Rīgas 64. vidusskola

# *YouTube* video komentāru sadaļas sentimenta nolasīšana: *VADER* un *RoBERTa* modeļu salīdzinājums

Zinātniski pētnieciskais darbs: Datorzinātņu un datorlingvistikas sadaļā

**Darba autors:**

Rīgas 64. vidusskolas 12. klases skolnieks

Kristers Laganovskis

**Darba vadītājs:**

Rīgas 64. vidusskolas programmēšanas skolotājs

Edvards Bukovskis

Rīga 2024

# Anotācija

Zinātniski pētniecisko darbu: “YouTube video komentāru sadaļas sentimenta nolasīšana: VADER un RoBERTa modeļu salīdzinājums”. Izstrādājis autors Rīgas 64. vidusskolas 12.DIT klases skolēns Kristers Laganovskis. Darba vadītājs Rīgas 64. vidusskolas programmēšanas skolotājs Edvards Bukovskis.

Zinātniski pētnieciskajā darbā tika apskatīts, kā ar dabiskās valodas apstrādi (NLP) iegūst sentimenta analīzi. Kādas ir būtiskākās atšķirības starp *VADER* un *RoBERTa* modeļiem. Kā, tiek ievākti un apstrādāti dati. Cik precīzi ir iespējams novērtēt komentāru sadaļu, balsoties uz mākslīgo intelektu un programmēšanas palīdzību.

Teorētiskajā daļā tika detalizēti izpētīti dabiskās valodas apstrādes (NLP) pamati teksta sentimenta analizēšanai. Tika apskatīts, kas ir *VADER* modelis un kas ir *RoBERTa* modelis un to atšķirības, kā tiek veikta efektīva datu ievākšana un tās principi, kā arī datu apstrādes principi. Kā arī tika apsvērti sentimenta analīzes ierobežojumi.

Praktiskajā daļā tika salīdzināti divi sentimenta nolasīšanas modeļi VADER un RoBERTa. Autors izstrādāja pārlūka paplašinājums, kas spēj sadalīt YoTube video komentāru sadaļu trīs emociju klasēs.



Atslēgas vārdi: sentimenta analīze, pārlūka paplašinājums, dabiskās valodas apstrāde (NLP), *VADER*, *RoBERTa*.



# Abstract

Scientific Research Project: " Sentiment Analysis of YouTube Video Comments: Comparison of VADER and RoBERTa Models" Developed by Kristers Laganovskis, a student of Class 12.DIT at Riga 64th Secondary School. Supervised by Edvards Bukovskis, the programming instructor at Riga 64th Secondary School.

This scientific research delved into the realm of Natural Language Processing (NLP) to analyze sentiment. It meticulously examined the significant distinctions between the VADER and RoBERTa models, as well as the methodologies for data collection and processing. How accurately can the comment section be evaluated with the aid of artificial intelligence and programming? This was a central inquiry.

In the theoretical segment, the foundational aspects of Natural Language Processing (NLP) for text sentiment analysis were thoroughly explored. The VADER model and the RoBERTa model, along with their disparities, were scrutinized, alongside effective data collection and processing principles. Moreover, the constraints of sentiment analysis were deliberated upon.

In the practical phase, two sentiment analysis models, VADER and RoBERTa, were juxtaposed. The author developed a browser extension capable of categorizing YouTube video comments into three emotion classes.



Keywords: sentiment analysis, browser extension, Natural Language Processing (NLP), VADER, RoBERTa.

# Saturs

# Ievads

Mūsdienās informācijas plūsma ir pārāk bagātīga. Dažkārt tas var radīt problēmu gan saturu patērētājam, gan to veidotājiem. Būtisks šķērslis ir tas, ka cilvēki bieži vien vērš pārāk lielu uzticību satura veidotājiem, kas, kā rāda pieredze, var nebūt vienmēr objektīvi. Šāda pieeja rada risku, ka satura veidotājs varētu apslēpt vai pat manipulēt ar informāciju, ietekmējot patērētājus. Tādēļ ir ārkārtīgi svarīgi uzzināt, kā citi cilvēki novērtē konkrētā video saturu. Īpaši šo var novērot interneta platformā "*YouTube*", kur satura veidotāji bieži veicina savu subjektīvo viedokli, ko patērētāji bieži uztver kā objektīvu faktisko informāciju. Tomēr tas var radīt lielas problēmas, jo ne vienmēr šim veidotājam būs pareiza nostāja. Tāpēc ir svarīgi uzzināt, ko citi cilvēki domā par konkrēto viedokli vai saturu, un veikt pašam savus secinājumus. Veikt secinājumus, pārskatot pāris komentārus, ir ātri un ērti, bet, kad to ir simtiem vai tūkstošiem, šāda analīze prasa ievērojami vairāk laika. Tāpēc sentimenta analīze ir lietderīga, jo tā ļauj strukturēt komentāru sadaļu nostājās - pozitīvas, neitrālas vai negatīvas.

Šajā pētījumā tiks salīdzināti divi valodas apstrādes (NLP) modeļi, *VADER* un *RoBERTa*. Tiks analizētas metodes, kas tiek izmantotas, lai iegūtu nepieciešamos datus sentimenta nolasīšanai, kā šie dati ir jāapstrādā, lai sasniegtu veiksmīgu rezultātu. Papildus tam, tiks izveidots paplašinājums, kas ļaus ikvienam analizēt komentārus un tos filtrēt pēc noteiktām kategorijām.

**Mērķis:** Izpētīt dabiskās valodas (NLP) principus kā arī salīdzināt *VADER* un *RoBERTa* modeļus un pēc teorijas izpētes izveidot paplašinājumu, kas ļauj analizēt komentāru sadaļu kādam konkrētam *YouTube* video.

**Hipotēze:** Autoram izdosies veiksmīgi salīdzināt *VADER* un *RoBERTa* modeļus, kā arī analizēt un interpretēt *YouTube* video komentārus, izmantojot (NLP) modeļus un programmēšanu.

**Darba uzdevumi:**

1. Izpētīt (NLP) modeli un kā to pielietot sentimenta novērtēšanai.
2. Salīdzināt *VADER* un *RoBERTa* sentimenta modeļus.
3. Izpētīt, kā iegūt nepieciešamos datus no *YouTube* video komentāru sadaļas.
4. Izpētīt, kā apstrādāt datus efektīvi, lai tie sniegtu precīzāku novērtējumu.
5. Izstrādāt paplašinājumu, kas ļauj analizēt un filtrēt komentārus pēc noteiktām kategorijām.
6. Veikt secinājumus.

**Izmantotās darba metodes:** Literatūras apskats, lai varētu izpētīt, dabiskās valodas apstrādes (NLP). Salīdzināšanas metode, lai varētu salīdzināt *VADER* un *RoBERTa* (NLP) modeļus. Statistikas un analīzes metodes. Atvērtā pirmkoda pielietošana, *Python (3.8)* programmēšanas valodas pielietošana praktiskās daļas veikšanai.

**Darba struktūra** – Darbs sastāv no ievada, 2 nodaļām, 12 apakšnodaļām, secinājumiem, izmantoto informācijas avotu saraksta un pielikuma. Darbā ir 27 attēliem un divām tabulām.

# 1. Teorētiskā daļa

## 1.1. Dabiskās Valodas Apstrādes (NLP) Pamati

Dabiskās Valodas Apstrāde (angliski: natural language processing, turpmāk — NLP) ir mākslīgā intelekta joma, kas nodarbojas ar datoru spēju saprast, analizēt un interpretēt cilvēku valodu. Tās sākotnējie pētījumi sākās jau 1950. un 1960. gadu mijā, kad tika veidoti pirmie mēģinājumi izveidot datorprogrammas, kas spētu analizēt un interpretēt tekstu. Šajā laikā tika izveidots pirmās valodas analīzes algoritms, kas varēja identificēt vārdus, frāzes un sintaktiskos elementus. [1.] Kopš tā laika NLP ir piedzīvojis ievērojamu progresu, pateicoties tehnoloģiju un algoritmu uzlabojumiem. Ar NLP mūsdienu pasaulē saskaramies teju katru dienu, jo to pielieto plaši ikdienas ērtībās, ko ikkatrs lietojam. To izmanto balss vadības GPS sistēmas, digitālajos asistentos, runas-uz-tekstu dikcijas programmās, tulkošanas programmās, klientu apkalpošanas "čatbotos" un citās patērētāju ērtībās. [2.]

Tomēr NLP arvien lielāku lomu ieņem uzņēmējdarbības risinājumos, kas palīdz optimizēt uzņēmējdarbības darbības, palielina darbinieku produktivitāti un vienkāršo būtiskus uzņēmējdarbības procesus. [2.] Arvien biežāk NLP tiek izmantota arī medicīnas jomā, lai analizētu medicīnisko dokumentāciju un izgūtu noderīgu informāciju ārstiem un pētniekiem. [3.]

Sentimenta nolasīšana jeb analīze ir viena no svarīgākajām NLP apakšnozarēm. Tā ļauj noteikt teksta vai runas emocionālo noskaņu, palīdzot izsekot un analizēt viedokļus un emocijas. Sentimenta nolasīšana piedāvā plašas pielietošanas iespējas, tostarp sociālajos medijos, uzņēmējdarbībā, mārketingā, pētniecībā un citviet. [4.]

Sentimenta analīze tiek veikta, izmantojot datora logaritmus, kas analizē vārdus, frāzes un teikumus, lai noteiktu to emocionālo saturu. Šie algoritmi izmanto gan vārdnīcas un emociju leksikonus, gan mašīnmācīšanos, lai klasificētu vārdus un frāzes polaritātes: pozitīvs, negatīvs vai neitrāls. Tas ļauj novērtēt teksta noskaņu. [4.]

Sentimenta analīze ir īpaši noderīga, lai novērtētu un klasificētu lielus teksta apjomus, piemēram, sociālo mediju ierakstus, produktu atsauksmes vai klientu aptaujas. Tā palīdz uzņēmumiem iegūt ieskatu par to, kā viņu produkti vai pakalpojumi tiek uztverti tirgū, un sniedz iespēju uzlabot darbību, balstoties uz klientu atsauksmēm un viedokļiem. [5.]

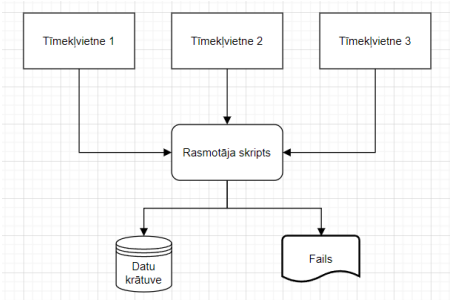
NLP ir kļuvusi par neatņemamu mūsdienu tehnoloģiju daļu, kas iesaistīta gan ikdienas patērētājus, gan uzņēmējdarbības risinājumus. Tā piedāvā iespēju saprast un analizēt cilvēku valodu, atvieglojot daudzas ikdienas darbības. Svarīga apakšnozare - sentimenta nolasīšana - kas palīdz novērtēt teksta emocionālo saturu un piedāvā plašas pielietošanas iespējas dažādās nozarēs.

## 1.2. Sentimenta nolasīšanas principi

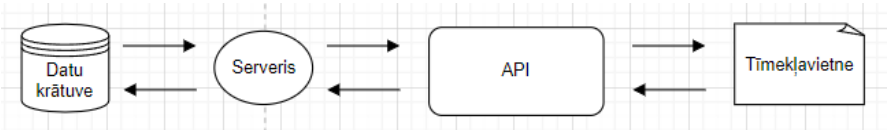
Kā jau iepriekš tika minēts, sentimenta nolasīšana ir viena no svarīgākajām NLP apakšnozarēm. Tāpēc šajā sadaļā tiks dziļāk apskatītas sentimenta analīzes pamatprincipi un tehnikas. Sentimenta nolasīšanas mērķis ir saprast un klasificēt tekstu vai runu, lai noteiktu tā emocionālo noskaņojumu. Sentimentu var klasificēt pēc polaritātes – pozitīvi, neitrāli un negatīvi, vai pēc vajadzības vēl mazākos iedalījumos. Sentimenta nolasīšanai, ir jāveic trīs būtiski soļi. Pirmais solis ir iegūt datus, otrais ir apstrādāt datus un trešais ir veikt datu analīzi jeb sentimenta nolasīšanu. [6.]

### 1.2.1. Datu ievākšana

Šis ir pirmais solis sentimenta nolasīšanā. Tas ietver iegūto datu kopumu, kuram tiks veikta sentimenta analīze. Šos datus var iegūt no ziņām, komentāriem, sociālo mediju ierakstiem, klientu atsauksmēm, klientu apkalpošanas centra datiem, darbinieku mijiedarbības datiem un citu veidu datiem, lai veiktu sentimenta nolasīšanu no tiem. [7.] Datu ievākšanu var veikt ar dažādām metodēm. Viens no populārākajiem veidiem ir lietot API *(Lietojumprogrammas saskarne)*, ko nodrošina sociālo mediju platformas, kas ļauj ievākt, datus straumējot. Piemēram: *Twitter API*, lai iegūtu ziņas ar noteiktām "heštegu" atsauksmēm, *News API*, lai izgūtu ziņas pēc kategorijām no dažādiem ziņu izdevējiem. Otrs populārs veids ir izmanot *web scraping (rasmošana)*, kas pārmeklē tīmekļa datus un savāc norādīto informāciju. Piemēram var izmantot *Python (3.8)* bibliotēku *BeautifulSoup*, lai iegūtu jebkādu informāciju no tīmekļa, piemēram, ziņu rakstus vai komentārus no blogiem, analizējot *HTML* tagus. Datus var ievākt arī izmantojot tīmekļa pārlūka paplašinājumus, ar kuru palīdzību lietotājs var izgūt informāciju no jebkuras publiskas tīmekļa vietnes un tad eksportēt datus vēlamā faila formātā. Piemēram: *Webscraper.io*. Kā arī datus var, ievākt no esošiem datu krājumiem, kas jau ir apstrādāti un ir gatavi lietošanai. Piemēram: *Rotten Tomatoes*, *IMDB* filmu apskats, *Yelp*, *Amazon* produkta apskats, *Twitter* tvīti un citas. [8.]




(1. attēls: Rasmošanas pamatprincipa shēma [28.])




(2. attēls: API pamatprincipa shēma [30.])

### 1.2.2 Datu apstrāde

Datu apstrāde: otrais svarīgais solis sentimenta noteikšanā ir iegūto datu apstrāde. Datu apstrāde ir kritisks solis sentimenta nolasīšanā, jo labi apstrādāti dati veido pamatu efektīvai sentimenta klasifikācijas modeļa izveidei. Teksta apstrāde ir process, kurā iepriekš iegūtie dati tiek attīrīti un apstrādāti. Teksta jeb datu apstrādē ietilpst tādi procesi, kā pieturzīmju izņemšana, bieži sastopamu vārdu noņemšana *(stop word removal),* visu tekstu pārveidojot, mazajos burtos *(lowercasing)*, lemmatizācija, tokenizācija *(tokenization)*, noliegumu apstrāde, intensitāšu apstrāde, "*emojie"* apstrāde, vektorizācija *(vectorization)* un retu vai zemu frekvenču vārdu apstrāde. [9.] Datu zinātniekiem ir teiciens *"garbage in, garbage out"*, tulkojumā *"atkritumi iekšā, atkritumi ārā."* Šī teiciena nozīme ir uzsvērt to, ka datu apstrāde ir ļoti svarīgs process sentimenta noteikšanā, jo gala rezultāts būs atkarīgs no tā, cik kvalitatīvi dati tiks padoti analīzes modelim. [10.]

Kā, jau autors minēja iepriekš, ir daudz un dažādas metodes, kā apstrādāt tekstu jeb datus. Taču, katra metode pilda noteiktu funkciju datu apstrādes procesā un ne vienmēr būs

būtiski izmantot visas metodes kopā. Tāpēc tagad tiks izskatītas pāris galvenās un nozīmīgākās metodes atsevišķi.

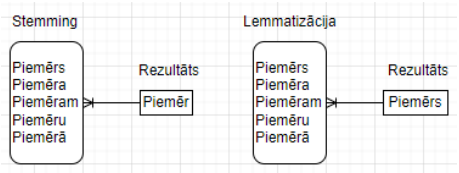
**Metode visa teksta pārveidošana mazajos burtos *(lowercasing)***: Viss teksts tiek pārveidots mazajos burtos, lai nodrošinātu vienmērību un novērstu iespējamu vārdu dublēšanos ar atšķirīgiem lielajiem un mazajiem burtiem. Piemēram: vārdi "Labs" un "labs" tiks uzskatīti par vienu un to pašu vārdu. [9.] Šo metodi izmanto, lai ietaupītu resursus, jo tas samazina vārdu skaitu, kas ir jāapstrādā. Kā arī strādāt, ar viena formāta vārdiem ir vieglāk, tas palīdz uzlabot precizitāti un vienkāršo algoritma darbu. Šo metodi izmanto, ja ir daudz vārdu, kas dublējas, taču nav vienā formātā un šo vārdu nozīme nemainās, kad tiek izmantoti lielie burti, piemēram, sociālajos mēdijos. Šo metodi var veikt ar *Python (3.8)* programmēšanas valodu.[11.]

**Metode tokenizācija *(tokenization)*:** viss teksts tiek sadalīts individuālos vārdos, kurus sauc par *tokeniem*, šis ir būtisks process datu apstrādē, jo tas ļauj standartizēt ievades datus mašīnmācīšanās modelī, padarot tos vieglāk apstrādājamus, lai analizētu teksta sentimentu. Jo bez tokenizācijas metodes sentimenta noteikšanas modelim būtu jāstrādā ar visu tekstu kā vienu virkni ar burtiem, kas prasītu lielus resursus un padarītu datus sarežģīti modelējamus. Šo metodi var veikt ar *Python (3.8)* bibliotēkas *nltk* palīdzību. [12.] Piemēram: teikums no ievāktajiem datiem "Šodien ir saulaina diena.", tiktu pārveidots pēc tokenizācijas uz "['Šodien', 'ir', 'saulaina', 'diena', '.']", taču, ja nebūtu piemērota toeknizācija un tiktu padoti ievades dati modelim bez tās, tad teksts izskatītos šādi - "Šodienirsaulinadiena." Šo metodi izmanto vienmēr, kad veic datu apstrādi sentimenta noteikšanai.

**Metode atbrīvošanās no visām pieturzīmēm:** no visa teksta tiek izņemtas ārā visas pieturzīmes un tagi. Tas iekļauj – komatus, punktus, izsaukuma zīmes un citas zīmes. Pieturzīmes nenes lielu informācijas vērtību sentimenta nolasīšanas modelim un tās var tikt noņemtas, lai samazinātu tekstā troksni. [11.]

**Metode bieži sastopamu vārdu noņemšana *(stop word removal)*:** bieži sastopami vārdi, piemēram, "un", "bet", "ir", utt. tiek noņemti, jo tie nenes lielu nozīmi, lai noteiktu sentimentu un pat var negatīvi ietekmēt sentimenta rezultātu. Šo metodi izmanto, kad veic datu apstrādi sentimenta noteikšanai. [11.]

**Metode lemmatizācija un pamatvārda noteikšana *(Lemmatization and Stemming)*:** lemmatizācija attiecas uz vārda pamatformas noteikšanu. Piemēram, vārda "ābols" lema joprojām būtu "ābols", bet vārda "ir" lema būtu "būt". [13.] Lemmatizācija ir pamats katram sentimenta nolasīšanas procesam. Tā ņem vērā dažādās esošās valodas struktūras, un katrā valodā lematizācija var būt atšķirīga. Pamatvārda noteikšana jeb *stemming* ir process, kurā vārdi tiek samazināti līdz to pamatformai, atgriežot tos vispārīgāko veidā. Piemēram, vārda "mājas" pamatvārds būtu "māj", un vārda "gāju" pamatvārds būtu "gāj". Pamatvārda noteikšana ir vienkāršāka un mazāk resursu prasīga nekā lematizācija, bet var būt mazāk precīza. Galvenais atšķirības punkts ir tas, ka lemmatizācija cenšas saglabāt vārda nozīmi, samazinot to līdz pamatformai, savukārt pamatvārda noteikšana vienkārši mēģina atgriezt vārdu vispārīgākajā formā. [14.]



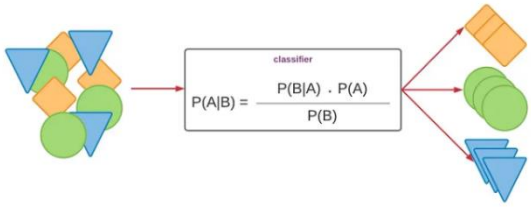
(3. attēls: Pamatvārda noteikšanas piemērs un lemmatizācijas piemērs shematiski [29])

**Metode negāciju apstrāde un intensifikatori *(handaling negations and intemsifiers)*:** noliegumu apstrāde ir metode, kas nosaka nolieguma ietekmes apjomu uz viedokļa vārdiem, ko ietekmē noliegums. Noliegumi ir vārdi kā "nav", "ne", "nekad", utt., kas ietekmē citu vārdu noskaņu frāzē. Noliegums var pilnībā mainīt citu vārdu noskaņu teikumā. Ir trīs noliegumu kategorijas: sintaktiskie, samazinātāji un morfoloģiskie noliegumi. [15.] Intensifikatori *(intemsifiers),* piemēram "ļoti" vai "ārkārtīgi", maina vārda nozīmi. Intensifikatori *(intemsifiers)* ir apstākļa vārdi, kas pastiprina citu izteicienu nozīmi. Par pastiprinātājiem parasti tiek lietoti tādi vārdi, kā "absolūti", "pilnīgi", "ārkārtīgi", "ļoti". Sentimenta analīzē noliegumu un intensifikatoru apstrāde ir būtiska, jo tā var mainīt teikuma noskaņu. [16.]

Šis ir tās metodes, kuras ir jāņem vērā, apstrādājot iegūtos datus. Lai gan, protams, ir daudzas citas metodes, kas īsumā tika pieminētas iepriekš, šīm izceltajām metodēm ir būtiska nozīme, lai nodrošinātu sentimenta nolasīšanu precīzāk un efektīvāk. Atkarībā no konkrētā uzdevuma prasībām optimālu rezultātu sasniegšanai var izmantot šo metožu kombināciju. Svarīgi atcerēties, ka metodes izvēlei ir jāsaskan ar konkrēto sentimenta noteikšanas mērķi.

### 1.2.3 Sentimenta nolasīšana

Sentimenta nolasīšana ļauj noteikt emocionālo noskaņu, sajūtu, vai tā būtu pozitīva, negatīva vai neitrāla. Lai veiktu sentimenta noteikšanu ar mašīnmācīšanās palīdzību, tiek izmantoti dažādi algoritmi un tehniskie risinājumi, piemēram, *Navie Bayes* algoritms. Iepriekš apstrādātie dati tiek padoti algoritmiem, kas analizē datus un veic sentimenta nolasīšanu. Šie algoritmi darbojas, apstrādājot tekstu un analizējot tā sastāvdaļas, lai saprastu, kāda ir tā emocionālā noskaņa. Katrā algoritmā ir savas stiprās un vājās puses. [16.] Kopumā ir trīs veida pieejas kā noteikt sentimentu: **uz noteikumiem balstīta *(rule-based)*, automātiskā *(automatic)* un hibrīds *(hybrid)***. [17.]




(4. attēls: *Navie Bayes* klasifikatora algoritms. To pielieto ne tikai sentimenta noteikšanā, bet arī citur. Klasificēt datus var, arī izmantojot mašīnmācīšanās ceļu, taču tā precizitāte pārsvarā ir tikai par 0,05% lielākā nekā izmantojot *Navie Bayes* varbūtības algoritmu, kas ir daudz ātrāks par mašīnmācīšanos, lai iegūtu tos pašus rezultātus. [31.])

**Uz noteikumiem balstīta pieeja *(rule-based)*:** šis ir praktisks veids, kā analizēt tekstu bez nepieciešamības apmācīt vai izmantot mašīnmācīšanās modeļus. Šī pieejas rezultātā tiek izstrādāti noteikumi, pēc kuriem teksts tiek iezīmēts kā pozitīvs/negatīvs/neitrāls. Šie noteikumi ir pazīstami arī kā leksikoni. Tāpēc šo pieeju sauc par leksikonu balstītu pieeju. Plaši izmantoti noteikumu balstīti risinājumi ir *TextBlob* modelis, *VADER* modelis, kā arī *SentiWordNet* modelis. [18.]

**Automātiskā pieeja *(automatic)*:** Atšķirībā no noteikumu balstītās metodes automātiskā metode nenosaka manuāli izstrādātus noteikumus, bet gan izmanto mašīnmācīšanās tehnikas. Emocionālās analīzes uzdevums parasti tiek modelēts kā klasifikācijas problēma. Klasifikators saņem tekstu un nosaka kategoriju, piemēram, pozitīvu,

negatīvu vai neitrālu. Piemēram, *RoBERTa* ir viens no populārākajiem modeļiem, kas izmanto automātisko pieeju. Kā arī polārākie algoritmi automātiskajai pieejai ir *Navie Bayes*, *Support Vector Machines (SVM)* un citi [19.]

**Hibrīds *(hybrid)* pieeja:** Hibrīda pieeja apvieno abu pieeju veidu, gan noteikumu balstītu, gan automātisko tehniku, integrējot tos vienā pieejā. Viena no nozīmīgākajām priekšrocībām izmantojot šo pieeju, ir tas, ka tā spēj sniegt precīzākus rezultātus. [19.]

Sentimenta precizitāti mēra ar dažādām metodēm, kā, piemēram: *F1-Score, Confusion Matrix, recall* un citām metodēm. Pieejamie sentimenta noteikšanas veidi sniedz dažādas iespējas, kā veikt sentimenta nolasīšanu. Piemērotāko pieeju var izvēlēties pēc projekta nosacījumiem un datu kvalitātes.

## 1.3 *VADER* un *RoBERTa* modeļu salīdzināšana

NLP Sentimenta nolasīšanas nozarē ir divi ievērojami modeļi – *VADER (Valence Aware Dictionary for Sentiment Reasoning)* un *RoBERTa (A Robustly Optimized BERT Pretraining Approach)*. *VADER* ir sentimenta lasīšanas modelis, kā arī algoritms, taču algoritms nav publiski nekur minēts. Modelis ir balstīts uz noteikumu pieeju *(rule-based approach)*. To ieviesa pētnieki C. J. Hutto un E. Gilberts savā darbā *"VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text",* tulkojumā - *"VADER: Taupīgs noteikumu balstīts modelis sociālo mediju tekstu sentimenta analīzei",* kas tika publicēts 2014. gadā. Modelis ir daļa no *Python (3.8)* bibliotēkas *nltk.sentiment* un tika speciāli radīts, lai strādātu ar sociālo tīklu tekstiem un to īpašībām piemēram, *"emoji"*, slengu un neformālās valodas lietojumu. *VADER* iedala sentimentu polaritātēs – negatīvs, pozitīvs un neitrāls. *VADER* darbības principi ietver vārdu kopu izmantošanu un to sentimenta vērtību analīzi, lai aprēķinātu kopējo teksta sentimentu. Turklāt modelis ņem vērā specifiskas īpašības, piemēram, negācijas un intensitāti, lai precīzi interpretētu tekstu. Tas ļauj *VADER* modelim veiksmīgi tikt galā ar sarežģītākiem teikumiem, saglabājot precizitāti. [25.]

2018. gadā *Google* izstrādāja atvērtā koda mašīnmācīšanās NLP modeli – *BERT (Bidirectional Encoder Representations from Transformers)*, taču šim modelim bija pāris ierobežojumi, tāpēc 2019. gadā *Facebook* komanda izstrādāja modificētu *BERT* modeli ar nosaukumu – *RoBERTa*. Šim modelim mērķis ir uzlabot oriģinālo *BERT* modeli, paplašinot to tā, lai labāk izmantotu transformatoru arhitektūru. *RoBERTa* izmanto automātisko pieeju apmācībā *(automatic approach)*. Tādējādi tiek radīta labāk izteikta un noturīgāka valodas reprezentācija. Ir pierādīts, ka ar *RoBERTa* modeli ir iespējams sasniegt izcilus rezultātus dažādos valodas apstrādes uzdevumos. Šis modelis tika trenēts, izmantojot plašu teksta datu kopu no vairākām valodām, kas ļauj tam saprast un ģenerēt tekstu dažādās valodās. *RoBERTa* ir viena no vadošajām NLP modeļu arhitektūrām, kas ir panākusi lielus sasniegumus dažādos valodas apstrādes uzdevumos. [26.]

Salīdzinot *VADER* un *RoBERTa,* var novērot, ka abiem modeļiem ir dažādas pieejas sentimenta nolasīšanai. *VADER* izmanto tradicionālu leksikona vērtējuma pieeju, kas balstīta uz noteikumiem *(rule-based approach*), un tas ir efektīvs, īpaši, strādājot ar sociālo tīklu tekstu un neformālu valodu. Savukārt *RoBERTa* ir jaunāka un uzlabota pieeja, kas balstās uz transformatoru arhitektūru. Tā nodrošina labāk izteiktu un noturīgāku valodas reprezentāciju, kas sniedz precīzākus rezultātus dažādos valodas apstrādes uzdevumos. Turklāt *RoBERTa* modelis ir apmācīts ar plašu teksta datu kopu no vairākām valodām, ļaujot tam saprast tekstu dažādās valodās. Tā kā *RoBERTa* ir jaunāka un pilnveidotāka modeļa versija, tai var būt priekšrocības, it īpaši, strādājot ar sarežģītiem un daudzveidīgiem valodas apstrādes uzdevumiem. [27.]

### 1.4 Datu ievākšana no *YouTube* video komentāru sadaļas

Datu iegūšana no kāda konkrēta *YouTube* video komentāru sadaļas ir svarīga, jo tā ļauj iegūt informāciju par sabiedrisko viedokli un reakcijām uz konkrētu video saturu. Tas var būt noderīgi, lai izprastu, kā cilvēki uztver noteiktu tēmu vai notikumu un reaģē uz to, kā arī, lai analizētu populārus viedokļus un tendences, kas izplatās sociālajā vidē. Pašlaik *YouTube* platformu lieto vairāk nekā divi miljardi lietotāju katru mēnesi. Tāpēc *YouTube* video komentāros esošo informāciju var izmantot, lai veidotu mašīnmācīšanās modeļus, veiktu lietotāju sentimenta nolasīšanu, radītu lielas datu kopas un veiktu datu analīzes. Tas piedāvā plašas iespējas gan akadēmiskajos pētījumos, gan komerciālajā jomā. No juridiskās perspektīvas datu ievākšana, piemēram, komentāri un citi atribūti, ir pilnīgi likumīgs process, ja tiek izmantots *YouTube data API* (*Application Programming Interface* jeb *Pielietojuma programmēšanas saskarne*). [20.]

Vispopulārākais veids, kā iegūt datus no kāda konkrēta video ir, izmantojot *YouTube data API v3*, ko piedāvā *Google*. *API* ir programmēšanas protokols vai interfeiss, kas ļauj vienai programmai (vai daļai programmatūras) komunicēt ar citu programmu vai komponenti. API nosaka, kādā veidā programmatūras komponentes vai sistēmas var mijiedarboties un kādas darbības tās var veikt. [21.] *YouTube* arī ir savs API, kuru var izmantot par velti un ar kura palīdzību ir iespējams iegūt dažādu informāciju no konkrēta *YouTube* video komentāru sadaļas. Kā, piemēram, ir iespējams iegūt komentārus, komentāru *"patīk"* balsu skaitu, komentāru atbildes, komentētāju lietotājvārdus un citus datus par komentāru sadaļu. [22.] Šo *API* var izmanot ar *Python (3.8)* un tās piedāvātajām bibliotēkām, un datus izvadīt *JSON* failā.

Taču ir arī citas metodes, piemēram, rasmošana *(web scraping)*. Rasmošana ir process, kurā programmētājs izmanto programmatūru, lai automātiski iegūtu informāciju no interneta lapām. Tas ietver HTML kodu analīzi un datu iegūšanu no tīmekļa vietnēm. Rasmošana var būt noderīga gadījumos, kad API nav pieejams vai neattiecas uz konkrēto informāciju, ko vēlaties iegūt. [23.] Tomēr ir svarīgi uzsvērt, ka rasmošana var pārkāpt *YouTube* lietošanas noteikumus specifiskos gadījumos. Ja tiek apskatīta konkrēta *YouTube* video komentāru sadaļa, tad nav optimāli izmantot rasmošanas metodi, jo datu iegūšana var prasīt ilgu laiku tāpēc, ka šai programmai jāapstrādā katrs komentārs atsevišķi. Taču, ja izvēlas lietot šo metodi, tad to var pilnveidot ar *Python (3.8)* bibliotēku *selenium* un tad datus ievākt *JSON* failā. [24.]

# 2. Praktiskā daļa

Autors praktisko daļu sadalīja divās daļās. Pirmajā salīdzina un pielieto *VADER* un *RoBERTa* modeļus, bet otrajā izveido *Google Chrome* paplašinājumu, kas ļauj klasificēt kāda konkrēta *YouTube* video komentāru sadaļu trīs kategorijās – pozitīvi, neitrāli un negatīvi.

Kā jau iepriekš teorijas daļā autors apskatīja divus sentimenta nolasīšanas modeļus *VADER* un *RoBERTa*. Praktiskajā daļā autors veica eksperimentus, kas ļāva salīdzināt šos modeļus pēc noteiktām mērsistēmām - ***accuracy, recall, precision, F1-Score un confusion Matrix.*** Abus modeļus autors salīdzināja ar trīs dažādu apjomu datu kopām - 10 komentāru, 100 komentāru un 3534 komentāru datu kopu.

Lai salīdzinātu *VADER* un *RoBERTa* modeļus, ir svarīgi izprast, ar kādām mērsistēmām autors veicis mērījumus, kā arī izprast, ko katra mērsistēma dara jeb mēra. Precizitāte jeb ***accuracy***, tā nosaukums jau izsaka mērsistēmas būtību. Precizitāte mēra modeļa pareizo prognožu procentuālo daļu. To aprēķina, dalot pareizo prognožu skaitu ar kopējo modeļa veikto prognožu skaitu. Tā formula: Precizitāte = (pareizo prognožu skaits / kopējais prognožu skaits) * 100. Piemērs: ja modelis pareizi paredz 80 no 100 paraugiem, tā precizitāte ir 80%. [32.]

Atgriezeniskā izsaukšana jeb ***recall***, kas pazīstama arī kā jutīgums, mēra patieso pozitīvo prognožu īpatsvaru starp visiem faktiskajiem pozitīvajiem paraugiem datu kopā. To aprēķina, dalot patieso pozitīvo rezultātu skaitu ar patieso pozitīvo un nepatieso negatīvo rezultātu summu. Tā formula: Jūtīgums = patiesie pozitīvie rādītāji / (patiesie pozitīvie rādītāji + viltus negatīvie rādītāji). Piemēram: ja teikums ir, pozitīvs un to modelis nolasa kā pozitīvu, tad to uzskata par patiesi pozitīvu teikumu, taču, ja modelis to nolasa par negatīvu, tad tas ir viltus negatīvs teikums. [32.]

Precīzums jeb ***precision*** izklausās līdzīgi taču būtība mazliet mainās. Precīzums fokusējas uz pozitīvajām prognozēm, ko veicis modelis. Tā ir attiecība starp patiesi pozitīvajām prognozēm un kopējo pozitīvo prognožu skaitu. Tā formula: precīzums = patiesie pozitīvie rezultāti / (patiesie pozitīvie rezultāti + viltus pozitīvie rezultāti). Piemēram, analizējot sentimentu komentāru sadaļai kādai filmai. Modelis prognozē, ka 80 no 100 komentāriem ir patiesi pozitīvi, bet 20 ir kļūdaini pozitīvi, tad tā precīzuma vērtējums būtu 0,8. [32.]

F1-rādītājs jeb ***F1-Score*** ir vidējais balanss starp precizitāti (precision) un jūtību (recall) un to izmanto, lai līdzsvarotu šīs divas mērsistēmas. Šī mērsistēma tiek izmantota situācijās, kad ir nepieciešams novērtēt modeļa veiktspēju, ņemot vērā gan precizitātes, gan jūtības nozīmi. Tās formula: F1-rādītājs = 2 * ((precizitāte * atsaukšana) / (precizitāte + atsaukšana)). Piemēram, ja modelim ir augsta precizitāte (***precision***), bet zems jūtīgums (***recall***), tas nozīmē, ka tas rada mazāk kļūdaini pozitīvu rezultātu, bet izlaiž daudz patiesi pozitīvu rezultātu. Turpretī modelis ar augsta jūtīguma (***recall***) spēju, bet zemu precizitāti (***precision***) rada vairāk kļūdaini pozitīvu rezultātu, bet uzrāda vairāk patiesi pozitīvu rezultātu. Šādos gadījumos F-1 rādītājs var palīdzēt noteikt, kurš modelis ir labāks. [32.]

Apjukuma matrica jeb ***confusion Matrix*** ļauj vizualizēt rezultātus. Tā ir tabula, kurā norādīts modeļa prognozēto patieso pozitīvo, patieso neitrālo, patieso negatīvo, viltus pozitīvo, viltus neitrālo un viltus negatīvo rezultātu skaits. Sajukuma matrica palīdz novērtēt, cik veiksmīgi modelis atšķir divas vai vairāk klases. [32.]

## 2.1 Datu kopas izvēle

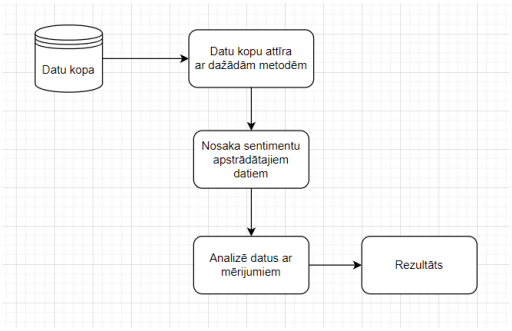
Lai testētu jebkāda veida modeli, ir jābūt datiem, ar ko to testēt. Autors izvēlējās testēt modeli ar īsa teksta datiem, kas līdzinās komentāriem, jo autors darba gaitā veidos, *Google Chrome* paplašinājumu, kas ļaus klasificēt kāda konkrēta *YouTube* video komentāru sadaļu. Pamatojoties uz to, autors izvēlējās datu kopu no pētījuma "*Twitter Sentiment Analysis Datasets: A Valuable Resource for Social Media Research*". Šajā datu kopā ir iekļauti 3534 komentāri, un katram no tiem ir piešķirts savs noskaņojums. (skatīt pielikumā – 6.) [33.]

| | A | B |
|---|---|---|
| 1 | Comment | Label |
| 2 | Last session of the day http://twitpic.com/67ezh | Neutral |
| 3 | Shanghai is also really exciting (precisely -- skyscrapers galc | Positive |
| 4 | Recession hit Veronique Branquinho, she has to quit her cor | Negative |
| 5 | happy bday! | Positive |
| 6 | http://twitpic.com/4w75p - I like it!! | Positive |
| 7 | that`s great!! weee!! visitors! | Positive |
| 8 | I THINK EVERYONE HATES ME ON HERE lol | Negative |
| 9 | soooooo wish i could, but im in school and myspace is com | Negative |
| 10 | and within a short time of the last clue all of them | Neutral |

(5. attēls: Daļa no datu kopas: "*Twitter Sentiment Analysis Datasets: A Valuable Resource for Social Media Research*", kur var novērot datu kopas struktūru.)

## 2.2 *VADER* modeļa testēšana

Autors visu praktisko daļu veica *Visual Studio Code* vidē, kā arī modeļu testēšanai izmantoja *Python (3.8)* programmēšanas valodu. Pirmkods ir pieejams pielikumā ar nosaukumu "*VADER*-testēšana" (skatīt pielikumā – 1.). *VADER* testēšana tiks veikta šādos soļos: datu kopas atvēršana, datu attīrīšana, sentimenta nolasīšana, mērījumu veikšana un rezultātu parādīšana. (skatīt 6. attēlā)




(6. attēls: *VADER* modeļa testēšanas shēma)

Pirmais solis - datu kopas atvēršana: (skatīt 7. attēlā)

```
with open("praktiskais\\scores_of_models\\dataset.csv", newline='') as f:
    reader = csv.reader(f)
    next(reader, None)
    rows = list(reader)
```

(7. attēls: Datu kopas atvēršana, *Python (3.8)*)

Septītajā attēlā var novērot, kā autors atver datu kopu ar *"csv.reader()"* funkciju, kas ir viena no "csv" bibliotēkas funkcijām, tad autors izmanto *"next"* funkciju, lai izlaistu pirmo datu kopas rindu. Tas ir nozīmīgi, jo modelim nevajag padot nekādu lieku informāciju. Pēc tā tiek definēts *"rows"*, kas pārtaisa katru rindu vienā lielā sarakstā.

Otrais solis – datu kopas attīrīšana: (skatīt 8. attēlā)

```
def clean(comment):
    comment = re.sub(r'http\S+', '', comment)
    comment = re.sub(r'[^a-zA-Z\s]', '', comment)
    comment = re.sub(r'@\w+', ' ', comment)
    comment = re.sub(r'<href+', '', comment)
    comment = re.sub(r'(\s+)', ' ', comment)
    comment = re.sub(r'\[[^]]*\]', ' ', comment)
    return comment.lower()
```

(8. attēls: Datu kopas attīrīšana, *Python (3.8)*)

Astotajā attēlā var novērot *"clean()"* metodi, kurā tiek attīrīti dati ar *"re.sub()"* metodi, kura ir viena no *"re"* bibliotēkas metodēm. Datu kopa tiek attīrīta septiņos dažādos veidos: *"'http\S+'"* attīra visus datus no linkiem, *"'[^a-zA-Z\s]'"* atbrīvojas no visiem simboliem, ciparīem un burtiem, kas nav angļu valodas alfabētā, *"'@\w+'"* tiek noņemti lietotājvārdi, kuriem sākumā ir simbols *"@"*, *'<href+'* attīra no vārda *"<href"* datu kopā, *"'(\s+)'"* atbrīvojas no liekajām atstarpēm, *"'\[[^]]*\]'"* atbrīvojas no teksta, kas atrodas starp kvadrāta iekavām. Tad visbeidzot tiek viss teksts samazināts uz mazajiem burtiem ar "comment.lower()" funkciju. Kā var ievērot, autors neizmantoja dažas metodes datu apstrādei, kā piemēram, – tokenizāciju (tokenization), negāciju un intensifikātoru metodes, jo *VADER* izmanto iepriekš

izveidotu leksikonu vārdnīcu, lai veiktu sentimenta analīzi. Tas neiesaista mašīnmācīšanos no teksta struktūras, kā to dara mašīnmācīšanās modeļi, kā, piemēram, *RoBERTa*. Tāpēc arī tokenzācija būtu lieks process, jo to izmanto modeļiem, kas analizēt tekstu ar mašīnmācīšanās palīdzību. Taču negācijas un intensifikātori ir jau iestrādāti modeļa leksikonu vārdnīcā. Visbeidzot viss teksts tiek samazināts uz mazajiem burtiem.

Trešais solis – sentimenta noteikšana attīrītajiem datiem: (skatīt 9. attēlā)

```
actual_labels = []
predicted_labels = []
sid = SentimentIntensityAnalyzer()

for row in rows:
    comment = row[0]
    actual_label = row[1]
    comment = clean(comment)
    sentiment = sid.polarity_scores(comment)['compound']
    if sentiment >= 0.05:
        predicted_label = 'Positive'
    elif sentiment <= -0.05:
        predicted_label = 'Negative'
    else:
        predicted_label = 'Neutral'
    actual_labels.append(actual_label)
    predicted_labels.append(predicted_label)
```

(9. attēls: sentimenta nolasīšana attīrītajiem datiem, *Python (3.8)*)

Devītajā attēlā ir novērojams, ka autors izveido divus sarakstus *"actual_label"* jeb "īstais sentiments" un *"predicted_lable"* jeb "pareģotais sentiments". Pēc tam tiek izveidots *"for"* cikls, ar kura palīdzību iziet cauri katrai datu kopas rindai, iegūstot komentārus un to faktiskos apzīmējumus jeb īsto sentimentu. Tad tiek palaista datu attīrīšanas metode *"clean"*, kura attīra katru rindu jeb komentāru. Pēc teksta datu attīrīšanas tas aprēķina sentimenta rezultātu, izmantojot *"SentimentIntensityAnalyzer"* jeb *VADER* modeli, kas ir no *"nltk"* bibliotēkas. Pamatojoties uz šo rezultātu, komentāri tiek iedalīti pozitīvos, negatīvos vai neitrālos komentāros. *VADER* modeļa rezultātu nosaka no polaritātes skalas, kur vērtības tiek izteiktas no -1 līdz 1. 1 norāda uz ļoti pozitīvu sentimentu, -1 norāda uz ļoti negatīvu sentimentu, un 0 norāda uz neitrālu sentimentu. Ja rezultāts ir lielāks par 0.05, tad komentārs tiek iedalīts kā "pozitīvs", taču, ja mazāks par -0.05, tad tas tiek iedalīts kā "negatīvs". Ja rezultāts ir starp -0.05 un 0.05, tad komentārs tiek iedalīts kā "neitrāls". Visbeidzot pareģotais un īstais sentiments tiek saglabāts attiecīgajos sarakstos.

Ceturtais solis – mērījumu veikšana un to rezultāti:

```
accuracy = accuracy_score(actual_labels, predicted_labels) * 100
precision = precision_score(actual_labels, predicted_labels, average='weighted', zero_division=1) * 100
recall = recall_score(actual_labels, predicted_labels, average='weighted', zero_division=1) * 100
f1 = f1_score(actual_labels, predicted_labels, average='weighted', zero_division=1) * 100
overall = (accuracy + precision + recall + f1) / 4
```

(10. attēls: mērījumu veikšana - *VADER*, *Python (3.8)*)

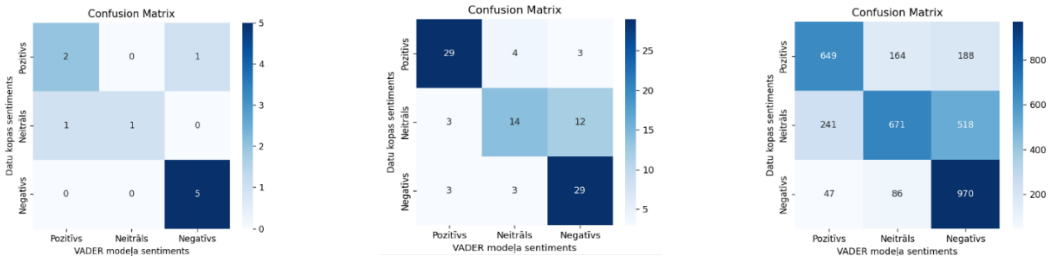
Desmitajā attēlā autors veica mērījumus ar pieciem mērījumu veidiem - *"accuracy"* jeb precizitāti, *"precision"* jeb precīzumu, *"recall"* jeb jūtīgumu, *"F1-score"* jeb F1 rezultātu un visbeidzot ar vispārīgo novērtējumu.

Autors pēc programmas izstrādes ieguva šādus rezultātus, testējot dažādus komentāru apjomus no "*Twitter Sentiment Analysis Datasets: A Valuable Resource for Social Media Research*" datu kopas.

*Tabula 2.2.1.*

| Mērsistēmas | 10 komentāri | 100 komentāri | 3534 komentāri |
|---|---|---|---|
| Accuracy (%) | 80.00% | 72.00% | 64.80% |
| Precision (%) | 81.67% | 72.23% | 67.16% |
| Recall (%) | 80.00% | 72.00% | 64.80% |
| F1 Score (%) | 78.79% | 71.34% | 63.86% |
| Kopumā (%) | 80.11% | 71.89% | 65.15% |

Autors arī veica apjukuma matricas jeb ***confusion Matrix*** mērījumus:

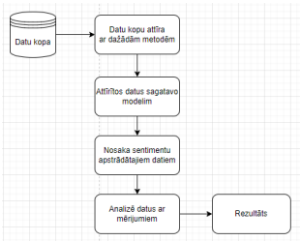



(11., 12., 13. attēli: Ir redzemas ***confusion Matrix*** tabulas, kur 11. attēlā ir 10 komentāru rezultāts, 12. attēlā 100 komentāru rezultāts un 13. attēlā 3534 komentāru rezultāts)

Apjukuma matricā var novērot, ka *VADER* modelis ir centies būt precīzs, taču ne pietiekami, jo var novērot, ka neveidojas lineāra taisne no tumši zilajiem toņiem. Komentāru apjomam pieaugot precizitāte, kā arī visi pārējie mērījumi progresīvi samazinājās. Tas ir izskaidrojams ar to, ka, komentāru skaitam pieaugot, rodas vairāk iespēju modelim nolasīt aplamus jeb viltus sentimentus.

## 2.3 *RoBERTa* modeļa testēšana

*RoBERTa* modeļa testēšana ir līdzīga *VADER* modeļa testēšanai, taču *RoBERTa* nav balstīts uz leksikoniem, bet gan uz mašīnmācīšanos, tāpēc būs būtiskas atšķirības datu sagatavošanā un programmas izstrādē, tādēļ autors iesaka izpētīt pievienoto pirmkodu pielikumā "*RoBERTa*-testēšana" (skatīt pielikumā – 2.). *RoBERTa* testēšana tiks veikta šādos soļos: datu kopas atvēršana, datu attīrīšana, datu sagatavošana modelim, sentimenta nolasīšana, mērījumu veikšana un rezultātu parādīšana. (skatīt 14. attēlā)




(14. attēls: *RoBERTa* modeļa testēšanas shēma)

Pirmais un otrais - datu kopas atvēršana un datu attīrīšana ir identiski *VADER* modeļa testēšanai. Trešais solis – datu sagatavošana modelim: atšķirībā no *VADER* modeļa, kur nevajadzēja veikt tokenzāciju jeb *tokenization*, šeit gan tā būs būtiska, jo tas palielina modeļa precizitāti un tas ir formāts, kā *RoBERTa* modelis saprot padotos datus. (skatīt 15. attēlā)

```
inputs = self.tokenizer(comment, return_tensors='pt', truncation=True, max_length=512)
```

(15. attēls: Attīrīto datu tokenizācija. *"AutoTokenizer.from_pretrained()"* metode no *"Hugging Face Transformers"* bibliotēku, *Python (3.8)*)

Ceturtais solis - sentimenta nolasīšana: Šis solis atšķiras būtiski no *VADER* modeļa sentimenta nolasīšanas, ja *VADER* modelim to varēja izdarīt ar vienu rindu kodu, tad šeit tas aizņems vairākas. Sākumā ir svarīgi atgādināt, ka tiek izmantots jau iepriekš uztrenēts modelis, jo resursu un laika patēriņa dēļ nebija objektīvi trenēt savu mašīnmācīšanās modeli. Autors izvēlējās veikt testēšanu uz *Twitter* komentāru datu kopu, tāpēc autors izvēlējās piemeklēt attiecīgi sagatavotu iepriekš uztrenētu *RoBERTa* modeli - *"cardiffnlp/twitter-roberta-base-sentiment"*.

```
class SentimentAnalyzer:
    MODEL = "cardiffnlp/twitter-roberta-base-sentiment"
    LABEL_MAPPING_LINK = "https://raw.githubusercontent.com/cardiffnlp/tweeteval/main/datasets/sentiment/mapping.txt"

    def __init__(self):
        self.tokenizer = AutoTokenizer.from_pretrained(self.MODEL)
        self.labels = self._download_label_mapping()
        self.model = AutoModelForSequenceClassification.from_pretrained(self.MODEL)
```

(16. attēls: Kā tiek definēts uztrenētais *RoBERTa* modelis "cardiffnlp/twitter-*roberta*-base-sentiment", autors ir inicializējis iepriekš uztrenēto modeli, kā arī tokenizatoru, ielādējis sentimenta marķējumus *"LABEL_MAPPING_LINK"* un izveidojis konstruktoru *"__init__(self)", Python (3.8).*

```
def analyze_sentiment(self, comment):

    inputs = self.tokenizer(comment, return_tensors='pt', truncation=True, max_length=512)
    output = self.model(**inputs)
    scores = output.logits[0].detach().numpy()
    scores = softmax(scores)
    ranking = np.argsort(scores)
    ranking = ranking[::-1]
    top_label = self.labels[ranking[0]]

    return top_label
```

(17. attēls: Izveidota metode *"analyze_sentiment(self, text)"*, *Python (3.8)*)

17. attēla var novērot, kā tiek nolasīts sentiments no katras datu kopas rindas jeb komentāra. *RoBERTa* modelis savus rezultātus arī pasniedz polaritātes skalā, taču katram komentāram tiek piemērotas trīs polaritātes – pozitīvo, neitrālo un negatīvo. Piemēram, teikumam *"Laiks ir jautrs, bet diena ir grūta."* Šādam teikumam būs trīs vērtējumi – pozitīvs: 0.5, neitrāls: 0.3 un negatīvs: 0.2. Augtākais vērtējumus ir teikuma sentiments, šajā gadījuma teikums tiktu novērtēts "pozitīvs" ar 0.5 pozitīvu vērtējumu. Tad kods atgriež atpakaļ *"top_label"* jeb augstāko vērtējumu, kas pēc tam tiek sadalīts sarakstos - pozitīvi, neitrāli, negatīvi.

Piektais solis – mērījumu veikšana un to rezultāti:

```
accuracy = (accuracy_score(actual_labels_lower, predicted_labels_lower)) * 100
precision = (precision_score(actual_labels_lower, predicted_labels_lower, average='weighted', zero_division=1)) * 100
recall = (recall_score(actual_labels_lower, predicted_labels_lower, average='weighted', zero_division=1)) * 100
f1 = (f1_score(actual_labels_lower, predicted_labels_lower, average='weighted', zero_division=1)) * 100
overall = (accuracy + precision + recall + f1)/4
```

(18. attēls: mērījumu veikšana - *RoBERTa*, *Python (3.8)*)

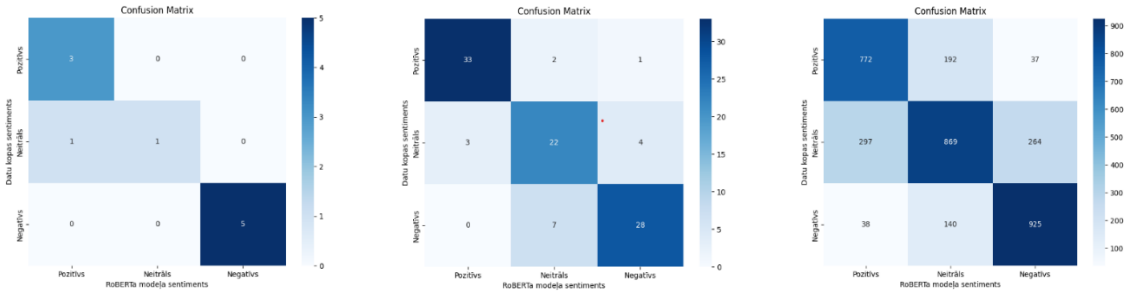
*RoBERTa* modelim tika veikti tie paši pieci mērījumi, kuri tika veikti *VADER* modelim - precizitāte, precīzums, jūtīgums, F1 rezultāts un vispārīgais novērtējumus.

Autors pēc programmas izstrādes ieguva šādus rezultātus, testējot dažādus komentāru apjomus no "*Twitter Sentiment Analysis Datasets: A Valuable Resource for Social Media Research*" datu kopas.

*Tabula 2.3.1.*

| Mērsistēmas | 10 komentāri | 100 komentāri | 3534 komentāri |
|---|---|---|---|
| Accuracy (%) | 90.00% | 83.00% | 72.61% |
| Precision (%) | 92.50% | 83.28% | 72.58% |
| Recall (%) | 90.00% | 83.00% | 72.61% |
| F1 Score (%) | 89.05% | 83.09% | 72.27% |
| Kopumā (%) | 90.39% | 83.09% | 72.52% |

Autors arī veica apjukuma matricas jeb ***confusion Matrix*** mērījumus:



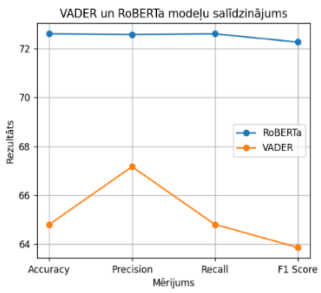

(19., 20., 21. attēli: Ir redzemas ***confusion Matrix*** tabulas, kur 19. attēlā ir 10 komentāru rezultāts, 20. attēlā 100 komentāru rezultāts un 21. attēlā 3534 komentāru rezultāts)

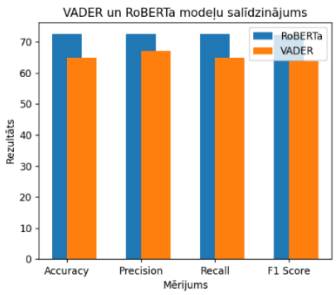
## 2.4 *VADER* un *RoBERTa* rezultātu salīdzināšana

Autors, analizējot abu modeļu veiktspēju, secināja par *RoBERTa* pārākumu salīdzinājumā ar *VADER*. Piemēram, *RoBERTa* modeļa precizitāte bija būtiski augstāka visos trīs mērījumu apjomos (10, 100 un 3534 komentāriem), salīdzinot ar *VADER* testēto modeli. Kopumā *RoBERTa* modeļa kopējais rezultāts pārsniedza *VADER* modeļa rezultātu. *RoBERTa* ieguva 84%, bet *VADER* testētais modelis ieguva 72,38%. Tas liecina, ka *RoBERTa* testētais modelis ir aptuveni par 10% precīzāks nekā *VADER* testētais modelis.

Šie secinājumi atspoguļo tikai abu modeļu veiktspēju konkrētā testa datu kopā. Lai gan *RoBERTa* modeļa pārākums ir ievērojams, ir svarīgi ņemt vērā arī citus faktorus, piemēram, datu kopas kvalitāti.

*RoBERTa* modeļa pārākums tā veiktspēja balstās uz tā spēju atpazīt un emulēt teikumu struktūru, saprotot to līdz noteiktai pakāpei. Šī spēja nav sasniedzama ar *VADER* modeļa leksikonu pamatotu pieeju. Svarīgi ir piebilst, ka, neskatoties uz *RoBERTa* modeļa prasmēm emulēt teikuma struktūru, tam trūkst patiesas saprašanas, kas līdzinātos cilvēku saprašanai. Piemēram, tas nespēj, uztver teikumus sarkasmu, kas bieži vien izvēršas viltus noteiktos sentimentos.



(22., 23. attēli: Ir redzemas divas tabulas, kur 22. attēlā ir attēlots poligona diagramma un 23. attēlā stabiņu diagramma. Abos attēlots ir attēlots *VADER* un *RoBERTa* testēto modeļu salīdzinājums balstoties uz veiktajiem mērijumiem.)

## 2.5 *Google Chrome* paplašinājuma izveide

Šodienas informācijas plūsma strauji pieaug, radot izaicinājumus gan saturu patērētājiem, gan veidotājiem. Cilvēki bieži pilnībā uzticas satura veidotājiem, kuri ne vienmēr ir objektīvi. Tas rada risku, ka veidotājs var manipulēt ar informāciju, ietekmējot patērētājus. Tāpēc ir svarīgi apsvērt arī citu cilvēku viedokļus par konkrēto saturu, it īpaši platformā "*YouTube*", kur satura veidotāji bieži pauž subjektīvus viedokļus, ko patērētāji var uzskatīt par faktiem. Tāpēc ir būtiski uzzināt, ko citi cilvēki domā par konkrēto viedokli, vai saturu, un veikt pašiem savus secinājumus.

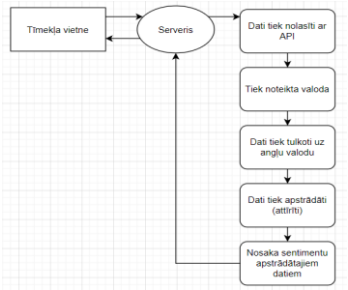
Veikt secinājumus, analizējot pāris komentārus, ir ātri un ērti, bet, kad to ir simtiem vai tūkstošiem, šāda analīze prasa ievērojami vairāk laika. Tāpēc izstrādāts *Google Chrome* paplašinājums, kas izskata un novērtē desmitiem, simtiem vai pat tūkstošiem komentāru sentimentus, tos iedalot - pozitīvos, neitrālos vai negatīvos. Paplašinājums arī parāda procentuālo sadalījumu par konkrētā video komentāru sadaļu attiecībā pret tēmu, kas pausta video saturā.

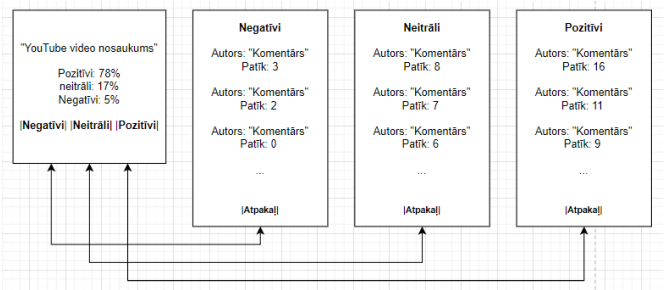
Šāda paplašinājuma izveide varētu atvieglot ne tikai pirkumus internetā, bet arī novērst viltus informācijas izplatīšanu, uzlabojot patērētāju drošību interneta vidē.

Sākumā ir svarīgi noskaidrot, kas vispār ir *Google Chrome* paplašinājumi. Tās ir programmas, ko varat instalēt pārlūkprogrammās, piemēram, *Chrome*, lai mainītu tās funkcionalitāti. Šie paplašinājumi var palīdzēt automatizēt noteiktas pārlūkprogrammas funkcijas vai pat mainīt esošās darbības un uzlabot programmatūras ērtības. [36.]

Paplašinājuma izveidi veidos divas daļas *front-end* un *back-end* jeb klienta lietojumprogramma un veidotāja lietojumprogramma, turpmāk *front-end* un *back-end*. *Front-end* ir visu lietotājam redzamo elementu daļa lietotnes vai tīmekļa vietnēs. Tas ir tas, ar ko lietotāji saskaras interaktīvi. Piemēram, tīmekļa interfeiss, dizains, pogas, formas, animācijas un citas vizuālas lietas. *Front-end* nodrošina lietotājam draudzīgu un pievilcīgu saskarni, lai viņi varētu efektīvi un ērti izmantot lietojumprogrammu. Toties *Back-end* darbojas aiz ekrāna un nodrošina funkcionalitāti, datu pārvaldību, datu bāzu savienojumu un citus darbības procesus, kas nav redzami lietotājiem. *Back-end* nodrošina datu apstrādi, uzglabāšanu un pārraidi, lai *front-end* varētu veikt vajadzīgas darbības. Saliekot kopā *front-end* un *back-end*, tas veido pilnīgu lietojumprogrammas struktūru.

Sākumā arutors izstrādāja *back-end* daļu. Tā tiks veidota šados soļos: datu rasmošana ar API, valodas noteikšana, datu tulkošana uz angļu valodu, datu apstrāde, sentimenta nolasīšana, servera izveide. Autors ir pievienojis *back-end* pirmkodu ar nosaukumu “back-end” (skatīt pielikumā – 3.), kā arī viss back-end kods ir rakstīts *Python (3.8)* OOP (objektu orientētā programmēšana) stilā. Autors iesaka sekot līdzi kodam, lai izprastu labāk aprakstīto.



(24., 25. attēls: *Google Chrome* pārlūka paplašinājuma izveides shēma un skice)

Pirmais solis ir komentāru iegūšana no kāda konkrēta *YouTube* video. Tam autors izvēlējās izmantot *YouTube data API v3*, jo tas ir paredzēts, lai rasmotu datus no *YouTube* platformas, tajā skaitā komentārus. Katram *YouTube* video ir savs unikāls identifikators, turpmāk ID. Piemēram, *url* -"https://www.*youtube*.com/watch?v=dQw4w9WgXcQ" unikālais ID ir "dQw4w9WgXcQ", padodot šo ID *YouTube data API v3*, tas spēj iegūt visus nepieciešamos datus no šī konkrētā video. Šo ID var uztvert kā atslēgu, kuru padodot API, tas spēj atvērt durvis uz visu informāciju noteiktajam video. Līdzīgā veidā konkrētā video ID no *front-end* daļas (kas tiks aprakstīts vēlāk) tiks aizsūtīts uz serveri un no servera uz *back-end* programmu, kas ievāc komentārus. Autors izvēlējas rasmot tikai oriģinālos komentārus, tas nozīmē, ka komentāri, kas atbild uz citiem komentāriem, netika rasmoti. Šādu izvēli autors veica, jo bieži cilvēki, atbildot uz citiem komentāriem, novirzās no konkrētā video satura un sāk komentēt paša komentāra autora personību. Šī pieeja var ietekmēt sentimenta novērtējumu attiecībā uz konkrēto video saturu. Taču nākotnē, iespējams, var izveidot un atsijāt šādus komentārus. API rasmošanas procesam autors izveidoja klasi "Scrape_Comments" un katram komentāram iedeva savu unikālo ID, lai pēc tam tos varētu sašķirot pareizi.

Valodas noteikšana: Kad komentāri ir iegūti ar API palīdzību, seko nākamais svarīgais solis – valodas noteikšana. Tā kā sentimenta noteikšanas modeļi spēj nolasīt sentimentu tikai angļu valodas tekstam, tad attiecīgi ir jāpārtulko komentāri uz angļu valodu. Taču ne vienmēr būs jāpārtulko visi komentāri, jo daudzos gadījumos droši vien lielākā daļa no komentāriem jau būs angliski. Tāpēc autors izveidotja klasi "Detect_Language", kura nosaka to, vai komentārs ir angļu valodā vai citā valodā, attiecīgi sadalot tos pa *json* failiem "only_english.json" un "comments_not_in_eng.json". Katram komentāram, kurš nav angļu valodā, tiek pimērots atribūts "language" jeb valoda, kas pasaka kurā valodā ir konkrētais komentārs.

Datu tulkošana angļu valodā: Autors šim procesam izveidoja klasi "Translate_All_Comments", kurā katrs komentārs, kas atrodas "comments_not_in_eng.json" jeb failā, kur nav angļu komentāri, tiek pārtulkots angļu valodā. Tad šie pārtulkotie komentāri tiek pārlikti failā "translated.json", kuram tiek pievienoti klāt orģināli angļu valodā esošie komentāri. Tulkošanas procesam tika izmantota "googletrans" bibliotēka.

Datu apstrāde: Šis process īsti neatšķiras no iepriekš pieminēto datu apstrādes, kad tika apskatīti *VADER* un *RoBERTa* modeļu testēšanā. Vienīgais, kas atšķiras - ja komentāram ir kāds "emoji", tad tas tiek noņemts. Datu apstrādei autors izveidoja klasi "Cleaing", kurā katrs komentārs no "translated.json" tiek apstrādāts ar dažādākajām metodēm. Pēc komentāru apstrādes tie tiek ielikti failā "cleaned.json".

Sentimenta nolasīšana: Pēc veiktās *VADER* un *RoBERTa* modeļu testēšanas izrādijās, ka *RoBERTa* modelis ir precīzāks komenāru sentimena nolasīsānai, taču autors vēlējās vēl precīzāk nolasīt sentimenu. Tāpēc autors apvienoja abus modeļus kopā un izveidoja hibrīdmodeli, kas ļaus iegūt, iespējams, vēl precīzāku sentimenta nolasīšanu. Šim procesam autors izstrādā klasi "SentimentAnalyzer", kurā visi komentāri no "cleaned.json" tiek tokenizēti, lai *RoBERTa* modelis var veikt sentimenta noteikšanu. *VADER* modelim tas netraucēs, jo tajā var arī palaist tokenizētu tesktu, kas nemainīs tā novērtējumu. Tad no abiem tiek iegūti rezultāti, pēc tam tie tiek sasummēti kopā katram komentāram un sadalīti ar divi, kas veido hibrīdmodeļa novērtējumu. Pēc tam, kad katram komentāram ir savs polaritātes rezultāts, tam tiek piemēroti sentimenta novērtējumi – pozitīvs, neitrāls vai negatīvs novētējums. Kā tika minēts iepriekš, katram komentāram ir piešķirts unikāls ID. Šis ID ir paredzēts, lai var viegli atrast oriģinālo komentāru pēc datu apstrādes. Šāds pieejas mērķis ir

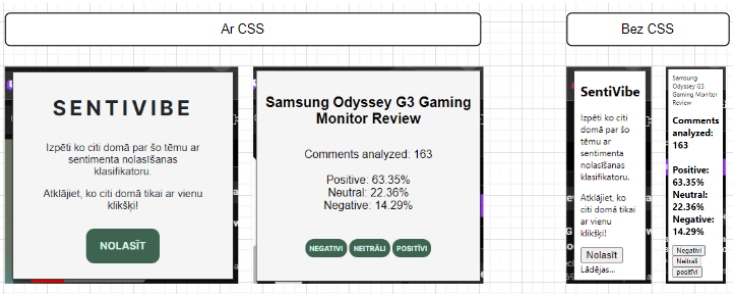
nodrošināt, ka, neskatoties uz iespējamo pārveidojumu pēc apstrādes, front-end lietotājiem tiek piedāvāts orģinālais komentārs, kas nav atšķirīgs no sākotnējā. Tas ļauj saglabāt sentimenta novērtējumu, tādējādi nodrošinot, ka pat komentāri, kuri sākotnēji bija citā valodā, tiek attēloti ar atbilstošu sentimentu. Šī pieeja ļauj saglabāt lietotājam saprotamu un kvalitatīvu informāciju, nemainot oriģinālo komentāru. Tad tie tiek iedalīti atseviškos json failos "positive.json", "neutral.json" un "negative.json".

Serveris: Autors veidoja mākoņa serveri uz VPS "Virtuālais Privātais Serveris", tas ir serveris, kas neatrodas lokāli, bet gan online vidē. Tas nodrošina saziņu starp interneta pārlūku un serveri, kas klausās uz localhost un to pārvērš savā API. Kad serverim nosūta GET pieprasījumu ar ceļu "/getData", tas saņem 'video_id' parametru no pieprasījumu no *front-end* paplašinājuma un saglabā to JSON failā. Pēc tam tas palaiž sentimenta nolasīšanas prgrammu. Rezultāti no sentimena nolasīšanas programmas tiek apkopoti un atgriezti kā JSON atbilde, iekļaujot negatīvu, neitrālu un pozitīvu sentimenta klasifikācijas. Tad tie tiek atgriesti *front-end* paplašinājumam.

Nākamais solis ir *front-end* daļas izstrāde, tā tiks veidota šados soļos: manifesta izveide, interface izstrāde, paplašinājuma mazais *back-end* un paplašinājuma augšuplāde *Chrome* pārlūkā. Autors pievienoja *front-end* pirmkodu ar nosaukumu "front-end" (skatīt pielikumā – 4.). Autors iesaka sekot līdzi kodam, lai labāk izprastu aprakstīto.

Manifesta izveide: Katra paplašinājuma izveidē pirmais solis ir uzrakstīt manifestu, ko raksta json failā. Manifesta fails ir kā instrukcijas pārlūkam, lai pateiktu, kas paplašinājumā atradīsies. Parasti tajos atrodas tādi dati, kā paplašinājuma versijas numurs, nosaukums, atļaujas, kas nepieciešamas, lai paplašinājums varētu darboties, un citi rekvizīti. Bez šāda manifesta faila pārlūks nesapratīs, ko darīt, tāpēc šis ir būtisks process paplašinājuma izveidē. Autors izveidoja šadu manifesta failu, kurā ievieotja nepiecšamos datus paplašinājuma izveidei. [37.]

Interface izstrāde: Autors izstrādāja inteface *HTML* un *CSS* valodās. Ar *HTML* valodu tika strukturēts, kas atradīsies jeb ko redzēs lietotājs, izmantojot paplašinājumu. *CSS* ir būtiska daļa, lai padarītu paplašinājumu pievilcīgāku lietotājam. Autors to izmantoja, lai nofromētu struktūru un elementus saliktu attiecīgajās vietās, kā arī, lai pievienotu elementiem un tesktiem noteiktu stilu. To autors veica failos "popup.html" un "style.css".




(25., 26. attēlā parāda paplašinājumu ar CSS pielieotjumu, taču 26. un 27. attēlā parāda paplašinājumu bez CSS pielieotjuma)

Mazais *back-end*: Paplašinājumiem var būt arī savs *back-end*, kas saszinās ar kādu serveri vai veic darbības, ko lietotājs neredz. Autora gadījumā back-end veic *YouTube* video ID iegūšanu, uzlikšķinot uz pogas "Nolasīt". Tad šo ID aizsūta atpakaļ uz lokālo server.i Autors veicis šo processu "background.js" failā. Tad, kā jau tika ierpiekš minēts, serveris saņem šo ID, palaiž sentimena nolasīsānas programmu un tā rezultātus atsūta atpakaļ uz paplašinājumu, kur

tos attēlo lietotājam paplašinājumā dilstošā secībā, kur augstākais komentārs ir ar vissvairāk "patīk" pogu skaitu.

Paplašinājuma ielāde pārlūkā: Autors izveidojot paplašinājuma *front-end* daļu un to lejuplādēja *Chrome* pārlūka. To autors izdarija ieviedojot visus *front-end* nepieciešamos failus vienā mapītē, tad mapīti augšuplādēja vietnē "*chrome*://extensions/" un paplašinājums bija gatavs lietošanai.

Paplašinājuma demonstrācija: No sākuma lietotājis ieiet kādā *YouTube* video, tad uzspiež uz autora veidotā paplašinājuma. Tad lietotājs nospiež "Nolasīt" pogu, tad "Lādējas…", tiek noteikts sentiments, tad tiek attēloti rezultāti un visbeidzot lietotājs var izvēlēties sadaļas un filtrēt cauri komentāriem (skatīt pielikumā - 6.).

# Secinājumi

Pētījuma mērķi un visus tajā izvirzītos uzdevumus autoram veiksmīgi izdevās sasniegt. Autors izpētīja, kas ir NLP pamati, kā un kur tie tiek pielietoti. Autors secināja, ka sentimenta nolasīšana ir svarīga apakšnozare, kura palīdz novērtēt teksta emocionālo saturu un piedāvā plašas pielietošanas iespējas dažādās nozarēs. Autoram izdevās izprast, kā tiek nolasīti dati, kā tie tiek apstrādāti un kā no tiem var nolasīt sentimentu ar divu veidu sentimenta nolasīšanas modeļiem – *VADER* un *RoBERTa*. Darba autors veiksmīgi analizēja un salīdzināja *VADER* un *RoBERTa* modeļus, kā arī veiksmīgi izveidoja *Google Chrome paplašinājumu*, kurš spēj klasificēti komentārus pēc to sentimenta.

Praktiskajā daļa autors, salīdzinot *VADER* un *RoBERTa* modeļus, secināja, ka *RoBERTa* modelis ir precīzāks, jo tas balstās uz tā spēju atpazīt un emulēt teikumu struktūru. Taču, kas tāds nav sasniedzams *VADER* modelim, jo tā pamatā ir leksikonu pieeja. Tā rezultātā *RoBERTa* iegūst kopējo novērtējumu 84%, bet *VADER* 72,38%.

Autors radīja pārlūka paplašinājumu, kas darbojās efektīvi un ir noderīgs rīks gan ikvienam *YouTube* platofrmas patērētājam, gan arī satura veidotājiem. Tomēr ir svarīgi atzīt dažus trūkumus. Pirmkārt, sentimentu sadalījums vai izpratne ne vienmēr būs pilnīgi precīza, jo *RoBERTa* modeļa precizitāte ir 84%, *VADER* precizitāte ir 72,38%, un hibrīdais modelis nespētu pietuvoties 100% precizitātei, jo komentāri, kuri ir rakstīti ar sarkasmu, var ietkmēt viltus pareiza sentimenta nolasījumu, tādēļ var parādīties nepareizi izvēlēti komentāri. Turklāt, nav iespējams vienmēr atpazīt komentārus, kuri izteikti sarkastiski, jo sentimentu nosaka algoritmi, kas var nesaprast šāda veida izteiksmes. Turklāt, tulkojot komentārus no citām valodām, angļu valodas tulkojums ne vienmēr atspoguļo pilnīgi to pašu nozīmi, kā oriģinālvalodā, kas var ietekmēt sentimenta novērtējumu. Lai novērstu šos trūkumus, varētu ieviest sarkasma nolasītāju, kas varētu novērst viltus pozitīvas atbildes, tā rezultātā uzlabojot sentimenta precizitāti. Varētu ieviest programmā mašīnmācīšanos, no katras komentāru analizētās sadaļas, tas padotu modelim trenēšanās datus no iepriekšējiem nolasījumiem, tādejādi tas paliktu ar laiku arvien precīzāks, jo mācītos pats no sevis.

Šī darba autoram izdevās veiksmīgi apstiprināt autora izvirzīto hipotēzi, kā arī izpildīt visus darba uzdevumus un sasniegt visu minēto darba mērķī.

# Izmantotā literatūra un citi avoti


1. Peter EH Smee and Linda Smee. Neuro-Linguistic Programming, The Key To Accelerated Learning. 9. lappa - NLP explained. (2002) (Skatīts 24.10.2023)
2. IBM - What is natural language processing (NLP)? Pieejams: https://www.ibm.com/topics/natural-language-processing (Skatīts 24.10.2023)
3. Foreseemed - NLP negation. Pieejams: https://www.foreseemed.com/natural-language-processing-in-healthcare#:~:text=NLP%20negation%20in%20healthcare%20is,have%20a%20condition%20or%20symptom. (Skatīts 24.11.2023)
4. Techtarget - How does sentiment analysis work? Pieejams: https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining (Skatīts 07.11.2023)
5. ScienceDirect - A review on sentiment analysis from social media platforms. Pieejams: https://www.sciencedirect.com/science/article/pii/S0957417423003639 (Skatīts 07.11.2023)
6. Voiceform - How to Do Sentiment Analysis in 4 Steps (With Examples). Pieejams: https://www.voiceform.com/blog-posts/sentiment-analysis (Skatīts 07.11.2023)
7. Repustate - What Are The Sources Of Gathering Sentiment Analysis Data? Pieejams: https://www.repustate.com/blog/sentiment-analysis-data-source/ (Skatīts 07.11.2023)
8. Medium - Ways to collect data for sentiment analysis. Pieejams: https://medium.com/analytics-vidhya/data-collection-and-annotation-measures-for-sentiment-analysis-767da1dd4272 (Skatīts 07.11.2023)
9. Dataconomy - Role of data preprocessing in sentiment analysis. Pieejams: https://dataconomy.com/2023/07/28/data-preprocessing-steps-requirements/ (Skatīts 11.11.2023)
10. Monkeylearn - Your Guide to Data Cleaning & The Benefits of Clean Data. Pieejams: https://monkeylearn.com/data-cleaning/ (Skatīts 11.11.2023)
11. Monkeylearn - 8 Effective Data Cleaning Techniques for Better Data. Pieejams: https://monkeylearn.com/blog/data-cleaning-techniques/ (Skatīts 11.11.2023)
12. Arize - Tokenization: Unleashing The Power of Words. Pieejams: https://arize.com/blog-course/tokenization/#:~:text=It%20involves%20breaking%20down%20a,entity%20recognition%2C%20and%20text%20classification. (Skatīts 11.11.2023)
13. Nlpcloud - Kas ir žetonizācija? Pieejams: https://nlpcloud.com/lv/nlp-tokenization-api.html (Skatīts 11.11.2023)
14. nlp.stanford.edu - Stemming and lemmatization. Pieejams: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (Skatīts 11.11.2023)
15. Analyticsindiamag - What is negation handling? Pieejams: https://analyticsindiamag.com/when-to-use-negation-handling-in-sentiment-analysis/#:~:text=Negation%20handling%20is%20a%20method,vicinity%20or%20scope%20of%20negation. (Skatīts 11.11.2023)
16. Monkeylearn - Sentiment Analysis & Machine Learning. Pieejams: https://monkeylearn.com/blog/sentiment-analysis-machine-learning/ (Skatīts 11.11.2023)
17. Monkeylearn - How Does Sentiment Analysis Work? Pieejams: https://monkeylearn.com/sentiment-analysis/ (Skatīts 11.11.2023)
18. Analyticsvidhya - Rule-Based Sentiment Analysis in Python. Pieejams: https://www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/ (Skatīts 11.11.2023)

19. Monkeylearn - Rule-based Approaches. Pieejams: https://monkeylearn.com/sentiment-analysis/#:~:text=Automatic%20Approaches,positive%2C%20negative%2C%20or%20neutral (Skatīts 11.11.2023)
20. Octoparse - Scrape *YouTube* Comments for Sentiment Analysis. Pieejams: https://www.octoparse.com/blog/*youtube*-comment-scraper (Skatīts 11.11.2023)
21. Mulesoft - What is an API?. Pieejams: https://www.mulesoft.com/resources/api/what-is-an-api (Skatīts 17.11.2023)
22. Developer.google.com - The following table defines the properties that appear in this resource: Pieejams: https://developers.google.com/*youtube*/v3/docs/comments#:~:text=The%20snippet%20object%20contains%20basic%20details%20about%20the%20comment.&text=The%20display%20name%20of%20the%20user%20who%20posted%20the%20comment.&text=The%20URL%20for%20the%20avatar%20of%20the%20user%20who%20posted%20the%20comment.&text=The%20URL%20of%20the%20comment%20author's%20*YouTube*%20channel%2C%20if%20available. (Skatīts 17.11.2023)
23. Zyte- What Is Web Scraping? Pieejams: https://www.zyte.com/learn/what-is-web-scraping/ (Skatīts 17.11.2023)
24. Quora - Is scraping *YouTube* legal? Pieejams: https://www.quora.com/Is-scraping-*YouTube*-legal (Skatīts 17.11.2023)
25. GitHub - *VADER*-Sentiment-Analysis. Pieejams: https://github.com/cjhutto/*vader*Sentiment (Skatīts 25.11.2023)
26. Comet - *RoBERTa*: A Modified BERT Model for NLP. Pieejams: https://www.comet.com/site/blog/*roberta*-a-modified-bert-model-for-nlp/#:~:text=An%20open%2Dsource%20machine%20learning,Facebook%20in%20the%20year%202019. (Skatīts 25.11.2023)
27. Github - *vader*-vs-*roberta*. Pieejams: https://github.com/topics/*vader*-vs-*roberta* (Skatīts 25.11.2023)
28. ReserchGate - [attēls] Pieejams: https://www.researchgate.net/figure/Overview-of-web-scraping-system_fig2_347999311 (Skatīts 04.12.2023)
29. Linkedin - [attēls] Pieejams: https://www.linkedin.com/pulse/stemming-lemmatization-ashik-kumar (Skatīts 11.12.2023)
30. CodeCademy - What is REST? Pieejams: https://www.codecademy.com/article/what-is-rest (Skatīts 11.12.2023)
31. Analyticsvidhya - Building Naive Bayes Classifier from Scratch to Perform Sentiment Analysis. [attēls] Pieejams: https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/ (Skatīts 12.12.2023)
32. Medium- Accuracy, Precision, Recall, F-1 Score, Confusion Matrix, and AUC-ROC. Pieejams: https://medium.com/@riteshgupta.ai/accuracy-precision-recall-f-1-score-confusion-matrix-and-auc-roc-1471e9269b7d (skatīts: 05.01.2024)
33. Gigasheet- Twitter Sentiment Analysis Datasets: A Valuable Resource for Social Media Research. Pieejams: https://www.gigasheet.com/sample-data/sentiment-analysis-dataset (skatīts: 06.01.2024)
34. GitHub- *VADER*-testēšana . Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/mode%C4%BCi (Skatīts: 10.01.2024)
35. GitHub-*RoBERTa*-testēšana, Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/mode%C4%BCi (Skatīts: 10.01.2024)
36. Neilpatel - How to Build a *Chrome* Extension. Pieejams: https://neilpatel.com/blog/*chrome*-extension/#:~:text=Google%20*Chrome*%20extensions%20are%20programs,and%20improve%20your%20software's%20convenience. (Skatīts: 10.01.2024)

37. Micrasoft - Manifest file format for extensions. Pieejams: https://learn.microsoft.com/en-us/microsoft-edge/extensions-chromium/getting-started/manifest-format?tabs=v3 (Skatīts: 10.01.2024)

# Pielikumi

1. GitHub- *VADER*-testēšana . Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/mode%C4%BCi
2. GitHub-*RoBERTa*-testēšana, Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/mode%C4%BCi
3. GitHub – back-end. Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/back-end
4. GitHub – front-end. Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/front-end
5. GitHub – Datu kopa. Pieejams: https://github.com/kristersla/school-work/tree/main/ZPD/praktiskais/mode%C4%BCi
6. 1. attēls: paplašinājuma demonstrācija: