

Divvirzienu LSTM jeb BiLSTM ir LSTM arhitektūras variants, kas sastāv no diviem atsevišķiem LSTM slāņiem - viens slānis apstrādā ievades datus secīgi, bet otrs apstrādā ievadi apgrieztā secībā. Šo divu LSTM slāņu izejas tiek apvienotas katrā laika solī, iegūstot galīgo izejas rezultātu. Šāda arhitektūra ilustrēta 3.12. attēlā.



3.12. att. **Divvirzienu LSTM ilustrācija [20]**

Galvenā šādas pieejas priekšrocība ir plašāka konteksta un vārdu sasvstarpējo atkarību izpratne, apskatot vārdus gan pēc, gan pirms konkrētā fragmenta.

3.2.4. Transformatori un lielie valodu modeļi

Lielie valodu modeļi (*Large language models* jeb LLM) kā ChatGPT pēdejos gados guvuši lielu popularitāti un tiek plaši pielietoti arī dabīgās valodas 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"* [21] 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

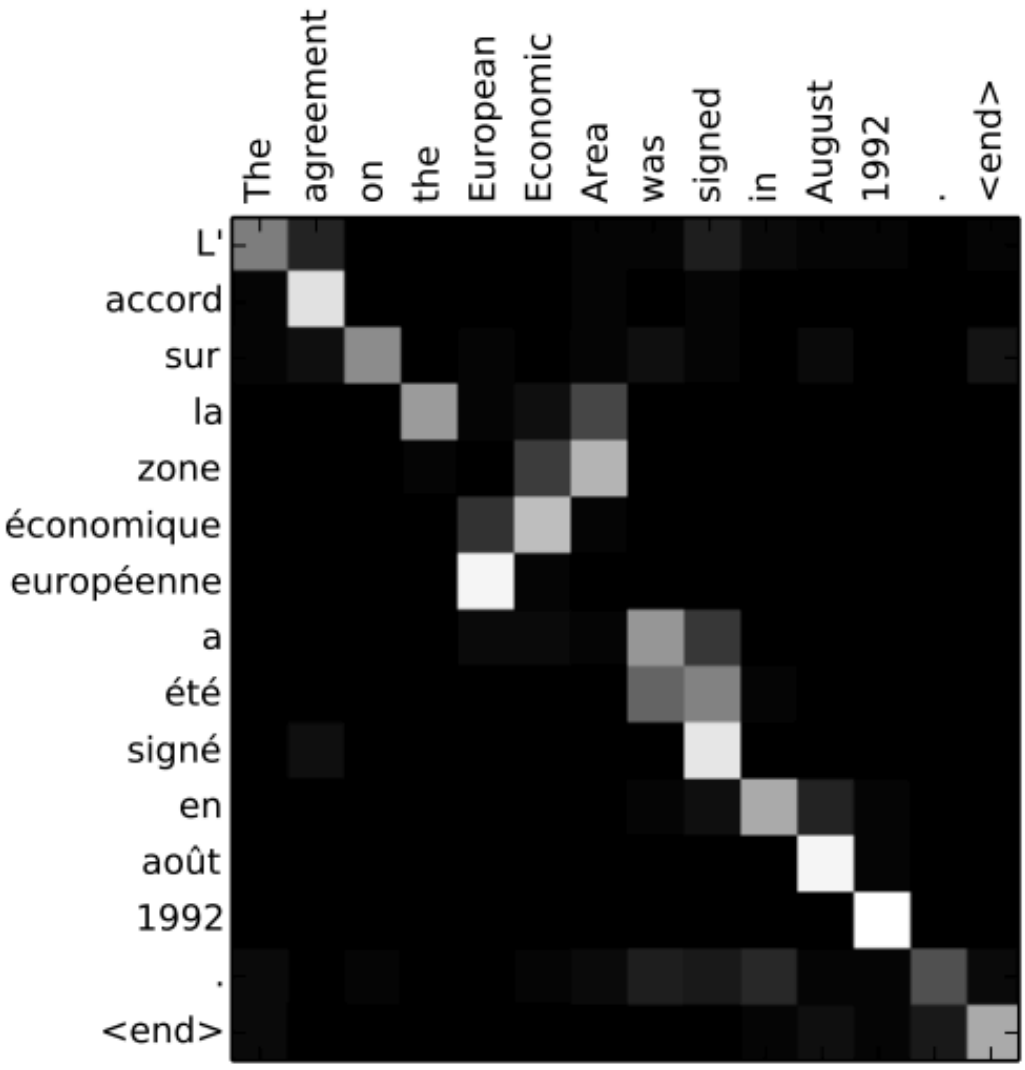
Sākotnēji uzmanības mehānisma ideja tika apskatīta tieši tulkošanas kontekstā, 2015. gada publikācijā "Neural Machine Translation by Jointly Learning to Align and Translate" [22]. Kā tieši šis uzmanības mehānisms izpaužas viegli saprast no pētījumā iekļautās problēmas apskatīšanas, kur tiek veikts tulkojums 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:

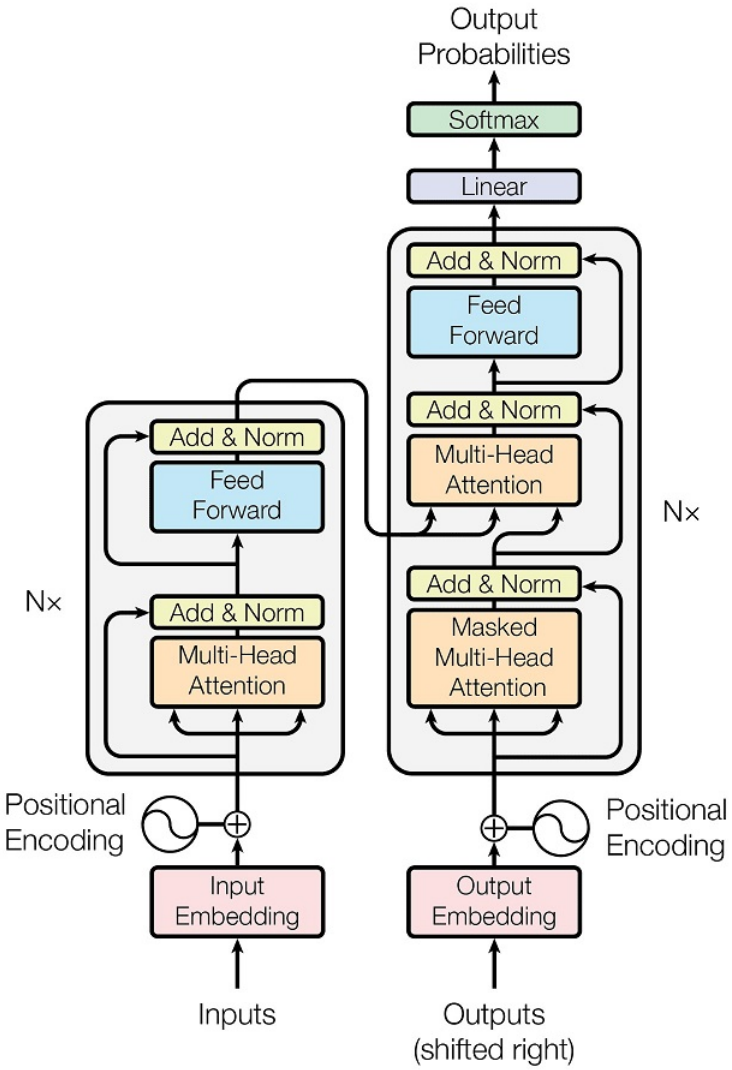


3.13. att. **Uzmanības mehānisma ilustrācija [22]**

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 dabīgās valodas apstrādes rīks kopš tā atklāšanas 2015. gadā, taču sākotnējā formā tas tika izmantots kopā ar rekurentiem neironu tīkliem (RNT).

Transformatora arhitektūra



3.14. att. **Transformatora arhitektūra [21]**

Transformatora arhitektūra tiek apskatīta 3.14. 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 var definēt tālāk norādītā veidā.

$$Uzmaniba(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d}}\right) \cdot \mathbf{V} \tag{3.10}$$

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 (*feed-forward*) slānim.

Padeves slānis

Padeves (*feed forward*) 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.

3.2.5. BERT

BERT modelis pirmo reizi publicēts 2019. gadā un autoru vārdiem BERT ir pirmā nepārraudzītās apmācības pieeja ar dziļu divvirziena konkteksta izveidi dabīgās valodas priekšapstrādei [23]. Viena no BERT priekšrocībām ir tā, ka visu BERT arhitektūru un jau apmācītos modeļus Google komanda ir publiskojusi caur GitHub [1], atšķirībā no citiem izplatītiem lielajiem valodas modeļiem kā GPT no OpenAI, kur modeļu izveides kods nav publiski pieejams un kuru nevaram izmantot darbā apskatītās problēmas risināšanai.

[1] https://github.com/google-research/bert

Ar BERT arhitektūru tiek apmācīts "valodas izpratnes" modelis, izmantojot lielapjoma teksta korpusus, piemēram, visus Wikipedia rakstus konkrētajā valodā. Pēc tam jāveic modeļa pielāgošana konkrētu dabīgās valodas apstrādes uzdevumu risināšanai kā tekstu klasifikācija. Nepārraudzītā apmācība ļauj veidot BERT modeli uz plašiem teksta korpusiem no jebkura interneta resursa.

Svarīga loma iekš BERT ir arī kā valodas dati tiek vektorizēti. Valodas attēlojumi vektora formā var iekļaut sevī kontekstu un var neiekļaut to. Citas iepriekš populārākās vektorizācijas pieejas kā word2vec ģenerē vārdlietojuma kartējumu bez konteksta katram vārdam apmācības vārdnīcā, respektīvi, 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, šis konteksts tiek veidots 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 apstrādā caur dziļu divvirzienu transformatora kodētāju, pēc tam paredz 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, tiek veikta apmācība 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 šādi.

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 kopu 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 divām un vairāk klasēm. Matrica sastāv no divām 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:

3.4. tabula

Pārpratuma matrica

	+	-
+	PA	KA
-	KN	PN

Kur tabulas 3.4. 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.

$$Akurātums = \frac{PA + PN}{PA + PN + KA + KN} \tag{2.11}$$

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.

$$Precizitāte = \frac{PA}{PA + KA} \tag{2.12}$$

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.

$$Pārklājums = \frac{PA}{PA + KN} \tag{2.13}$$

F1 mērs

F1 mērs ir precizitātes un pārklājuma mēru harmoniskais vidējais.

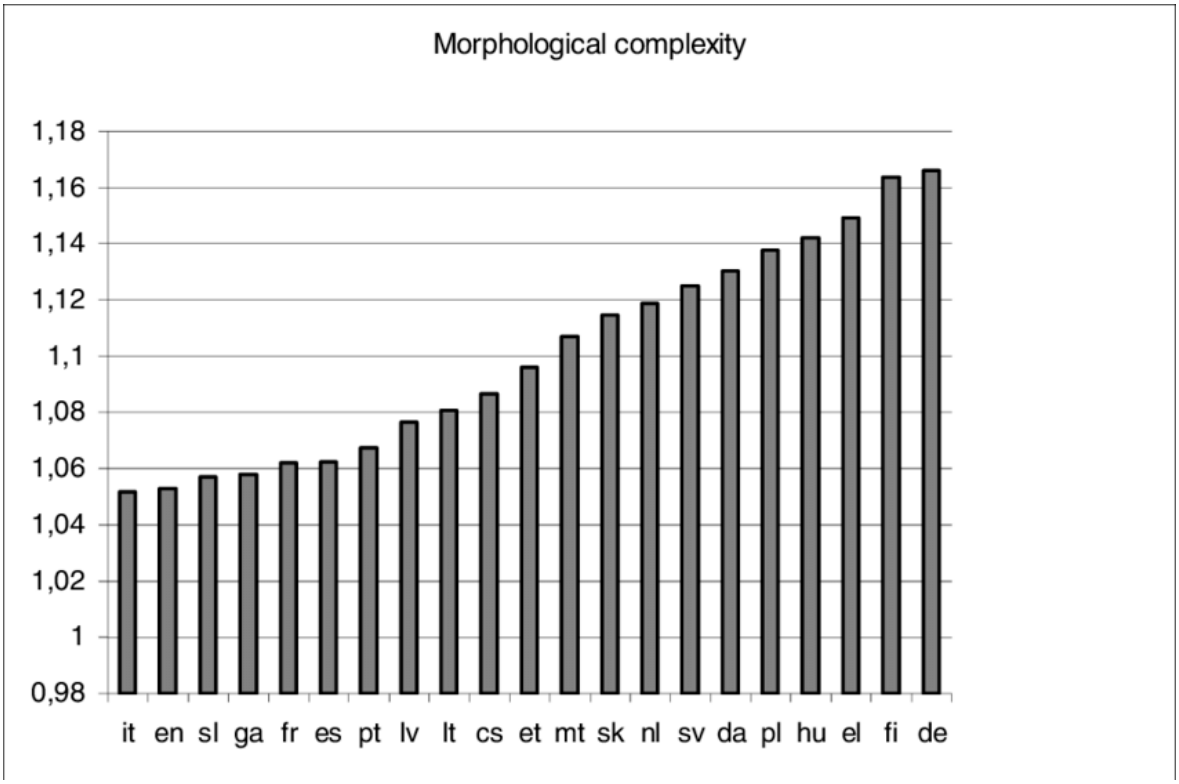
$$F1 = \frac{(2 * precizitāte * pārklājums)}{(precizitāte + pārklājums)} \tag{2.14}$$

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 [24], kurai dabīgās valodas apstrāde ir pētīta visplašāk. Morfoloģiskā sarežģītība Eiropā izplatītajām valodām apskatīta attēlā 3.15..



3.15. att. **Morfoloģiskā sarežģītība Eiropas valodām[24]**

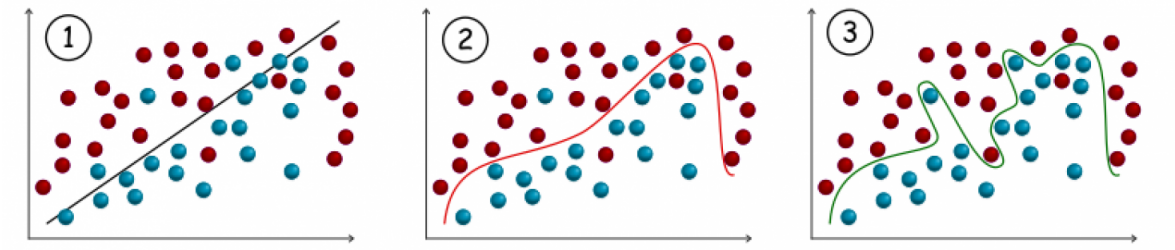
Papildus tam - dabiskā valoda var būt neskaidra, ar vārdiem un frāzēm, kuriem ir vairākas nozīmes atkarībā no konteksta. Šīs neskaidrības precīza risināšana ir izaicinājums teksta klasifikācijas modeļiem.

Tekstu nevienmērība

Teksta dati ir dažādi un atšķiras pēc garuma, struktūras un kvalitātes. Tie var ietvert konkrētu autoru tipiskās 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 [25]. Zemāk apskatāmajā attēlā 3.16. attēloti piemēri divdimensiju klasifikācijas scenārijā.



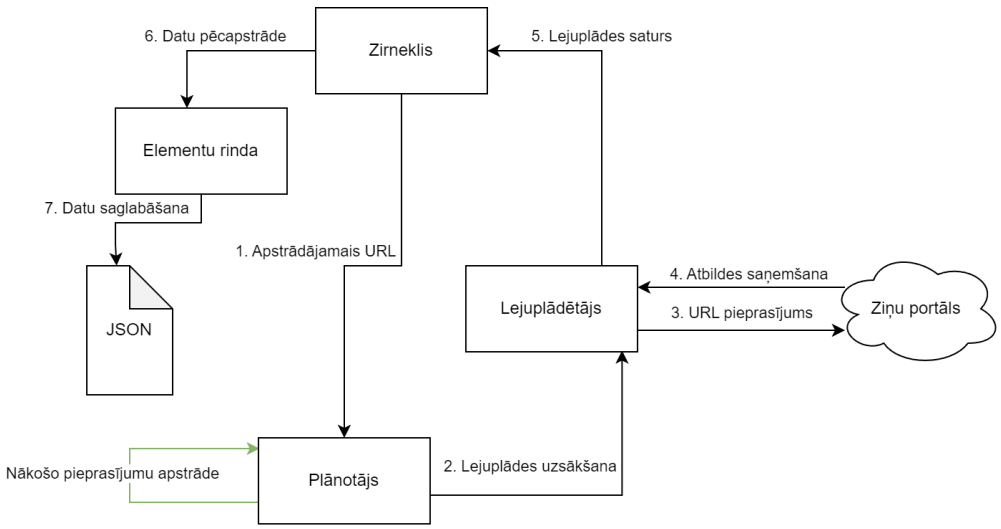
3.16. att. **Pielāgošanas scenāriji (1 - nepietiekama, 2 - optimāla, 3 - pārmērīga)**

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. Rāpuļa darbība apskatīta 4.1. diagrammā.



4.1. att. **Rāpuļa darbības ilustrācija**

Detalizētāks darbības princips rāpulim ir raksturojams sekojoši.

- Rāpulim sākotnēji apskatāmās saites tiek norādītas kā 10 izvēlēto kategoriju lapas (piemēram, politikas ziņām - https://www.delfi.lv/193/politics?page=1)
- Šīs saites tiek nodotas pieprasījumu plānošanai (vienlaicīgi tiek veikts ierobežots skaits pieprasījumu ar 0.2 sekunžu intervālu starp pieprasījumiem)
- Saitei sagaidot kārtu pieprasījuma veikšanai - tiek veikts pieprasījums uz ziņu portālu un saņemta atbilde
- Atgrieztās lapas saturs tiek nodots apstrādei un nākošo saišu izguvei
- No lapām tiek iegūtas hipersaites uz rakstiem, ierobežojot tālāk apskatāmās saites tā, lai tās aizvien piederētu apskatāmajām kategorijām un nesaturētu nevēlamas saites (kā komentāru lapas rakstiem)
- Ja lapa satur rakstu - tas tiek apstrādāts, piefiksējot hipersaiti un dažādas komponentes - virsrakstu, kategoriju, saturu, atlasot tos pēc HTML elementu atbilstības konkrētas komponentes kritērijiem
- Katrs raksts tiek saglabāts JSON formātā un ierakstīts failā, atkārtojot procedūru līdz vairs netiek atrasti unikāli raksti ko rāpulim apmeklēt

Scrapy ietvars kopumā dod iespēju diezgan efektīvi realizēt ziņu rakstu ievākšanas rāpuli – sākot no apskatāmo rakstu ierobežošanas līdz elementu atlasei un apstrādei. Salīdzinoši grūtāk ir tieši formalizēt kādus komponentes kritērijus jeb 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 2. pielikumā. Rezultātā tika ievākti 13762 raksti ar sadalījumu pa kategorijām kāds redzams 4.1. tabulā.

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 5. pielikumā.

Rāpuļa kods pieejams 1. pielikuma 1. repozitorijā.

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 [1]) . 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.

Lemmatizācija netiek pielietota, dazādiem autoriem norādot tās mazo nozīmi tekstu klasifikācijas modeļu uzlabošanā [26]. Citos pētījumos, piemēram par angļu un čehu valodas datu kopu priekšapstrādi [27], secināts, ka kopumā stopvārdu izņemana gandrīz vienmēr uzlabo iegūtos rezultātus, tomēr lemmatizācija gandrīz vienmēr atstāj negatīvu ietekmi uz gala modeli.

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 pie-

[1] https://github.com/stopwords-iso/stopwords-lv

eju, tomēr *FastText* pielietošana sniedza labākus sākotnējos rezultātus, arī citi pētījumi par vārdlietojuma kartējuma pieejām latviešu valodas tekstiem [28] norāda *FastText* kā pieeju ar labāko veiktspēju, salīdzinot ar *word2vec*, *ngram2vec*, SSG (*Structured Skip-Gram*).

FastText gadījumā tiek lietots jau apmācīts vārdu vektors, apmācīts uz *Common Crawl*[1] rāpuļa un *Wikipedia* rakstu kopām latviešu valodā. Vārdu vcktors ir publicēts FastText oficiālajā mājaslapā [2] un tiek pielietots kodā caur *Gensim* bibliotēku.

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).

4.2.2. Algoritmu implementācijas

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.

[1] https://commoncrawl.org/
[2] https://fasttext.cc/docs/en/crawl-vectors.html

4.2. tabula

Akurātuma un F1 mēri pielietojot AVM

	Ar priekšapstrādi	Bez priekšapstrādes
Atbalsta vektora mašīna (TF-IDF)	Ak.: 0.9738 F1: 0.9737	Ak.: **0.9746** F1: 0.9746
Atbalsta vektora mašīna (Bigrammu maiss)	Ak.: **0.9679** F1: 0.9678	Ak.: 0.9642 F1: 0.9641
Atbalsta vektora mašīna (Vārdu maiss)	Ak.: **0.9696** F1: 0.9695	Ak.: 0.9667 F1: 0.9666
Atbalsta vektora mašīna (FastText)	Ak.: **0.9596** F1: 0.9594	Ak.: 0.9592 F1: 0.9590

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 un F1 mēri pielietojot Naivā Bajesa metodi

	Ar priekšapstrādi	Bez priekšapstrādes
Naivā Bajesa metode (TF-IDF)	Ak.: **0.9617** F1: 0.9616	Ak.: 0.9537 F1: 0.9538
Naivā Bajesa metode (Bigrammu maiss)	Ak.: **0.9558** F1: 0.9559	Ak.: 0.94 F1: 0.9406
Naivā Bajesa metode (Vārdu maiss)	Ak.: **0.9629** F1: 0.9628	Ak.: 0.9567 F1: 0.9565
Naivā Bajesa metode (FastText)	Ak.: **0.8758** F1: 0.8757	Ak.: 0.8396 F1: 0.8399

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 un F1 mēri pielietojot loģistisko regresiju

	Ar priekšapstrādi	Bez priekšapstrādes
Loģistiskā regresija (TF-IDF)	Ak.: **0.9663** F1: 0.9662	Ak.: 0.9625 F1: 0.9625
Loģistiskā regresija (Bigrammu maiss)	Ak.: **0.9633** F1: 0.9633	Ak.: 0.9567 F1: 0.9566
Loģistiskā regresija (Vārdu maiss)	Ak.: **0.9663** F1: 0.9662	Ak.: 0.9550 F1: 0.9549
Loģistiskā regresija (FastText)	Ak.: **0.9575** F1: 0.9574	Ak.: 0.9558 F1: 0.9557

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

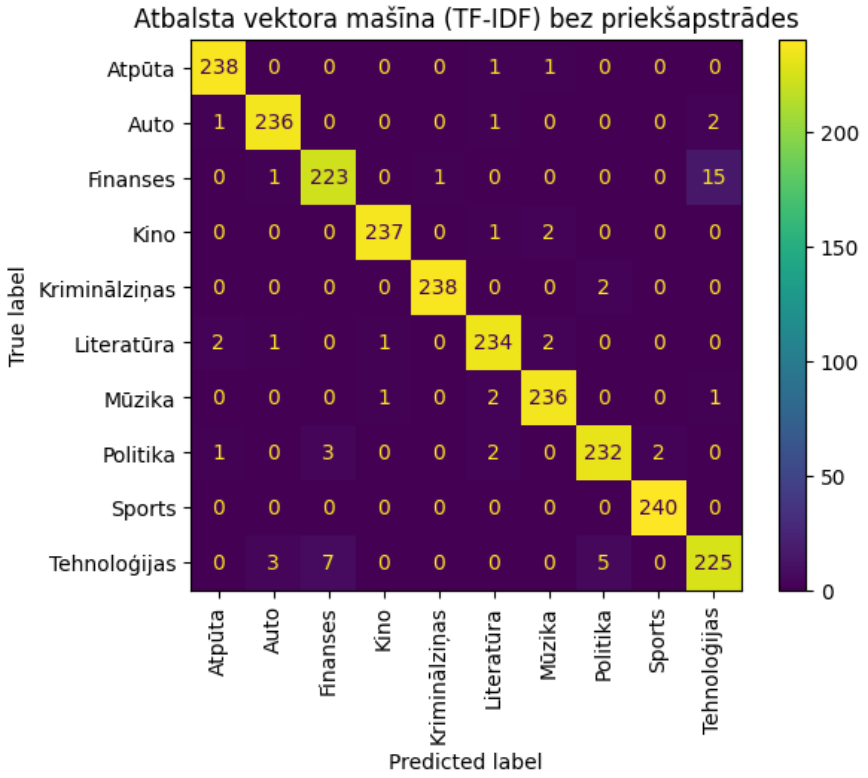
Akurātuma un F1 mēri pielietojot lēmumu kokus

	Ar priekšapstrādi	Bez priekšapstrādes
Lēmumu koki (TF-IDF)	Ak.: **0.8** F1: 0.8004	Ak.: 0.7850 F1: 0.7853
Lēmumu koki (Bigrammu maiss)	Ak.: **0.8250** F1: 0.8249	Ak.: 0.7979 F1: 0.7985
Lēmumu koki (Vārdu maiss)	Ak.: **0.8129** F1: 0.8129	Ak.: 0.7992 F1: 0.8001
Lēmumu koki (FastText)	Ak.: **0.7529** F1: 0.7517	Ak.: 0.7308 F1: 0.7294

Priekšapstrādes nozīme un rezultātu salīdzinājums

Lai veiksmīgi varētu novērtēt priekšapstrādes lomu modeļu veiktspējā, iepriekš apskatītie algoritmi un vektorizācijas pieejas tika pārbaudīti ar un bez tekstu papildus priekšapstrādes. Īpaši nozīmīgi tas ir modeļiem ar FastText vektorizāciju, jo šajā gadījumā priekšapmācītie vārdu vektori ir apmācīti bez priekšapstrādes [29]. Pēc tabulu datiem varam novērot, ka priekšapstrāde gandrīz vienmēr uzlabo gala rezultātu, lai gan sastopami arī izņēmums tieši modelī ar labāko rezultātu (atbalsta vektora mašīnas). Kopumā gan priekšapstrādes ietekme nav vērtējama kā pārāk liela – labākajam modelim sniedzot akurātuma uzlabojumu kā 0,0008.

Vislabākais sasniegtais akurātums iegūts ar atbalsta vektora mašīnām un TF-IDF vektorizāciju - **0.9738**, šai pieejai varam vizualizēt kategorizāciju ar pārpratuma matricu. Kā redzams 4.2. attēlā – desmit kategoriju klasifikācija tiek veikta ļoti precīzi un biežākā kļūda ir nepareizi klasificētas finanšu ziņas, klasificējot tās kā tehnoloģiju ziņas.



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

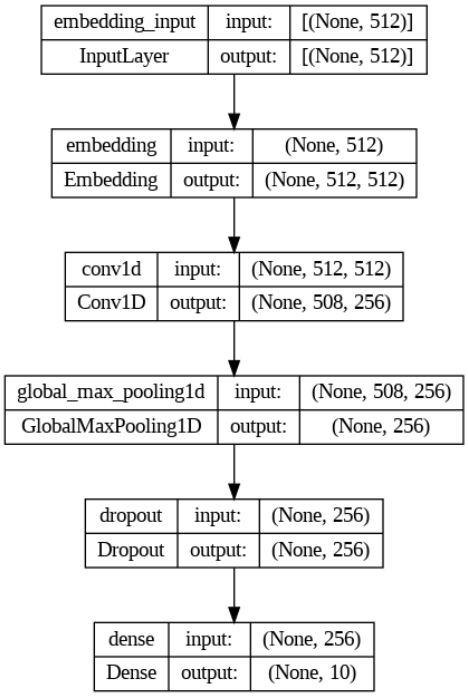
Klasisko mašīnmācīšanās algortimu implementācijas kods pieejams 1. pielikuma 2. repozitorijā zem ”Zinu_klasifikacijas.ipynb” faila. Izveidotie modeļi pieejami šī paša repozitorija apakšmapē ”models”. Labāko modeļu ielāde un pielietošana caur lietotāja saskarni implementēta 1. pielikuma 3. repozitorijā.

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 apmācību 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 4.3. attēlā.



4.3. att. **Konvolūcijas neironu tīkla uzbūve**

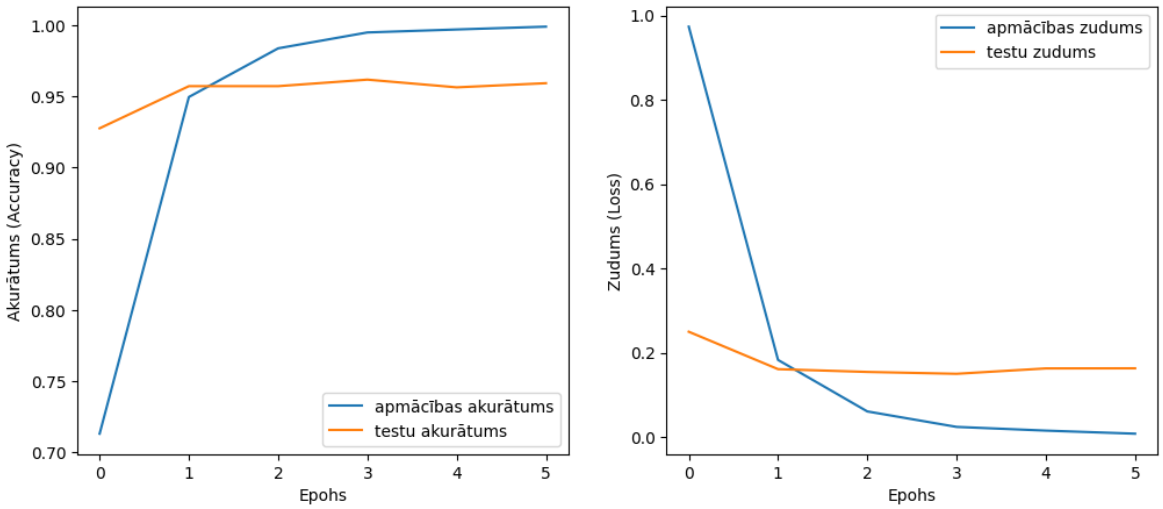
Apmācība tiek veikta ar partijas izmēru (*batch size*) kā 32, turpinot apmācību 10 epohos, saglabājot modeli posmos ar mazāko validācijas zuduma (*validation loss*) vērtību, mēģinot izvairīties no pārmērīgas pielāgošanas. Apmācība tiek pārtraukta priekšlaicīgi ja 2 epohos pēc kārtas validācijas zuduma vērtība turpināja pieaugt. Modeļi tiek pārbaudīti ar un bez priekšapstrādes (lielo burtu pārveidošana, simbolu un stopvārdu atmešana). Papildus tiek pārbaudīts kā apmācības rezultātu ietekmē jau apmācītu vārdu vektoru (*FastText*) svaru lietošana salīdzinājumā ar svaru atrašanu apmācības laikā.

4.6. tabula

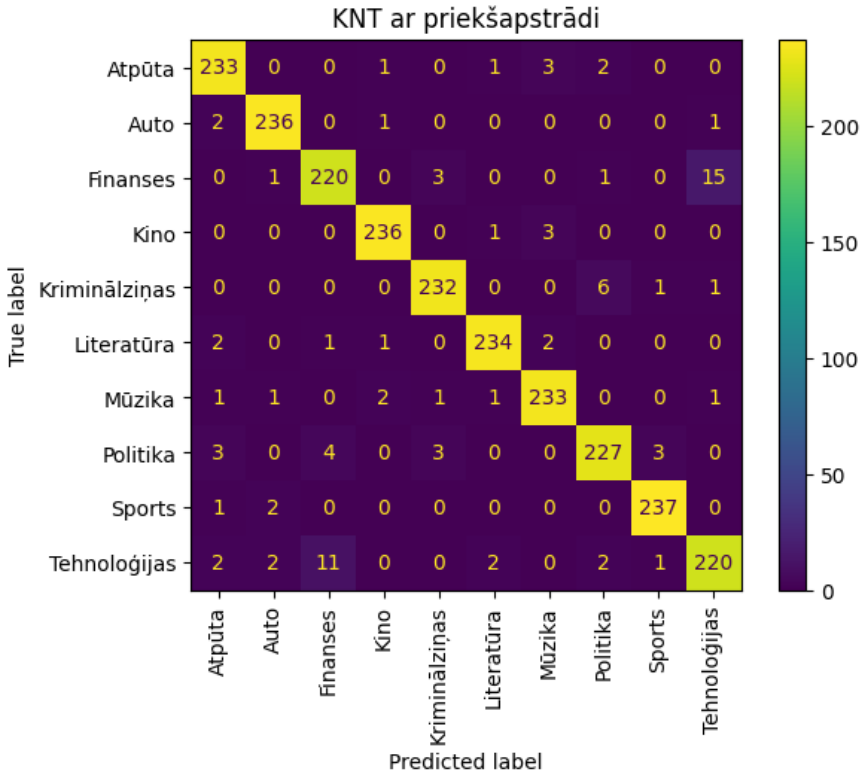
Veiktspējas mēri pielietojot konvolūcijas neironu tīklu

	Ar priekšapstrādi	Bez priekšapstrādes
KNT	Ak.: **0.9617** F1: 0.9616	Ak.: 0.9533 F1: 0.9532
KNT + FastText	Ak.: 0.9192 F1: 0.9195	Ak.: **0.9537** F1: 0.9537

Pielietojot attēlā 4.3. ilustrēto konvolūcijas neironu tīkla uzbūvi tiek iegūti rezultāti kā 4.6. tabulā. Pārpratuma matricu labākajam modelim iespējams apskatīt 4.5. attēlā, kur var novērot līdzīgu kļūdu izplatību pa klasēm kā iepriekš apskatītos modeļos. Arī ar šo pieeju visgrūtāk modelim ir atšķirt tieši tehnoloģiju un finanšu ziņas. Akurātuma un zuduma evolūciju pa epohiem iespējams apskatīt 4.4. attēlā.

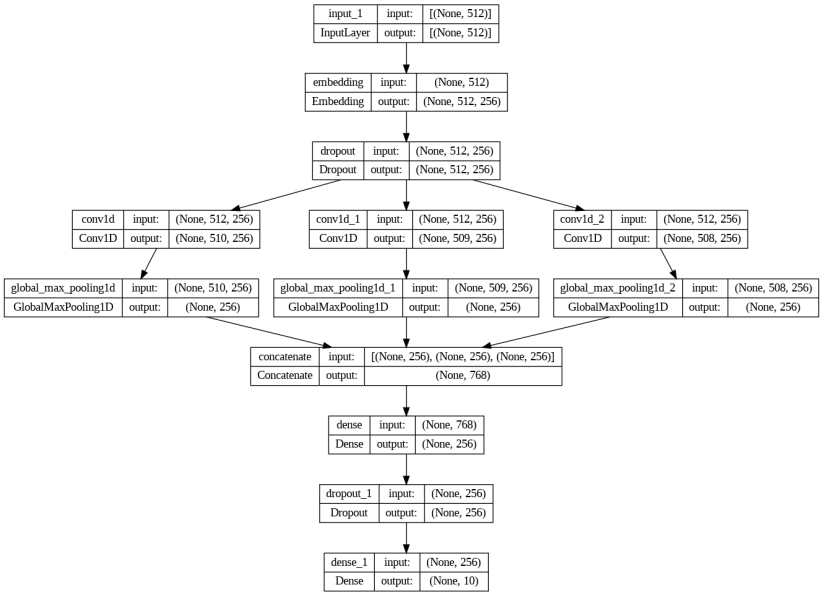


4.4. att. **Konvolūcijas neironu tīkli - novērtējums pa apmācības posmiem**



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

Mēģinot uzlabot modeļa veiktspēju tiek pielietota plaši citēta konvolūcijas neironu tīklu arhitektūra tekstu apstrādei no Kim Yoon [17]. 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 4.6. attēlā.



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

Modelis kurš apmācīts ar augstākminēto arhitektūru nesniedza veiktspējas uzlabojumus, kā redzams 4.7. tabulā.

4.7. tabula

Veiktspējas mēri - konvolūcijas neironu tīkls ar parelēlu konvolūciju

	Ar priekšapstrādi	Bez priekšapstrādes
KNT (paralēla konvolūcija)	Ak.: **0.9575** F1: 0.9576	Ak.: 0.9558 F1: 0.9558
KNT (paralēla konvolūcija) + FastText	Ak.: 0.9058 F1: 0.9064	Ak.: **0.9496** F1: 0.9494

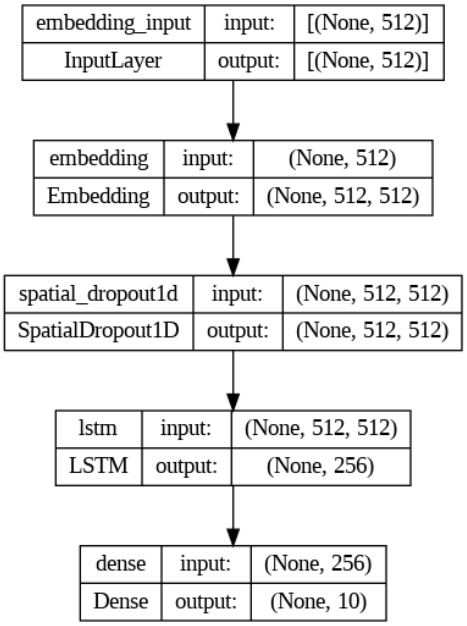
Priekšapstrādes nozīme

Arī konvolūcijas neironu tīkliem tika pārbaudīta priekšapstrādes nozīme, apskatot jau minētās arhitektūras ar un bez priekšapstrādes pielietošanas. Iespējams secināt, ka pielietojot konvolūcijas neironu tīklus, priekšapstrāde aizvien ir svarīgs solis, kas spēj pozitīvi ietekmēt modeļu klasificēšanas akurātumu, ja netiek pielietoti jau apmācīti vārdu vektoru svari. *FastText* apmācītie vārdu vektoru svari nespēja sniegt modeļa uzlabojumus rezultātam kas sasniegts ar svaru noteikšanu apmācības laikā.

Konvolūcijas neironu tīklu implementācijas kods pieejams 1. pielikuma 2. repozitorijā zem ”Zinu_klasifikacijas_KNT.ipynb” faila. Izveidotie modeļi pieejami šī paša repozitorija apakšmapē ”models”.

4.3.2. LSTM neironu tīkli

LSTM tīklu implementācijai tiek veidots Keras modelis ar secīgiem slāņiem – iegulšanas, viendimensiju telpiskās atmešanas (kur atšķirībā no regulārās atmešanas aizvieto nevis nejašus elementus ar nulles vērtību, bet visas vērtības vienā dimensijā), LSTM un pilnīgi savienotā slāņa gala klasifikācijas veikšanai. Ilustrēti ar papildus informāciju par izvēlētajām slāņu dimensijām varam apskatīt modeli 4.7. attēlā.



4.7. att. **Vienkārša LSTM tīkla uzbūve**

Apmācība tiek veikta ar partijas izmēru kā 32, turpinot apmācību 20 epohos, saglabājot modeli posmos ar mazāko validācijas zuduma (*validation loss*) vērtību, mēģinot izvairīties no pārmērīgas pielāgošanas. Papildus tam apmācība tiek pārtraukta kad validācijas zudums turpina pieaugt 3 epohus pēc kārtas. Modeļi tiek pārbaudīti ar un bez priekšapstrādes (lielo burtu pārveidošana, simbolu un stopvārdu atmešana).

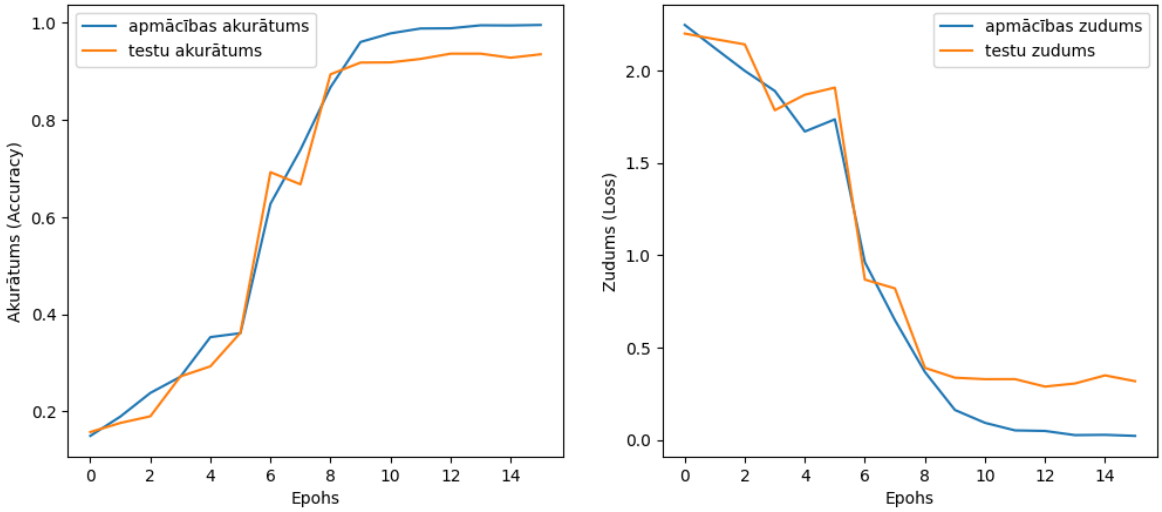
4.8. tabula

Veiktspējas mēri pielietojot LSTM neironu tīklu

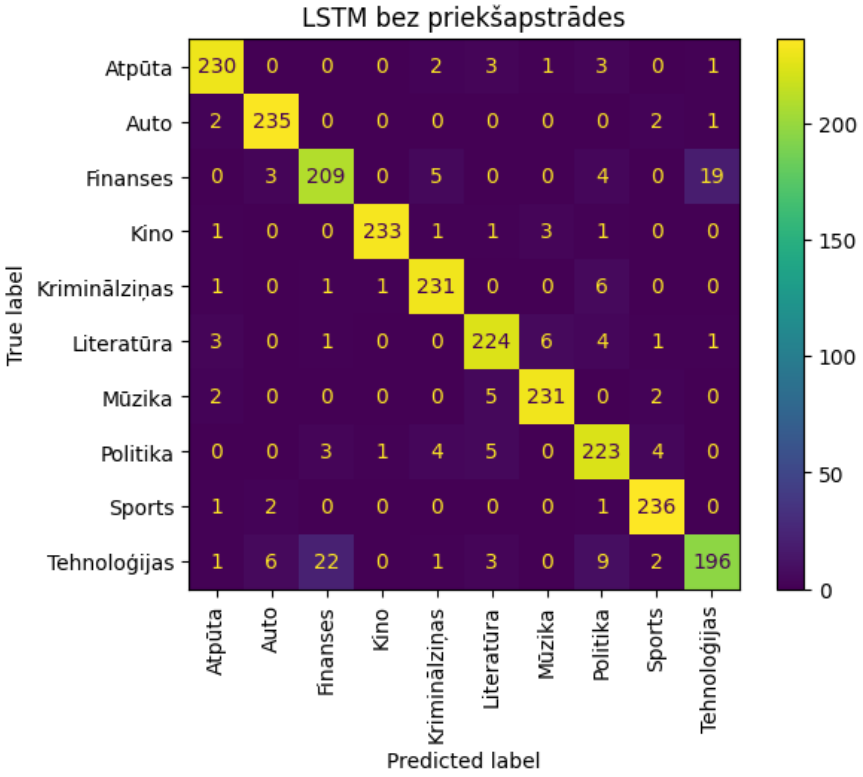
	Ar priekšapstrādi	Bez priekšapstrādes
LSTM	Ak.: 0.9329 F1: 0.9328	Ak.: **0.9367** F1: 0.9363

Pielietojot attēlā 4.7. ilustrēto LSTM neironu tīkla uzbūvi tiek iegūti rezultāti kā 4.8. tabulā. Pārpratuma matricu labākajam modelim iespējams apskatīt 4.9. attēlā, kur var

novērot līdzīgu kļūdu izplatību pa klasēm kā iepriekš apskatītos modeļos. Akurātuma un zuduma evolūciju pa epohiem iespējams apskatīt 4.8. attēlā.

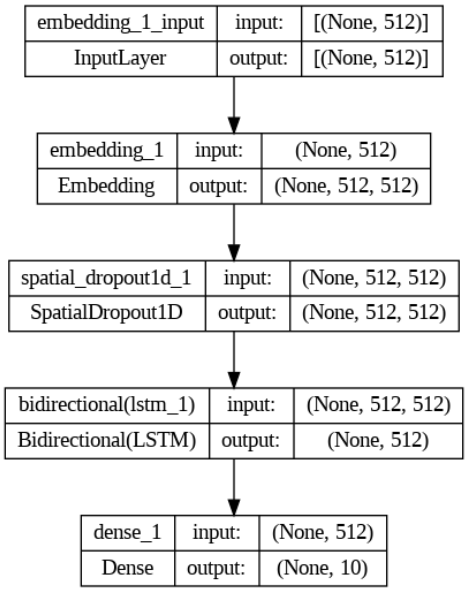


4.8. att. **LSTM neironu tīkls - novērtējums pa apmācības posmiem**



4.9. att. **LSTM neironu tīkls - pārpratuma matrica**

Apskatīta arī uzlabota LSTM arhitektūras pieeja, pielietojot LSTM slāņus divos virzienos, kas palīdz modelim gūt labāku konteksta un vārdu savstarpējo atkarību izpratni tekstā pirms un pēc konkrētā fragmenta.



4.10. att. **Divvirzienu LSTM neironu tīkla uzbūve**

Modelis kurš apmācīts ar augstākminēto arhitektūru nesniedza veiktspējas uzlabojumus, kā redzams 4.9. tabulā.

4.9. tabula

Veiktspējas mēri – divvirzienu LSTM

	Ar priekšapstrādi	Bez priekšapstrādes
Divvirzienu LSTM	Ak.: 0.9304 F1: 0.9301	Ak.: **0.9313** F1: 0.9311

Priekšapstrādes nozīme

Arī LSTM neironu tīkliem tika pārbaudīta priekšapstrādes nozīme, apskatot jau minētās arhitektūras ar un bez priekšapstrādes pielietošanas. Šo tīklu gadījumā labāki rezultāti tika sasniegti bez priekšapstrādes, kas ir sagaidāms rezultāts, jo svarīga LSTM tīklu priekšrocību ir konteksta un vārdu savstarpēju atkarību uztveres apmācība, kas kļūst sarežģītāka pēc saistošo vārdu un simbolu izņemšanas priekšapstrādes laikā.

LSTM neironu tīklu implementācijas kods pieejams 1. pielikuma 2. repozitorijā zem ”Zinu_klasifikacijas_LSTM.ipynb” faila. Izveidotie modeļi pieejami šī paša repozitorija apakšmapē ”models”.

4.3.3. 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 [30]. 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 [23] 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 [31], 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.

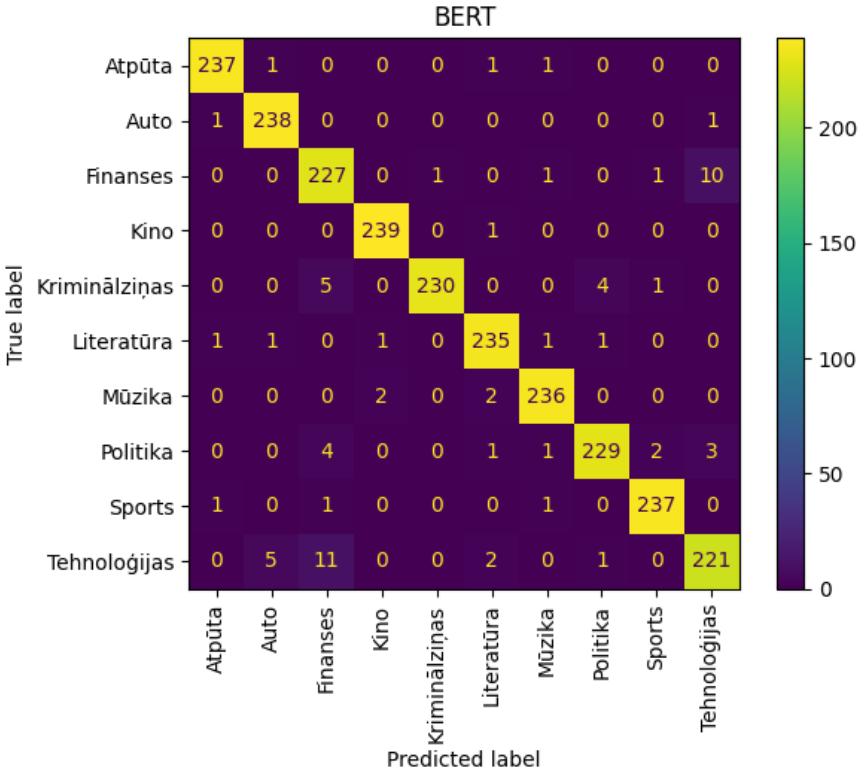
4.10. tabula

Veiktspējas mēri pielietojot BERT

Akurātums	F1
0.9704	0.9704

Rezultātā iegūtā modeļa raksturojošie mēri skatāmi 4.10. 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.



4.11. att. **BERT - pārpratuma matrica**

Papildus apskatot pārpratuma matricu 4.11. attēlā, BERT modelim var novērot līdzīgu klasifikācijas kļūdu sadalījums kā jau iepriekš apskatītajiem modeļiem.

LVBERT modeļa pielāgošanas kods pieejams 1. pielikuma 2. repozitorijā zem ”Zinu_klasifikacijas_BERT.ipynb” faila. Izveidotais modelis pieejams šī paša repozitorija apakšmapē ”models”.

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 0.9746 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 arhitektūras autora ieskatā bija jēgpilni, izveidotie modeļi nespēja sasniegt augstāku precizitāti par vienkāršākām mašīnmācīšanās pieejām. Daļēji tas izskaidrojams ar izvēlēto problēmu - galvenā neironu tīklu un lielo valodas modeļu priekrocība ir labāka valodas un konteksta izpratne, kas ir ļoti svarīgi veicot valodas apstrādes uzdevumus kā, piemēram, mašīntulkošana, tekstu ģenerēšana, kopsavilkumu veidošana. Klasifikācijā dziļa teksta izpratne ne vienmēr ir nepieciešama, lai labi klasificētu piederību kādai klasei.

Salīdzinot ar angļu valodas lielajiem valodas modeļiem kā BERT un XLNet, kuri pārsvarā tomēr uzrāda labākus rezultātus par klasiskiem klasifikācijas algoritmiem un iepriekšējām neironu tīkla arhitektūrām, LVBERT nespēja sniegt uzlabojumus autora izveidotajā datu kopā. Iegūtie rezultāti gan ierindoja to kā 2. labāko no apskatītajiem modeļiem, sniedzot labākus rezultātus arī par konvolūcijas neironu tīkliem.

Teksta priekšapstrāde latviešu valodā ir ierobežota morfoloģisko rīku vieglas pieejamības dēļ. Zināmus uzlabojumus priekšapstrādē autoram ir izdevies panākt ar paplašināta stopvārdu saraksta izveidi, tomēr metodes kā lemmatizācija nav viegli pielietojama datiem. Darba laikā gan secināts, ka priekšapstrāde kopumā neatstāj lielu ietekmi uz gala rezultātu, priekšapstrādes loma samazinās arī pielietojot modernākas metodes kā lielo valodu modeļus.

Priekšlikumi:

Autora ieskatā publiskas ziņu rakstu datu kopas, marķētas ar ziņu kategoriju piederību, 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 ar kategoriju marķējumiem netika atrastas, ziņu raksti pārsvarā pieejami tikai kā viena no sastāvdaļām lielākos valodas korpusos. Autora ievākto datu kopu publiskojot iespējama tālāka tās pielietošana citu autoru darbos, sniedzot labāku salīdzinājumu dazādu modeļu veiktspējā nākotnē uz vienādas datu kopas.

Iespējams veikt tālāku izpēte par neironu tīkliem un iespējām sasniegt augstāku precizitāti – autora izvēlētie slāņi un parametri tīkla izveidei, iespējams, nebija optimāli, labāku rezultātu varētu uzrādīt arī citas, darbā neapskatītas, tīklu arhitektūras pieejas.

LVBERT veiktspēja bija ļoti tuva labākajam apmācības modelim. Iespējams, apmācot BERT modeli uz lielāka latviešu valodas tekstu korpusa (līdzvērtīgu angļu valodas BERT modeļa apmācības kopai) vai pielietojot citu lielā valodas modeļa arhitektūru, būtu iespējams izveidot modeli ar augstāku precizitāti. Tas ir perspektīvs temats nākotnes pētījumiem.

IZMANTOTĀS LITERATŪRAS UN AVOTU SARAKSTS

[1] Papers With Code. Text classification comparison. https://paperswithcode.com/task/text-classification, 2024. [Tiešsaistē; Skatīts Apr. 24, 2024].

[2] Nacionālā korpusu kolekcija. LVK2022: Līdzsvarotais mūsdienu latviešu valodas tekstu korpuss. https://korpuss.lv/id/LVK2022, 2024. [Tiešsaistē; Skatīts Apr. 24, 2024].

[3] IBM. What is natural language processing (nlp)? https://www.ibm.com/topics/natural-language-processing, 2024. [Tiešsaistē; Skatīts Apr. 14, 2024].

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

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

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

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

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

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

[10] 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.

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

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

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

[14] 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.

[15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2015.

[16] Mathieu Cliche. BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 573–580, Vancouver, Canada, August 2017. Association for Computational Linguistics.

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

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

[19] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Tiešsaistē; Skatīts Apr. 20, 2024].

[20] Enes Zvornicanin. Differences between bidirectional and unidirectional lstm. https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm, 2021. [Tiešsaistē; Skatīts Apr. 20, 2024].

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

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

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

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

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

[26] Lucca de Freitas Santos and Murilo Varges da Silva. The effect of stemming and lemmatization on portuguese fake news text classification, 2023.

[27] Michal Toman, Roman Tesar, and Karel Jezek. Influence of word normalization on text classification. 01 2006.

[28] 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.

[29] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages, 2018.

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

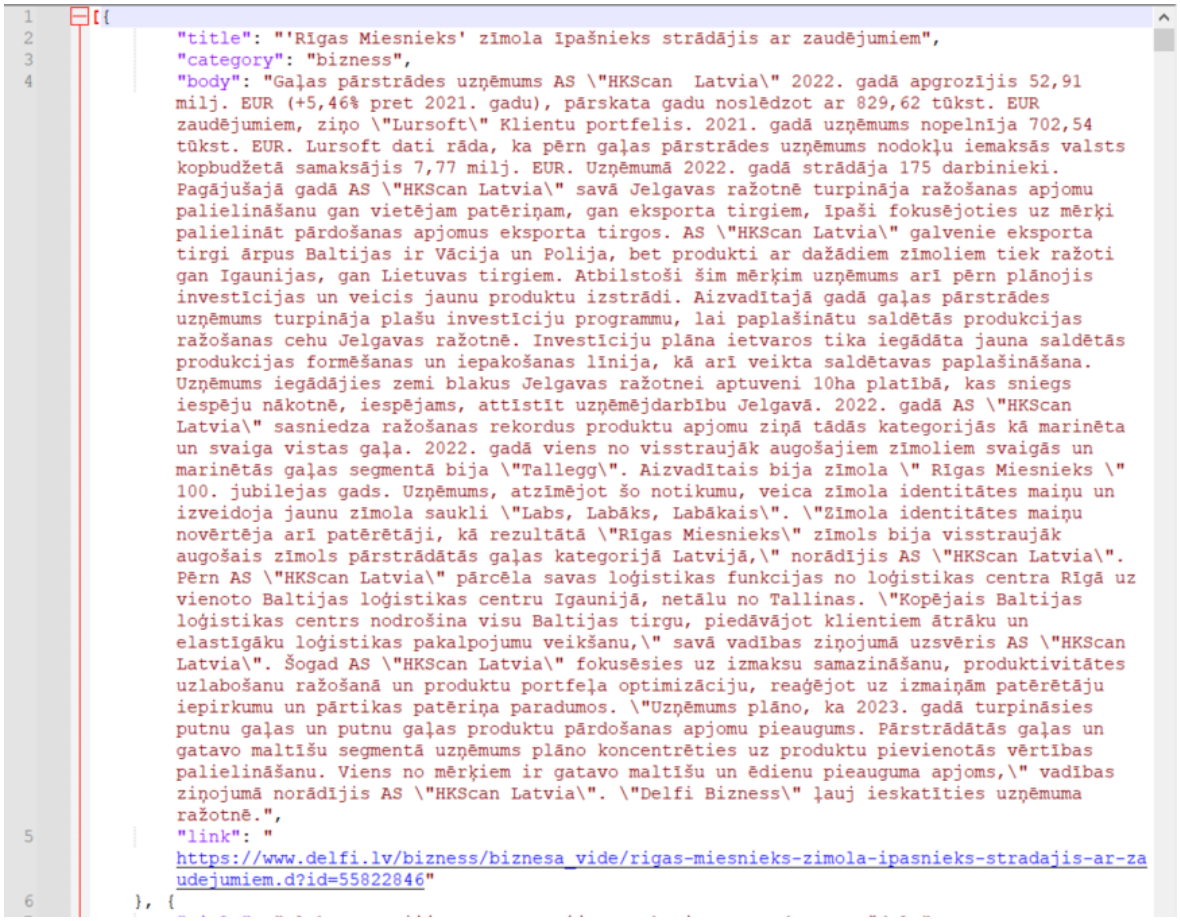
[31] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2020.

PIELIKUMI

1. pielikums. Darba koda repozitoriji

1. https://github.com/MatMatLV/delfinarija-zirneklis
2. https://github.com/MatMatLV/latvian-news-classification
3. https://github.com/MatMatLV/latvian-news-classification-web

2. 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"
}, {
```

3. pielikums. Loģistiskās regresijas, KNT un LSTM modeļu salīdzinājums - Xiang Zhang

Table 4: Testing errors of all the models. Numbers are in percentage. "Lg" stands for "large" and "Sm" stands for "small". "w2v" is an abbreviation for "word2vec", and "Lk" for "lookup table". "Th" stands for thesaurus. ConvNets labeled "Full" are those that distinguish between lower and upper letters

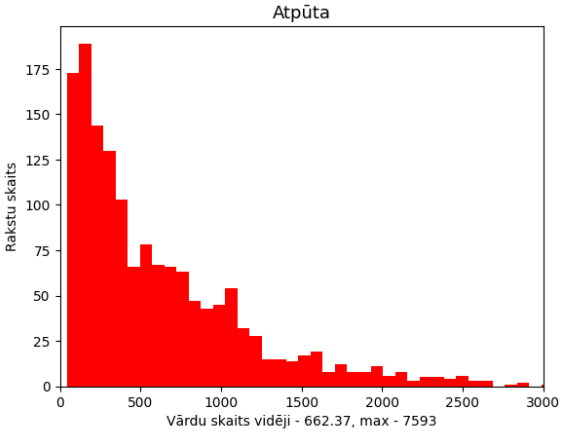
Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

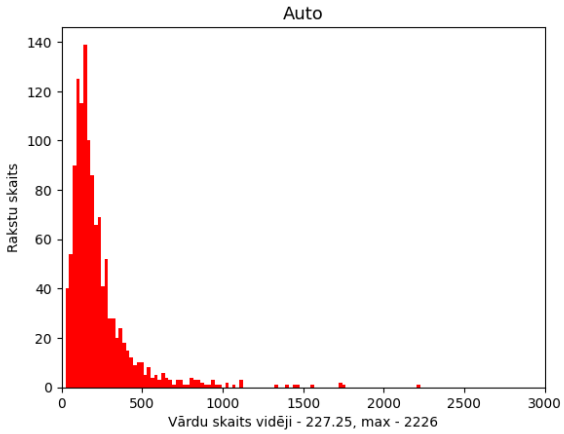
4. pielikums. Loģistiskās regresijas, KNT un LSTM modeļu salīdzinājums - Mathieu Cliche

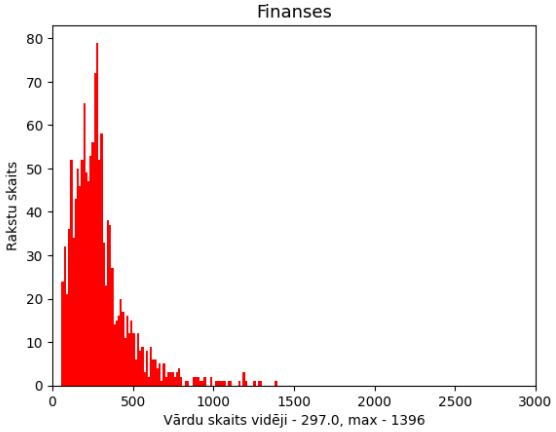
System	2013	2014	2015	2016
Logistic regression on 1-3 grams baseline	0.627	0.629	0.586	0.558
CNN (word2vec, convolution size=[3,4,5])	0.715	0.723	**0.688**	0.643
CNN (fasttext, convolution size=[3,4,5])	0.720	0.733	0.665	0.640
CNN (glove, convolution size=[3,4,5])	0.709	0.714	0.660	0.637
CNN (word2vec, convolution size=[1,2,3])	0.712	0.735	0.673	0.642
CNN (word2vec, convolution size=[5,6,7])	0.710	0.732	0.676	0.646
CNN (word2vec, convolution size=[3,4,5], no class weights)	0.682	0.679	0.659	0.640
CNN (word2vec, convolution size=[3,4,5], no distant training)	0.698	0.716	0.660	0.636
CNN (word2vec, convolution size=[3,4,5], no fully connected layer)	0.715	0.724	0.683	0.641
LSTM (word2vec)	0.720	0.733	0.677	0.636
LSTM (fasttext)	0.712	0.730	0.666	0.633
LSTM (glove)	0.710	0.730	0.658	0.630
LSTM (word2vec, no class weights)	0.689	0.661	0.652	0.643
LSTM (word2vec, no distant training)	0.698	0.719	0.647	0.629
LSTM (word2vec, no fully connected layer)	0.719	0.725	0.675	0.634
Ensemble model	0.725	**0.748**	0.679	**0.648**
Previous best historical scores	**0.728**	0.744	0.671	0.633

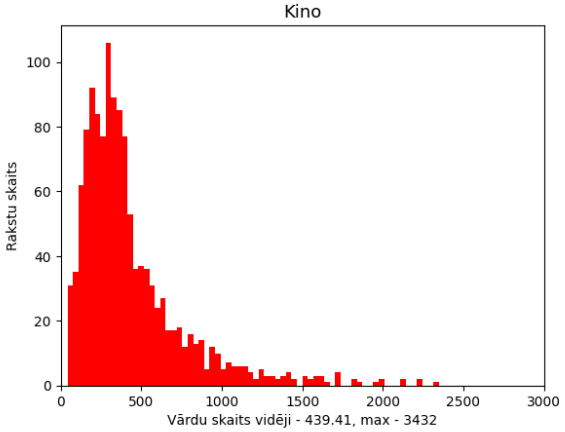
Table 1: Validation results on the historical test sets of subtask A. Bold values represent the best score for a given test set. The 2013 test set contains 3,813 tweets, the 2014 test set contains 1,853 tweets, the 2015 test set contains 2,392 tweets and the 2016 test set contains 20,632 tweets. Word2vec, fasttext and glove refer to the choice of algorithm in the unsupervised phase. No class weights means no weights were used in the cost function to counteract the imbalanced classes. No distant training means that we used the embeddings from the unsupervised phase without distant training. No fully connected layer means we removed the fully connected hidden layer from the network. Ensemble model refers to the ensemble model described in Sec. 3.4. The previous best historical scores were collected from (Nakov et al., 2016). They do not come from a single system or from a single team; they are the best previous scores obtained for each test set over the years.

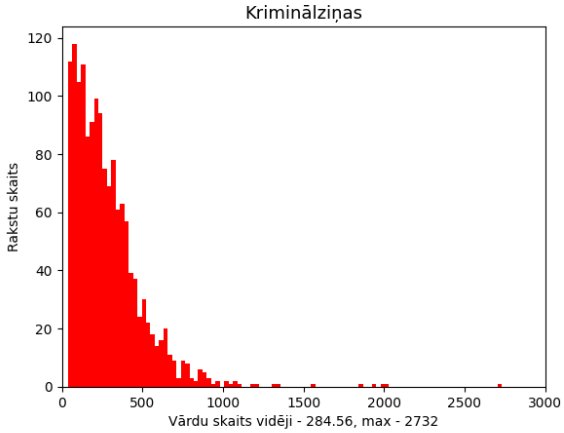
5. pielikums. Klasificējamo kategoriju rakstu garumi

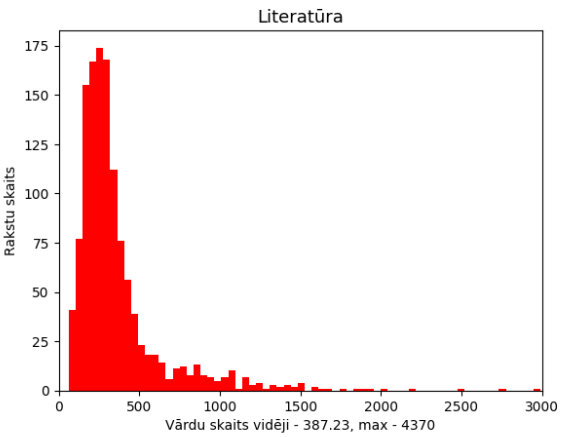


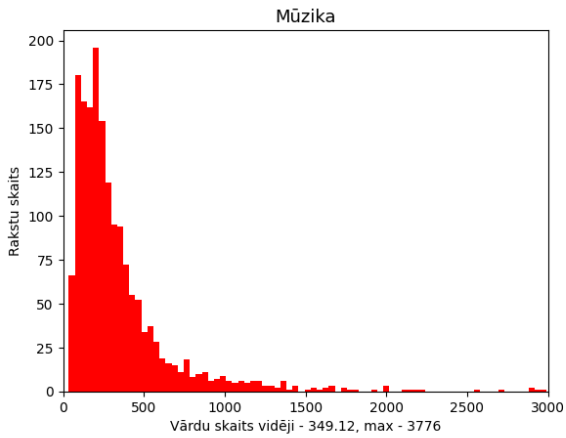


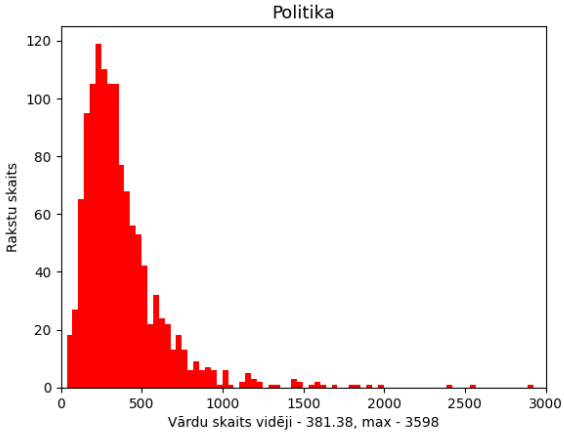


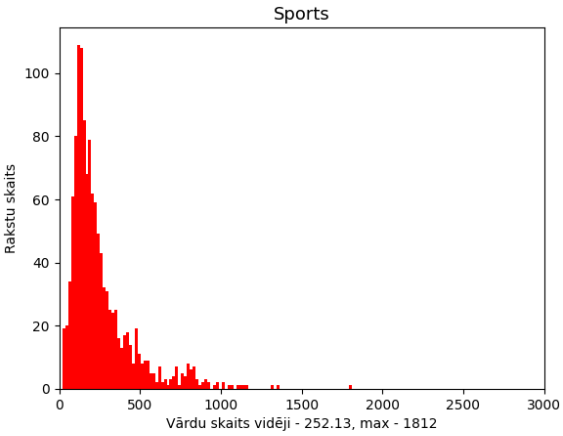


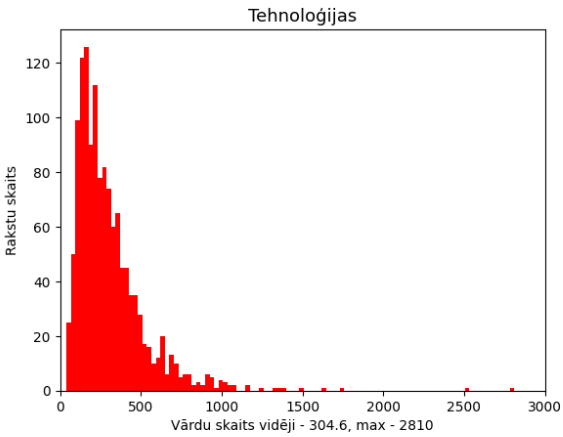





Mūzika
Rakstu skaits
Vārdu skaits vidēji - 349.12, max - 3776


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

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 _____._____