# Analiza podataka sa socijalnih mreža podržana strojnim učenjem

**Bikić, Arijana**




SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

**Arijana Bikić**

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

Mentori:

izv. prof dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Arijana Bikić

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se profesoru Tomislavu Stipančiću za pruženu podršku te pomoć prilikom izrade ovog rada.

Posebice se zahvaljujem svojoj obitelji, prijateljima i mom Blagi na velikoj podršci i razumijevanju tijekom studiranja.

Arijana Bikić

SVEUČILIŠTE U ZAGREBU
**FAKULTET STROJARSTVA I BRODOGRADNJE**
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

| Sveučilište u Zagrebu<br>Fakultet strojarstva i brodogradnje | |
|---|---|
| Datum | Prilog |
| Klasa: **602 – 04 / 24 – 06 / 1** | |
| Ur.broj: **15 – 24 –** | |

# ZAVRŠNI ZADATAK

Student: **Arijana Bikić** JMBAG: **0035231834**

Naslov rada na hrvatskom jeziku: **Analiza podataka sa socijalnih mreža podržana strojnim učenjem**

Naslov rada na engleskom jeziku: **Analysis of data from social networks supported by machine learning**

Opis zadatka:

Računalna analiza teksta te prepoznavanje osjećaja se kao interdisciplinarna i srodna područja nalaze na presjecištu računalnih znanosti i lingvistike. Analiza teksta prvenstveno se koristi da bi se računalima omogućilo da interpretiraju ljudski jezik. Cilj je da računalo može "razumjeti" sadržaj dokumenata, uključujući kontekstualne nijanse, gdje posebno do izražaja dolazi analiza sentimenta. Tehnologija omogućuje računalni i automatizirani uvid u digitalizirane podatke, što se kao temelj može koristiti u različite svrhe. Dostupna tehnologija za analizu i obradu teksta te tehnike strojnog učenja se učinkovito koriste prilikom analize raznih trendova te ispitivanja javnog mijenja. Dobivene spoznaje se potom mogu analizirati prilikom izrade te kod usklađivanja strategije tvrtke u ovisnosti o njenim interesima i aktivnostima (npr. praćenje društvenih medija da bi se prilagodio izlazni proizvod, istraživanje tržišta da bi se odredile proizvodne količine i cijena, upravljanje reputacijom brenda i slično).

U radu je potrebno:

- upotrijebiti NLTK alat (engl. Natural Language Toolkit) kod analize i obrade podataka sa socijalne mreže Twitter (sada X)
- klasificirati dobivene podatke koristeći Naive Bayes klasifikator kako bi se kreirao upotrebljiv model za analizu velikih tokova podataka
- evaluirati rezultate te dati kritički osvrt na upotrebljivost navedene metodologije te na rad Naive Bayes modela.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2023.

Zadatak zadao:
Izv. prof. dr. sc. Tomislav Stipančić

Datum predaje rada:
**1. rok:** 22. i 23. 2. 2024.
**2. rok (izvanredni):** 11. 7. 2024.
**3. rok:** 19. i 20. 9. 2024.

Predviđeni datumi obrane:
**1. rok:** 26. 2. – 1. 3. 2024.
**2. rok (izvanredni):** 15. 7. 2024.
**3. rok:** 23. 9. – 27. 9. 2024.

Predsjednik Povjerenstva:
Prof. dr. sc. Damir Godec

# SADRŽAJ

# POPIS SLIKA

# POPIS OZNAKA

| Oznaka | Jedinica | Opis |
|---|---|---|
| $S$ | / | Skup mogućih opažanja |
| $P$ | / | Skup distribucija vjerojatnosti na $S$ |
| $\theta$ | / | Skup parametara modela |
| $x$ | / | Ulazni signal |
| $w$ | | Težinski faktor |
| $y$ | | Izlazni signal |
| $f$ | | Aktivacijska funkcija |

# SAŽETAK

Sažetak ovog završnog rada istražuje primjenu analize sentimenta kroz strojno učenje na podacima s Twitter-a. Analiza sentimenta je proces analiziranja i prepoznavanja emocija ili stavova iz teksta, a strojno učenje je grana umjetne inteligencije koja omogućuje računalima da nauče iz podataka i izvode zadatke bez eksplicitnog programiranja. Ovaj rad istražuje kako se modeli strojnog učenja, posebno Naivni Bayesov klasifikator, mogu koristiti za analizu sentimenta na Twitter-ovim podacima. Kroz eksperimente i evaluaciju, analizira se učinkovitost ovih modela u prepoznavanju pozitivnih i negativnih sentimenta u tweetovima te pruža uvid u izazove i mogućnosti ovog pristupa.

Ključne riječi: NLP, Python, Sentiment Analysis

# SUMMARY

The summary of this thesis explores the application of sentiment analysis through machine learning on Twitter data. Sentiment analysis is the process of analyzing and recognizing emotions or attitudes from text, and machine learning is a branch of artificial intelligence that allows computers to learn from data and perform tasks without explicit programming. This paper explores how machine learning models, specifically the Naive Bayes classifier, can be used for sentiment analysis on Twitter data. Through experiments and evaluation, the effectiveness of these models in recognizing positive and negative sentiments in tweets is analyzed and provides insight into the challenges and opportunities of this approach.

Key words: NLP, Python, Sentiment Analysis

# 1. UVOD

Računalna analiza teksta te prepoznavanje osjećaja (Natural Language Processing – NLP) predstavlja granu računalne znanosti koja se bavi interakcijom između računala i ljudskog jezika, omogućuju računalima da razumiju, tumače i generiraju ljudski jezik. Kroz napredne algoritme i modele, NLP omogućava analizu velikih skupova teksta, uključujući komentare, poruke i postove na društvenim mrežama. Ova tehnologija pruža mogućnost otkrivanja uzoraka, sentimenta te identifikacije ključnih tema koje dominiraju među korisnicima.

U kombinaciji s NLP-om, strojno učenje omogućava automatiziranu obradu podataka te stvaranje modela koji mogu predviđati ponašanje korisnika ili identificirati relevantne informacije. Ova integracija tehnologija omogućuje nam bolje razumijevanje dinamike društvenih mreža, detektiranje trendova, kao i praćenje stajališta i reakcija korisnika na određene događaje.

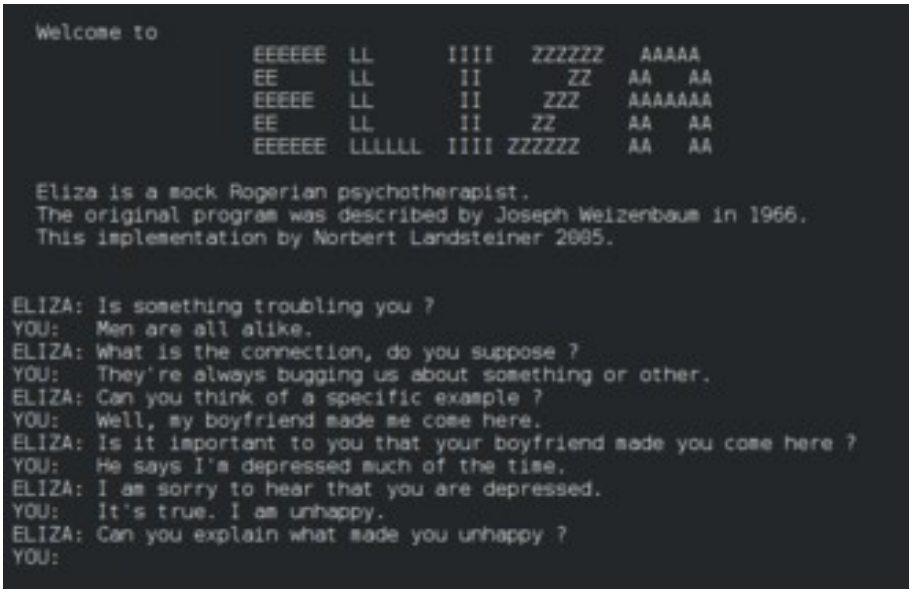
U nastavku rada istražit ćemo konkretne primjene NLP-a i strojnog učenja u analizi podataka s društvenih mreža, razmatrajući koristi i izazove ove metodologije. Osim toga, istražit ćemo moguće pristupe obradi podataka, kao i implementaciju modela za optimalno korištenje informacija dobivenih s društvenih mreža. Ova istraživanja ne samo da pružaju uvid u suvremene metode analize podataka, već otvaraju prostor za daljnji razvoj i primjenu ovih tehnologija u različitim područjima istraživanja i poslovanja.

# 2. RAČUNALNA ANALIZA I PREPOZNAVANJE OSJEĆAJA

## 2.1. Razvoj računalne analize i prepoznavanje govora

### *2.1.1. Simbolički NLP*

Simbolički NLP je evoluirao od 1950-ih do kasnih 1990-ih, pri čemu je osnovna ideja ilustrirana eksperimentom "Kineska soba" Johna Searlea. SHRDLU, sustav iz 1960-ih, i ELIZA, simulacija psihoterapeuta, bili su najuspješniji u ograničenim "blokovskim svjetovima". U 1980-ima, simboličke metode dominirale su NLP-om s istraživanjima usmjerenim na parsiranje, morfologiju, semantiku te razvoj chatterbota poput Ractera i Jabberwackya. Ključan trenutak bio je rastući značaj kvantitativne evaluacije, što je konačno dovelo do statističkog preokreta 1990-ih. [1]




***Slika 1:*** *Prikaz korištenje sustava ELIZA [2]*

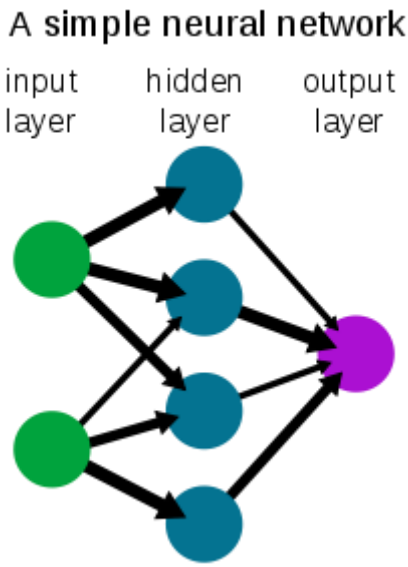
### *2.1.2. Statistički NLP*

Do 1980-ih, sustavi za obradu prirodnog jezika temeljili su se na složenim skupovima ručno napisanih pravila. No, krajem 1980-ih došlo je do revolucije s uvođenjem strojnog učenja za obradu jezika, potaknuto povećanom računalnom snagom. U 1990-ima, statističke metode u strojnom prijevodu postigle su značajan uspjeh, posebice rad u IBM-ovom istraživačkom odjelu, koristeći IBM-ove modele predviđanja. U 2000-ima, s rastom interneta, fokus

istraživanja prebacio se na nesupervizirano i polusupervizirano učenje, gdje algoritmi uče iz obilja neoznačenih jezičnih podataka dostupnih na internetu. Iako često manje precizna od superviziranog učenja, ova tehnika iskoristila je bogatstvo neoznačenih podataka dostupnih na internetu. [1]

### *2.1.3. Neuronski NLP*

Neuronska obrada koristi umjetne neuronske mreže kako bi modelirala i interpretirala jezične uzorke, omogućujući precizniju analizu i obradu prirodnog jezika u računalnom okruženju. Tomáš Mikolov primijenio je jednostavnu rekurentnu neuronsku mrežu s jednim skrivenim slojem na modeliranje jezika. U 2010-ima, učenje reprezentacija i metode strojnog učenja s dubokim neuronskim mrežama postale su uobičajene u obradi prirodnog jezika. Popularnost ovih tehnika proizlazi iz rezultata koji pokazuju da takve tehnike mogu postići vrhunske rezultate u mnogim zadacima obrade NLP-a, uključujući modeliranje jezika i parsiranje. Ovo je posebno važno u medicini i zdravstvu, gdje obrada prirodnog jezika pomaže analizirati bilješke i tekst u elektroničkim zdravstvenim zapisima.



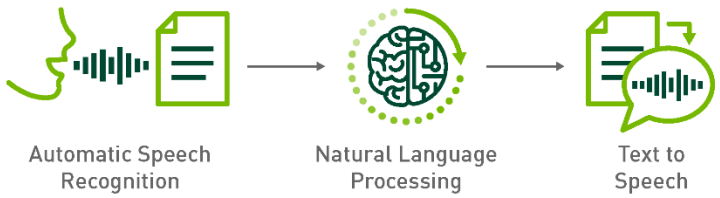

***Slika 2:*** *Shematski prikaz neuronske mreže [3]*

## 2.2. Metode obrade prirodnog jezika

### *2.2.1. Statističke metode i modeli*

Statistički model je matematički model koji obuhvaća skup statističkih pretpostavki o generiranju uzoraka podataka (i sličnih podataka iz veće populacije). On predstavlja, često u značajno idealiziranom obliku, proces generiranja podataka. Statistički model obično se definira kao matematički odnos između jedne ili više slučajnih varijabli i drugih neslučajnih varijabli. Kao takav, statistički model je „formalna reprezentacija teorije“. U matematici, statistički model se promatra kao par $(\boldsymbol{S}, \boldsymbol{P})$, gdje $\boldsymbol{S}$ je skup mogućih opažanja, a $\boldsymbol{P}$ je skup distribucija vjerojatnosti na $\boldsymbol{S}$. $\boldsymbol{P} = \{\boldsymbol{\theta} \in \boldsymbol{\theta}\}$, gdje skup $\boldsymbol{\theta}$ označuje parametre modela. [4]

Statistički modeli, tj. statistička metoda u procesiranju prirodnog jezika, često se primjenjuje u sustavima za prepoznavanje govora ili glasa. Ti sustavi omogućuju računalu identifikaciju i prijevod jezika izgovorenog od strane čovjeka u tekstualni zapis. Demonstracija primjene sustava za prepoznavanje govora prikazana je na slici [Slika 3.]. U nekim slučajevima za prepoznavanje glasa korisnici moraju proći kroz obuku, čitajući tekst ili određene riječi iz vokabulara kako bi ih sustav pohranio. [5] Iako su neuronske mreže postale vodeće u području neuronskog procesiranja prirodnog jezika, statističke metode i dalje zadržavaju važnost u situacijama gdje je potrebna statistička interpretacija i transparentnost.




***Slika 3:*** *Shematski prikaz uporabe sustava za prepoznavanje govora [6]*

Statistički NLP se primjenjuje još u:

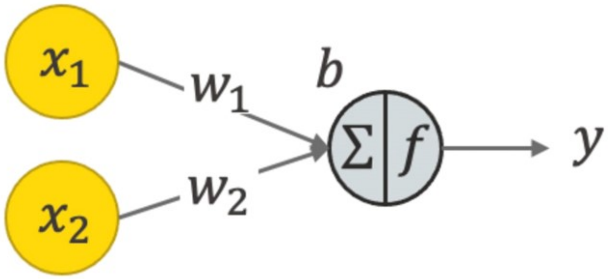
- Prijedlozima automatskog dovršavanja.
- Prepoznavanju rukopisa s leksičkim usvajanjem, čak i u teško čitljivim tekstovima.
- Otkrivanju i ispravljanju pravopisnih grešaka.
- Prepoznavanju govora.

- Sažimanju tekstova.
- Opisivanju slika. [7]

### *2.2.2. Neuronske mreže*

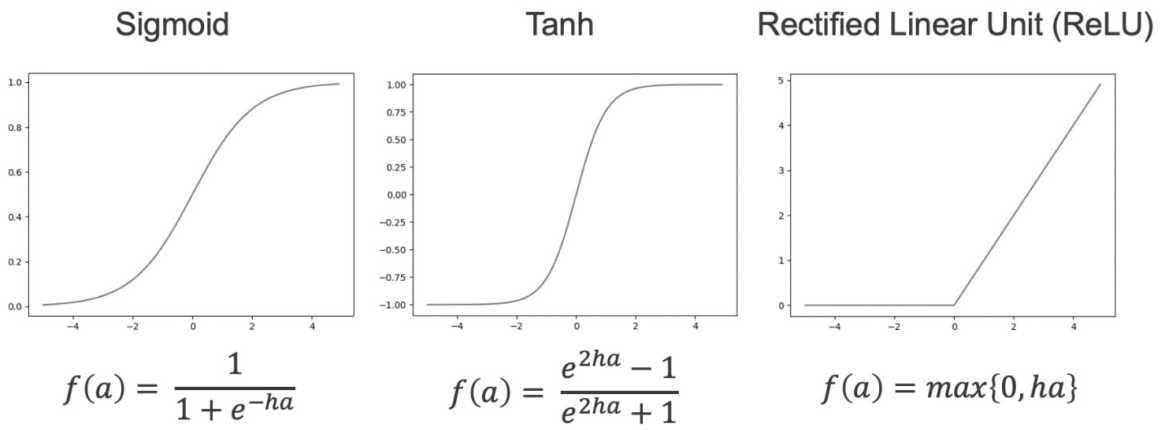
Umjetna neuronska mreža (ANN) predstavlja skup umjetnih neurona, često apstraktnih entiteta, koji su međusobno povezani i komuniciraju kroz operacije obrade signala te s organizacijom inspiriranom ljudskim mozgom. Ova mreža može imati jedan ili više ulaznih i uvijek jedan izlazni čvor s jednim ili više skrivenih slojeva, poznatih kao višeslojne mreže. Neuroni unutar mreže povezani su vezama kroz koje se prenose signali, a aktiviraju se aktivacijom funkcija kad ispunjavaju određeni uvjet. [8]

Jedan umjetni neuron donekle je inspiriran biološkim neuronima. Ima određene signale na ulazu ($\boldsymbol{x_1, x_2}$), koji se pomnože s težinskim faktorima ($\boldsymbol{w_1, w_2}$) te njihov umnožak rezultiraju u informaciju o jakosti spoja dva neurona. Umnoženi signali se zbroje, nakon čega se na dobivenu sumu primjenjuje aktivacijska funkcija $\boldsymbol{f}$, rezultirana izlaznim signalom $\boldsymbol{y}$. Prijenos signala prikazan je na slici [Slika 4]. [9]



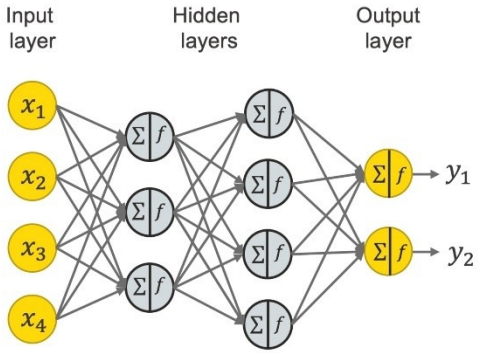

***Slika 4:*** *Građa umjetnog neurona [9]*

Odabir aktivacijske funkcije predstavlja jedno od pitanja dizajna prilikom definiranja neuronske mreže. Na slici [Slika 5] prikazane su neke uobičajene aktivacijske funkcije. Na primjer, izlaz sigmoidne funkcije uvijek se nalazi između 0 i 1, pružajući interpretaciju izlaza kao vjerojatnosti pripadnosti određenoj klasi. Stoga se često primjenjuje u izlaznom sloju mreže za probleme binarne klasifikacije. [9]




***Slika 5:*** *Prikaz aktivacijskih funkcija [9]*

U neuronskoj mreži, mnogi umjetni neuroni se kombiniraju u mrežu. Primjer kako ih povezati prikazan je na slici [Slika 6]. Ovo se naziva potpuno povezanom neuronskom mrežom s unaprijednim prijenosom podataka ili višeslojnim perceptronima. [9]




***Slika 6:*** *Prikaz primjera neuronske mreže s unaprijednim prijenosom podataka. [9]*

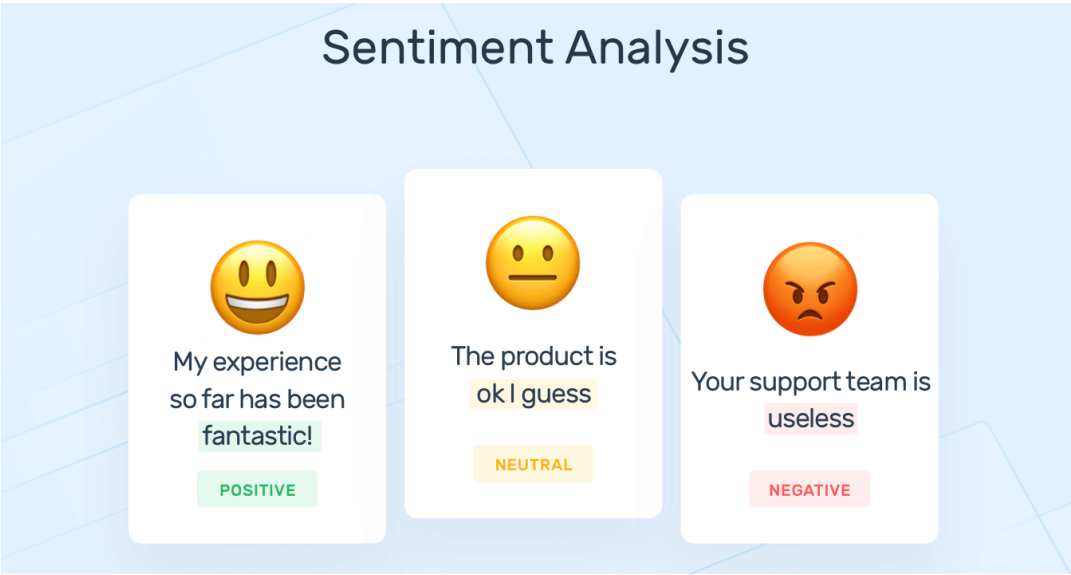
Mreža prikazana na slici [Slika 6] počinje s ulaznim slojem s lijeve strane. Taj sloj definira broj ulaza u mrežu i ne obavlja nikakve izračune. Zatim slijede dva skrivena sloja. Prvi skriveni sloj koristi ulazne vrijednosti za izračun izlaza svoja tri neurona, provodeći navedene korake za svaki neuron. Drugi skriveni sloj koristi izlaze prvog skrivenog sloja za izračun izlaza svoja četiri neurona. Konačno, izlazni sloj koristi ove vrijednosti kao ulaz za izračunavanje svojih izlaza. [9]

### *2.2.3. Analiza sentimenta*

Analiza sentimenta (engl. Sentiment Analysis) predstavlja korištenje obrade prirodnog jezika, analize teksta, računalne lingvistike i biometrije kako bi se sustavno identificirala i proučavala afektivna stanja i subjektivne informacije. Analiza sentimenta široko se primjenjuje na materijale od strane potrošača, poput recenzija i odgovora na ankete, na internetu i društvenim medijima, te materijale u zdravstvu za primjene koje sežu od marketinga do korisničke podrške do kliničke medicine. [10]

Primjene analize sentimenta su beskrajne i mogu se primijeniti u bilo kojoj industriji, od financija i maloprodaje do ugostiteljstva i tehnologije. Niže su navedeni neki od najčešće korištenih načina primjene analize sentimenta u poslovanju:

- Praćenje društvenih medija.
- Praćenje branda.
- Glas korisnika (VoC).
- Korisnička podrška.
- Marketinško istraživanje. [11]




***Slika 7:*** *Primjer primjene analize sentimenta putem recenzija. [11]*

## 2.3. Primjene NLP-a u programskim aplikacijama

Obrada prirodnog jezika (NLP) ima raznolike primjene u programskim aplikacijama. Nekoliko primjera su

1. *Virtualni asistenti*: Aplikacije kao što su Siri, Google Assistant i Amazon Alexa koriste NLP za razumijevanje i odgovaranje na korisničke upite.
2. *Automatsko označavanje entiteta*: U programima za analizu teksta, NLP se koristi za automatsko prepoznavanje i označavanje entiteta poput imena, mjesta i ključnih pojmova.
3. *Strojno prevođenje*: Aplikacije poput Google Translate koriste NLP za prijevod teksta s jednog jezika na drugi.
4. *Generiranje prirodnog jezika*: NLP se koristi za generiranje prirodnog jezika, što je vidljivo u chatbotima, automatiziranim odgovorima na e-poštu i sličnim aplikacijama.
5. *Autokorekcija i predikcija teksta*: Mobilne tipkovnice i aplikacije za pisanje koriste NLP za autokorekciju i predikciju riječi kako bi poboljšale točnost i brzinu unosa teksta.
6. *Pretraga i filtriranje e-pošte*: Algoritmi pretraživanja i filtriranja u aplikacijama za e-poštu koriste NLP kako bi poboljšali relevantnost i učinkovitost pretrage.
7. *Sumarizacija teksta*: Aplikacije za sumarizaciju teksta koriste NLP za izdvajanje ključnih informacija i stvaranje sažetaka izduženih tekstova.
8. *Prilagodba korisničkog iskustva*: NLP se koristi za personalizaciju korisničkog iskustva u aplikacijama, prilagođavajući se korisničkim preferencijama i ponašanju.
9. *Otkrivanje prijevare*: U financijskim aplikacijama, NLP se koristi za otkrivanje potencijalnih prijevara analizom teksta u transakcijama i komunikaciji s klijentima.

Ove primjene ilustriraju široki spektar mogućnosti koje NLP donosi u svijet programskih aplikacija, poboljšavajući korisničko iskustvo, automatizirajući procese i pružajući korisne analize. [12]

# 3. IZVEDBA ZADATKA U PYTHON-U

U idućim poglavljima rada detaljno je opisan praktični segment, tj. implementacija postavljenog zadatka koristeći programski jezik Python. Prvi dio obuhvaća opis funkcionalnosti programske aplikacije. U drugom dijelu pružena je teorijska osnova programskog jezika Python, zajedno s pregledom nekih od korištenih biblioteka tijekom izrade ovog projekta. Na kraju, detaljno je razrađena funkcionalnost programa, počevši od načina prikupljanja podataka sa socijalnih mreža pa sve do analize svih primijenjenih tehnika obrade prirodnog jezika (NLP).

## 3.1. Programski jezik Python

Python je popularan programski jezik koji je stvorio Guido van Rossum, a pušten je u uporabu 1991. godine. Python je jezik općenite namjene, nije specijaliziran za određene probleme te se koristi za stvaranje različitih programa. Karakteristike poput svestranosti i jednostavnosti za početnike učinile su ga jednim od najkorištenijih programskih jezika danas. Koristi se za:

- Razvoj web aplikacija (na strani poslužitelja).
- Razvoj softvera.
- Matematiku.
- Sistemsko pisanje skripti. [13]

U ovom radu je korišten Python 3.8. verzija Pythona koja uključuje većinu potrebnih biblioteka korištenih u razvoju programske aplikacije i implementaciji NLP tehnika. Biblioteke koje nisu inherentno prisutne u ovoj verziji Pythona instalirane su putem naredbe "pip install" u naredbenom retku. Detaljnije informacije o bibliotekama prikazane su u daljnjem dijelu rada.

### *3.1.1. Python biblioteke i paketi*

Tijekom razvoja programske aplikacije bilo je nužno uvesti sve nužne biblioteke i pakete na početku koda kako bi se osigurala funkcionalnost odabranih NLP tehnika. Ovaj segment koda prikazan je na slici [Slika 8].

```
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import twitter_samples, stopwords
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
from nltk import FreqDist, classify, NaiveBayesClassifier
import re,string,random
import matplotlib.pyplot as plt
import wordcloud
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

***Slika 8:*** *Implementiranje biblioteka i paketa u kodu*

U narednim odjeljcima opisane su određene biblioteke koje su uključene, zajedno s njihovom namjenom.

#### *3.1.1.1. NumPy (Matplotlib)*

NumPy, popularna Python biblioteka, pruža snažne alate za učinkovito manipuliranje i rad s numeričkim podacima, posebno višedimenzijskim poljima, te nudi funkcionalnosti u području linearne algebre i matrica. [14]

Matplotlib je biblioteka koja je realizirana kao matematičko proširenje NumPy biblioteke i upotrebljava se za stvaranje dijagrama i njihovu integraciju u programske aplikacije. [15]

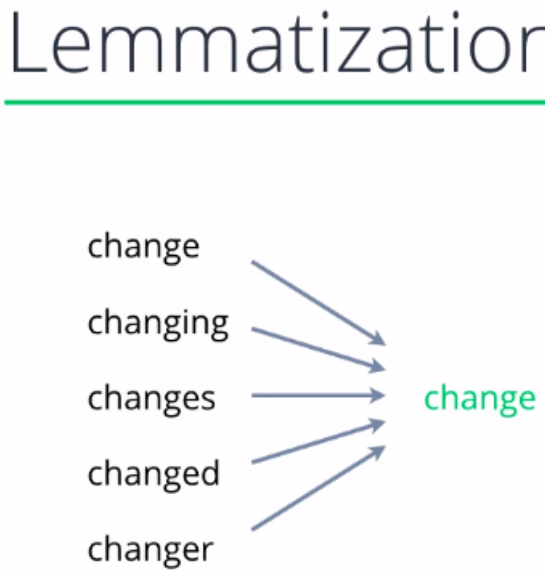
#### *3.1.1.2. NLTK*

NLTK, skraćeno od Natural Language Toolkit, je kolekcija biblioteka i programa za obradu prirodnog jezika na engleskom jeziku, napisana u programskom jeziku Python. Ova platforma omogućava funkcionalnosti kao što su klasifikacija, tokenizacija, korenovanje, označavanje, parsiranje i semantičko zaključivanje. [16]

U sklopu NLTK-a, mogu se koristiti različite tehnike za obradu teksta, kao što su WordNetLemmatizer i funkcija word_tokenize.

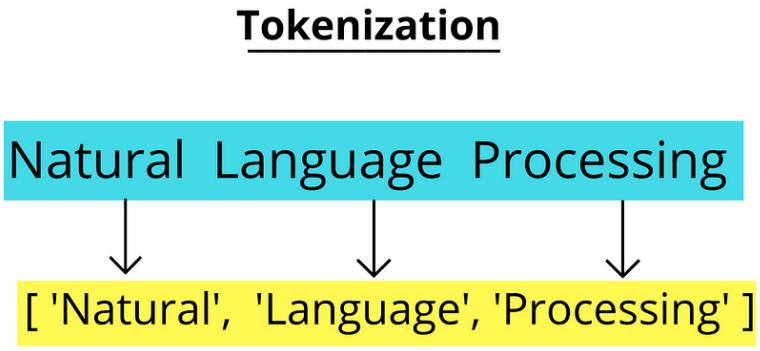
Lematizacija je proces koji grupira različite izvedene oblike iste riječi kako bi se analizirali kao jedan entitet. Lematizacija je slična korenovanju, ali donosi kontekst riječima, povezujući ih

sličnim značenjima u jednu riječ.[17] WordNetLemmatizer je biblioteka koja se uvozi iz NLTK-a, a koristi se za traženje osnovnih oblika riječi iz WordNet baze podataka.




***Slika 9:*** *Primjer lematizacije [18]*

Tokenizacija u području obrade prirodnog jezika (NLP) i strojnog učenja predstavlja proces pretvaranja teksta u manje dijelove, nazvane tokeni. Ovi tokeni mogu biti pojedinačni znakovi ili cijele riječi. Glavni cilj ovog procesa je omogućiti strojevima da bolje razumiju ljudski jezik razbijanjem teksta na manje dijelove, što olakšava analizu.[19] Funkcija word_tokenize u programskom jeziku Python razdvaja zadatu rečenicu na riječi koristeći NLTK biblioteku.




***Slika 10:*** *Primjer tokenizacije [20]*

#### *3.1.1.3. WordCloud*

WordCloud je jedna od najmoćnijih i najjednostavnijih metoda vizualizacije kada je riječ o tekstualnim podacima. Veličina riječi ovisi o frekvenciji pojavljivanja (važnosti teksta u kontekstu), stoga je korisna za projekte obrade prirodnog jezika koji koriste strojno učenje ili analizu teksta. [21]


***Slika 11:*** *Primjer WordCloud-a [21]*

WordCloud je koristan alati za vizualizaciju ključnih informacija iz različitih izvora. On sažima velike količine podataka na jasan i privlačan način, te ima višestruku primjenu:

- Vizualni alati za analizu: Pružaju kvalitativnu sliku rezultata analize teksta, posebno korisni za prezentiranje rezultata analize sentimenta ili teksta na jasan način.
- Sažimanje podataka: Efikasno sažimaju obimne skupove podataka, često korišteni za brzi pregled bitnih tačaka.
- Potiču daljnju analizu: Dok sami po sebi ne pružaju duboke uvide, potiču dalju analizu podataka ističući ključne riječi i potičući istraživanje razloga za njihovu uočljivost.

#### *3.1.1.4. Naive Bayes Classifier*

Naive Bayes algoritam je nadzirani algoritam strojnog učenja koji se temelji na Bayesovoj teoremi. To je probabilistički klasifikator koji se često koristi u zadacima obrade prirodnog jezika poput analize sentimenta (identificiranje emocionalnog ili sentimentnog tona ili mišljenja teksta). [22]

## 3.2. Analiza zadatka u Python-u

U sljedećim potpoglavljima, analizirana je izvedba zadatka u Python-u po koracima te output-ovi koji proizlaze iz dijelova kodova.

### *3.2.1. Uklanjanje šuma iz podataka*

U ovom koraku, fokus će biti na uklanjanju nepotrebnog sadržaja iz skupa podataka, što se naziva šumom. Šum obuhvaća sve dijelove teksta koji ne doprinose značenju ili informaciji podacima. U ovom radu, šum predstavlja uobičajene riječi u jeziku koje se često nazivaju „stop riječima“ i koje nisu relevantne prilikom obrade teksta. Pomoću koda sa slike [Slika 12], korišteni su regularni izrazi u Python-u kako bi određeni elementi bili pronađeni i uklonjeni poput:

- Hiperveze: Sve poveznice na Twitteru su pretvorene u URL skraćivač t.co. Stoga, zadržavanje tih poveznica u obradi teksta ne bi dodalo vrijednost analizi.
- Korisnička imena na Twitteru u odgovorima: Ta imena su obično prethodno označena simbolom @, ali ne pružaju nikakvo značenje u analizi.
- Interpunkcija i posebni znakovi: Iako ti znakovi često daju kontekst tekstualnim podacima, mogu biti teški za obradu. Stoga, jednostavno ćemo ih ukloniti iz tweetova radi jednostavnosti analize.

```
11  def remove_noise(tweet_tokens, stop_words = ()):
12      cleaned_tokens = []
13      for token, tag in pos_tag(tweet_tokens):
14          token = re.sub( pattern: 'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'\
15                          '(?:%[0-9a-fA-F][0-9a-fA-F]))+', repl: '', token) # makni URL's
16          token = re.sub( pattern: "(@[A-Za-z0-9_]+)", repl: "", token) # makni twitter handles
17
18          if tag.startswith('NN'):
19              pos = 'n' #noun
20          elif tag.startswith('VB'):
21              pos = 'v' #verb
22          else:
23              pos = 'a' #adj
24
25          lemmatizer = WordNetLemmatizer()
26          token = lemmatizer.lemmatize(token, pos)
27
28          if len(token) > 0 and token not in string.punctuation and token.lower() not in stop_words:
29              cleaned_tokens.append(token.lower()) # makni interpunkcije
30      return cleaned_tokens
```

***Slika 12:*** *Prikaz dio koda koji uklanja šum iz podataka*

Ovaj kod stvara funkciju remove_noise() koja uklanja šum i uključuje tokenizaciju i lematizaciju koje su spomenute u prethodnom potpoglavljima.

### *3.2.2. Određivanje gustoće riječi*

Najosnovniji način analize tekstualnih podataka je utvrditi frekvenciju pojavljivanja riječi. Jedan pojedinačni tweet nije dovoljno reprezentativan da bi se procijenila distribucija riječi, pa će se analiza frekvencije riječi provesti na svim pozitivnim tweet-ovima.

Na sljedećoj slici [Slika 13] definirana je generator funkcija, nazvana get_all_words, koja prima listu tweet-ova kao argument kako bi pružila listu svih riječi iz svih tokena tweetova spojenih zajedno.

```
31  # word density
    2 usages
32  def get_all_words(cleaned_tokens_list):
33      for tokens in cleaned_tokens_list:
34          for token in tokens:
35              yield token
```

***Slika 13:*** *Prikaz dio koda za gustoću riječi*

### *3.2.3. Tokenizacija podataka*

Jezik u svojem izvornom obliku ne može biti precizno obrađen od strane stroja, stoga je potrebno obraditi jezik kako bi se olakšalo razumijevanje stroju. Prvi korak u razumijevanju podataka je kroz proces koji se naziva tokenizacija, ili razdvajanje nizova u manje dijelove nazvane tokene.

Putem „`from nltk.corpus import twitter_samples`“ vidljivo na slici na prijašnjim poglavljima [Slika 8] prenešeno je tri skupa podataka iz NLTK koji sadrže različite tweet-ove za obuku i testiranje modela:

- `negative_tweets.json`: 5000 tweetova s negativnim sentimentima.

- `positive_tweets.json`: 5000 tweetova s pozitivnim sentimentima.

```
positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
stop_words = stopwords.words('english')
positive_tweet_tokens = twitter_samples.tokenized('positive_tweets.json')
negative_tweet_tokens = twitter_samples.tokenized('negative_tweets.json')
# print(positive_tweet_tokens)
print(pos_tag(positive_tweet_tokens[0]))
```

***Slika 14:*** *Prikaz koda za tokenizaciju podataka*

Nakon učitavanja koda dobiven je rezultat koji je prikazan na slici [Slika 15].

```
[('#FollowFriday', 'JJ'), ('@France_Inte', 'NNP'), ('@PKuchly57', 'NNP'), ('@Milipol_Paris', 'NNP'),
('for', 'IN'), ('being', 'VBG'), ('top', 'JJ'), ('engaged', 'VBN'),
('members', 'NNS'), ('in', 'IN'), ('my', 'PRP$'), ('community', 'NN'),
('this', 'DT'), ('week', 'NN'), (':)', 'NN')]
```

***Slika 15:*** *Prikaz rezultata od procesa tokenizacije podataka*

Iz liste oznaka, popis najčešćih stavki i njihovo značenje:

- NNP: Imenica, jednina.
- NN: Imenica, jednina ili množina.
- IN: Prijedlog ili veznik.
- VBG: Glagol, gerund ili prezent.
- VBN: Glagol, prošli particip.

Općenito, ako oznaka započinje s NN, riječ je imenica, a ako započinje s VB, riječ je glagol.

### *3.2.4. Pozitivne riječi i negativne riječi iz tweet-ova*

Ovaj dio koda koristi biblioteku WordCloud kako bi stvorio vizualnu reprezentaciju pozitivnih riječi s Twittera, iako najprije uključuje i šum. Najprije se tekst pozitivnih tweetova spaja u jedan string pomoću metode join(), a zatim se koristi funkcija WordCloud za generiranje samog oblaka riječi.

```
#wordcloud positive (before cleaning out noise)
positive_tweets_string = (' ').join(positive_tweets)
positive_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Greens').generate(positive_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(positive_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

**Slika 16:** Prikaz dio koda za vizualnu reprezentaciju pozitivnih riječi sa šumom

Učitavanjem ovog koda, dobiven je rezultat prikazan na slici [Slika 17].


***Slika 17:*** *Pozitivne riječi sa šumom*

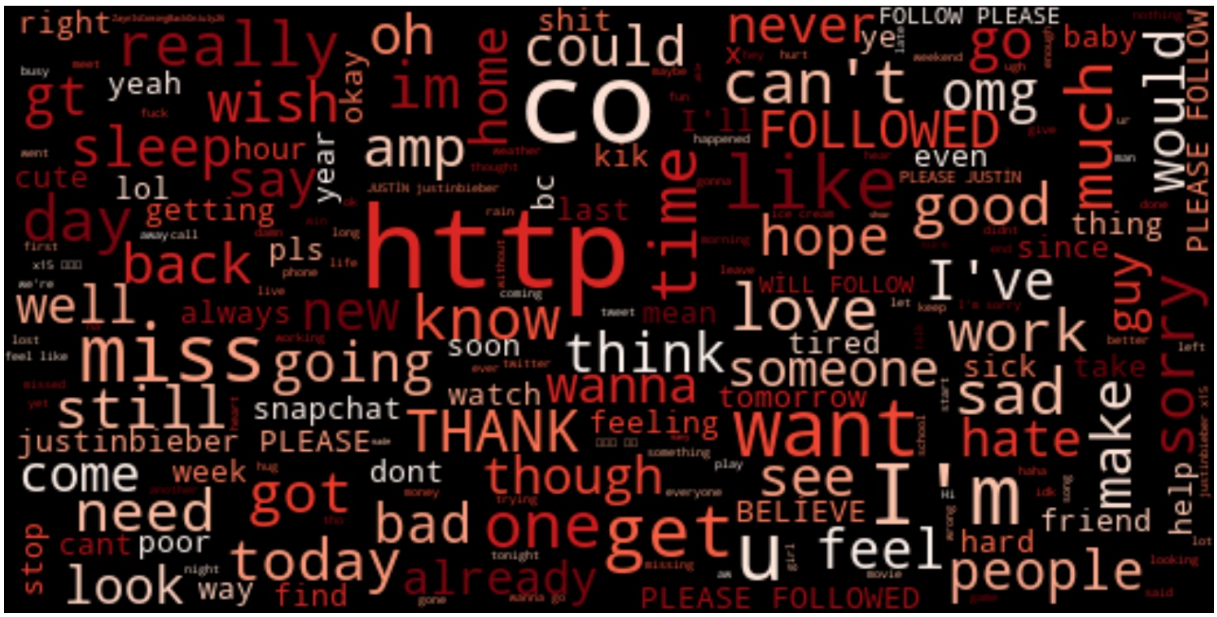
Ovaj oblak riječi prikazuje se u crnoj pozadini, s maksimalnom veličinom fonta od 50, koristeći zelenu paletu boja. Nakon toga, oblak riječi se prikazuje pomoću matplotlib biblioteke unutar matrice s dimenzijama 16x10, s isključenim oznakama osi kako bi bio fokus na riječima.

Nadalje, idući dio koda služi za stvaranje vizualizacije negativnih riječi s Twittera koristeći biblioteku WordCloud, uz uzimanje u obzir prisutnosti šuma u podacima. Prvo, negativni tweetovi se spajaju u jedan string koristeći metodu join(), kako bi se formirao jedinstveni tekst za daljnju analizu. Nakon toga, funkcija WordCloud koristi se za generiranje same vizualizacije oblaka riječi.

```
#wordcloud negative (before cleaning out noise)
negative_tweets_string = (' ').join(negative_tweets)
negative_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Reds').generate(negative_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(negative_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

***Slika 18:*** *Prikaz dio koda za vizualnu reprezentaciju negativnih riječi sa šumom*

Učitavanjem ovog koda, dobiven je rezultat prikazan na slici [Slika 19].




***Slika 19:*** *Negativne riječi sa šumom*

Ova vizualizacija riječi pojavljuje se na crnoj pozadini, s najvećom veličinom fonta postavljenom na 50, koristeći crvenu paletu boja. Zatim se generirani oblak riječi prikazuje pomoću matplotlib biblioteke u formatu matrice dimenzija 16x10, pri čemu su oznake osi isključene kako bi se naglasak stavio na same riječi.

Na slici [Slika 20] prikazan je dio koda koji kreira novu lista pozitivnih i negativnih tokena, nakon što su prošli kroz proces čišćenja od nepotrebnih sadržaja. Za svaki set tokena u pozitivnim i negativnim tweetovima, funkcija remove_noise se primjenjuje kako bi se uklonili šumovi, koristeći stop riječi za filtriranje. Nakon čišćenja, tokeni se dodaju u odgovarajuće liste pozitivnih i negativnih čišćenih tokena.

```
positive_cleaned_tokens_list = []
negative_cleaned_tokens_list = []
for tokens in positive_tweet_tokens:
 positive_cleaned_tokens_list.append(remove_noise(tokens, stop_words))

for tokens in negative_tweet_tokens:
 negative_cleaned_tokens_list.append(remove_noise(tokens, stop_words))
```

***Slika 20:*** *Prikaz koda koji uklanja šum*

U ovom dijelu koda [Slika 21] stvara se jedinstveni string pozitivnih tokena koji su prošli kroz proces čišćenja. Prvo, lista pozitivnih tokena se ravnopravno spaja u jedan niz pomoću list comprehension metode. Nakon toga, ti tokeni se spajaju u jedan string koristeći metodu join(). Zatim se stvara oblak riječi za pozitivne tokene koristeći biblioteku WordCloud. Definiraju se parametri oblaka riječi kao što su širina, visina, maksimalna veličina fonta, pozadinska boja i paleta boja. Nakon što je oblak riječi generiran, prikazuje se pomoću biblioteke matplotlib u formi matrice dimenzija 16x10, pri čemu su oznake osi isključene kako bi se fokusirali samo na vizualizaciju riječi.

```
#wordcloud positive (post cleaning out noise)
flattened_list = [item for sublist in positive_cleaned_tokens_list for item in sublist]
positive_tweets_string = ' '.join(flattened_list)
positive_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Greens').generate(positive_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(positive_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

***Slika 21:*** *Prikaz koda za vizualizaciju pozitivnih riječi bez šuma*

Nakon učitavanja koda, dobiven je rezultat koji je prikazan na slici [Slika 22].


***Slika 22:*** *Pozitivne riječi bez šuma*

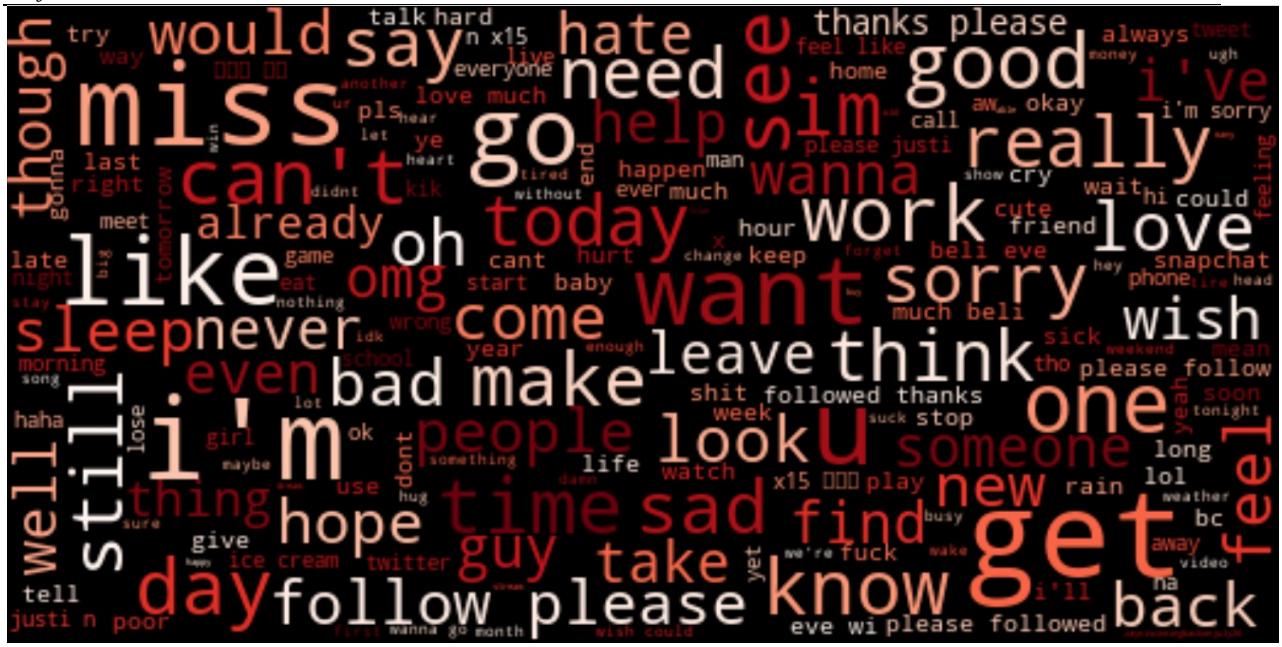
U ovom dijelu koda, stvara se jedinstveni niz negativnih tokena koji su prošli kroz proces čišćenja. Prvo, tokeni se objedinjuju u jedan niz koristeći list comprehension pristup. Nakon toga, ti tokeni se spajaju u jedan string koristeći odgovarajuću metodu.

Nakon formiranja niza negativnih tokena, generira se oblak riječi za te tokene koristeći biblioteku WordCloud. Definiraju se različiti parametri oblaka riječi, kao što su dimenzije, maksimalna veličina fonta, boja pozadine i paleta boja. Kada je oblak riječi generiran, prikazuje se pomoću biblioteke matplotlib u obliku matrice određenih dimenzija. Oznake osi su isključene kako bi se naglasila vizualizacija riječi.

```
#wordcloud negative (post cleaning out noise)
flattened_list = [item for sublist in negative_cleaned_tokens_list for item in sublist]
negative_tweets_string = ' '.join(flattened_list)
positive_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Reds').generate(negative_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(positive_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

***Slika 23:*** *Prikaz koda za vizualizaciju negativnih riječi bez šuma*

Nakon učitavanja koda, dobiven je rezultat koji je prikazan na slici [Slika 24].




***Slika 24:*** *Negativne riječi bez šuma*

U sljedećem dijelu koda [Slika 25], prvo se koristi funkcija get_all_words za dobivanje svih riječi iz liste čišćenih pozitivnih tokena. Zatim se koristi funkcija FreqDist iz biblioteke NLTK za stvaranje frekvencijske distribucije riječi. Nakon toga, ispisuje se frekvencijska distribucija svih riječi, te se ispisuju deset najčešćih pozitivne riječi i njihove frekvencije u obliku rječnika.

```
99  all_pos_words = get_all_words(positive_cleaned_tokens_list)
100 freq_dist_pos = FreqDist(all_pos_words)
101 print(freq_dist_pos)
102 print(freq_dist_pos.most_common(10))
```

***Slika 25:*** *Prikaz dio koda za ispisivanje deset najčešćih pozitivnih riječi*

Nakon učitavanja koda, rezultat je prikazan na slici [Slika 25].

```
<FreqDist with 7428 samples and 34493 outcomes>
[(':)', 3691), (':-)', 701), (':d', 658), ('thanks', 388), ('follow', 357), ('love', 333), ('...', 290), ('good', 283), ('get', 263), ('thank', 253)]
```

***Slika 26:*** *Deset najčešćih pozitivnih riječi*

U ovom segmentu [Slika 26] se kreira frekvencijska distribucija negativnih riječi iz liste čišćenih tokena. Prvo, funkcija get_all_words koristi se za dobivanje svih riječi iz liste čišćenih negativnih tokena. Zatim se ta lista riječi koristi za stvaranje frekvencijske distribucije pomoću funkcije FreqDist iz biblioteke NLTK. Nakon što je kreirana frekvencijska distribucija, ispisuje se kompletna distribucija svih riječi, te se ispisuje deset najčešćih riječi i njihove frekvencije u obliku rječnika. Ovaj proces omogućuje uvid u distribuciju riječi u negativnom kontekstu, što može biti korisno za analizu i interpretaciju sentimenta u tekstualnim podacima.

```
105  all_pos_words = get_all_words(negative_cleaned_tokens_list)
106  freq_dist_pos = FreqDist(all_pos_words)
107  print(freq_dist_pos)
108  print(freq_dist_pos.most_common(10))
```

***Slika 27:*** *Prikaz dio koda za ispisivanje deset najčešćih negativnih riječi*

Nakon učitavanja koda, rezultat je prikazan na slici [Slika 27].

```
<FreqDist with 7022 samples and 33749 outcomes>
[(':(', 4585), (':-(', 501), ("i'm", 343), ('...', 332), ('get', 325), ('miss', 291), ('go', 275), ('please', 275), ('want', 246), ('like', 218)]
```

***Slika 28:*** *Deset najčešćih negativnih riječi*

### *3.2.5. Sentiment Analysis*

Za obuku modela sentimentne analize potrebno je pripremiti skup podataka koji će se koristiti. U ovom radu, koristi se nadzirano strojno učenje, gdje se svakom podatku dodjeljuje „sentiment“ - ili pozitivan ili negativan - kako bi se model obučio. Fokus je na prepoznavanju pozitivnih i negativnih sentimenta. Nakon podjele podataka na skup za obuku i skup za testiranje, prvi dio služi za izgradnju modela, dok se drugi koristi za evaluaciju njegove učinkovitosti. Priprema podataka uključuje pretvaranje riječi u oblik razumljiv za računalo te podjelu podataka u svrhu obuke i testiranja modela.

#### *3.2.5.1. Pretvaranje tokena u rječnik*

Prvo se pripremaju podaci koji će se koristiti u modelu. Kako bi se izveo postupak modeliranja, koristi se Naivni Bayesov klasifikator u NLTK biblioteci. Važno je napomenuti da model zahtijeva ne samo listu riječi u tweetu, već Python rječnik s riječima kao ključevima i True kao

vrijednostima. Slijedeća funkcija stvara generator funkciju koja mijenja format čišćenih podataka. Nadalje, dodaje se kod koji konvertira tweetove iz liste čišćenih tokena u rječnike s riječima kao ključevima i True kao vrijednostima. Ovi rječnici pohranjuju se u varijable positive_tokens_for_model i negative_tokens_for_model.

```
38  def get_tweets_for_model(cleaned_tokens_list):
39      for tweet_tokens in cleaned_tokens_list:
40          yield dict([token, True] for token in tweet_tokens)
```

***Slika 29:*** *Prikaz dio koda za tokenizaciju*

### *3.2.5.2. Razdvajanje skupa podataka za obuku i testiranje modela*

Potrebno je pripremiti podatke za obuku klase Naive Bayes Classifier pomoću idućeg koda prikazanog na slici [Slika 29].

```
115 positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
116 negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)
117 positive_dataset = [(tweet_dict, "Positive")
118 for tweet_dict in positive_tokens_for_model]
119 negative_dataset = [(tweet_dict, "Negative")
120 for tweet_dict in negative_tokens_for_model]
121 #print(positive_dataset)
122 #print(negative_dataset)
123 dataset = positive_dataset + negative_dataset
124 random.shuffle(dataset)
125 train_data = dataset[:7000]
126 test_data = dataset[7000:]
```

***Slika 30:*** *Prikaz dio koda za razdvajanje podataka*

Ovaj kod dodjeljuje svakom tweetu oznaku „Pozitivno“ ili „Negativno“. Zatim formira skup podataka spajanjem pozitivnih i negativnih tweetova. Podrazumijevano, podaci sadrže sve pozitivne tweetove, a zatim sve negativne tweetove u nizu. Pri obuci modela važno je osigurati uzorak podataka koji ne pokazuje nikakvu pristranost. Kako bi to bilo postignuto, uključen je kod koji nasumično raspoređuje podatke korištenjem metode .shuffle() iz modula random.

Na kraju, kod podatke podijeli u omjeru 70:30 za obuku i testiranje. S obzirom na ukupan broj od 10000 tweetova, prvih 7000 se koristi za obuku modela, dok se preostalih 3000 koristi za testiranje modela.

### *3.2.6. Izgradnja i testiranje modela pomoću Naive Bayes Classifier-a*

Zadnji dio koda koristi Naive Bayes Classifier za izgradnju modela. Korišten je .train() metoda za obuku modela i .accuracy() metoda za testiranje modela na testnim podacima.

```
127    classifier = NaiveBayesClassifier.train(train_data)
128    print("Accuracy is:", classify.accuracy(classifier, test_data))
129    print(classifier.show_most_informative_features(10))
```

***Slika 31:*** *Prikaz dio koda za izgradnju modela*

Nakon učitavanja koda, rezultat je prikazan na slici [Slika 31].

```
Accuracy is: 0.9963333333333333
Most Informative Features
                     :( = True           Negati : Positi =   2064.3 : 1.0
                     :) = True           Positi : Negati =    990.4 : 1.0
                    sad = True           Negati : Positi =     34.3 : 1.0
               follower = True           Positi : Negati =     22.3 : 1.0
                   glad = True           Positi : Negati =     21.4 : 1.0
                    bam = True           Positi : Negati =     19.4 : 1.0
              community = True           Positi : Negati =     18.7 : 1.0
                    x15 = True           Negati : Positi =     18.6 : 1.0
               followed = True           Negati : Positi =     16.0 : 1.0
                   damn = True           Negati : Positi =     15.2 : 1.0
None
```

***Slika 32:*** *Prikaz točnosti i tablica najinformatijivnijih značajkih*

Točnost se definira kao postotak tweetova u testnom skupu podataka za koje je model ispravno mogao predvidjeti sentiment. Točnost od 99.6% na testnom skupu prilično je visoka.

U tablici koja prikazuje najinformativnije značajke, svaki redak u izlazu pokazuje omjer pojavljivanja tokena u pozitivnim i negativnim označenim tweetovima u skupu podataka za obuku. Prvi redak u podacima označava da je u svim tweetovima koji sadrže token „:(„ omjer negativnih u odnosu na pozitivne tweetove bio 2064.3 naprema 1. Zanimljivo je primijetiti da postoji jedan token s „:(„ u pozitivnim skupovima podataka. Može se primijetiti da su najdiskriminirajući elementi u tekstu emotikoni. Nadalje, riječi poput „sad“ vode negativnim sentimentima, dok „followed“ i „glad“ su povezane s pozitivnim sentimentima.

# 4. KRITIČKI OSVRT

Kritički osvrt na analizu sentimenta za Twitter koristeći obradu prirodnog jezika i strojno učenje pokazuje mnoge prednosti, ali i izazove koji se pojavljuju u ovom području. Analiza sentimenta omogućuje nam dublje razumijevanje mišljenja i emocija izraženih na Twitteru, što je ključno za razumijevanje raspoloženja korisnika, trendova i percepcije brendova.

Prednosti ovog pristupa uključuju mogućnost automatskog praćenja sentimenta velike količine podataka s Twittera, što omogućuje brze i široke uvide u stavove i reakcije korisnika. Također, strojno učenje omogućuje razvoj modela koji se mogu prilagoditi specifičnim potrebama i karakteristikama korisnika Twittera.

Međutim, postoje i izazovi. Twitter je poznat po svojoj raznolikosti i dinamičnosti sadržaja, što može stvarati izazove u obradi i interpretaciji teksta. Također, analiza sentimenta može biti osjetljiva na kontekst, ironiju, i „slang“, što može dovesti do netočnih rezultata.

Daljnji razvoj u ovom području zahtijevat će poboljšanja u modelima strojnog učenja kako bi se bolje nosili s ovim izazovima, kao i bolje razumijevanje specifičnosti Twitter-ovog korisničkog jezika. Integracija različitih pristupa i tehnika, poput dubokog učenja i analize slike, također bi mogla pružiti nove mogućnosti i poboljšanja u analizi sentimenta za Twitter.

# 5. ZAKLJUČAK

U zaključku, ovaj rad istražuje važnost i učinkovitost analize sentimenta putem strojnog učenja na podacima s Twittera. Kroz provedene eksperimente i evaluaciju, pokazuje se da modeli strojnog učenja, poput Naivnog Bayesovog klasifikatora, mogu uspješno prepoznati pozitivne i negativne sentimente u tweetovima. Međutim, isto tako se identificiraju izazovi u ovom pristupu, poput rukovanja s nepreciznostima u lingvističkoj obradi ili razumijevanju konteksta. S obzirom na sveukupne rezultate, ovo istraživanje potvrđuje važnost analize sentimenta za dublje razumijevanje mišljenja i emocija izraženih na društvenim medijima poput Twittera. Nadalje, otvara se prostor za daljnja istraživanja koja bi se mogla fokusirati na poboljšanja u modelima strojnog učenja kako bi se bolje nosili s izazovima specifičnim za analizu sentimenta na Twitteru.

# LITERATURA

[1] Natural Language processing Neural NLP, https://en.wikipedia.org/wiki/Natural_language_processing#Neural_NLP_(present), dostupno dana: 12.02.2024.

[2] ELIZA, https://en.wikipedia.org/wiki/ELIZA, dostupno dana: 12.02.2024.

[3] Neural network, https://en.wikipedia.org/wiki/Neural_network, dostupno dana: 12.02.2024.

[4] Statistical model, https://en.wikipedia.org/wiki/Statistical_model, dostupno dana: 12.02.2024.

[5] Speech recognition, https://en.wikipedia.org/wiki/Speech_recognition, dostupno dana: 12.02.2024.

[6] How to build domain specific automatic speech recognition models on gpus, https://developer.nvidia.com/blog/how-to-build-domain-specific-automatic-speech-recognition-models-on-gpus/, dostupno dana: 12.02.2024.

[7] Fundamentals of statistical natural language processing, https://www.proxet.com/blog/fundamentals-of-statistical-natural-language-processing, dostupno dana: 12.02.2024.

[8] Umjetna neuronska mreža, https://hr.wikipedia.org/wiki/Umjetna_neuronska_mre%C5%BEa, dostupno dana: 12.02.2024.

[9] A friendly introduction to deep neural networks, https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks, dostupno dana: 12.02.2024.

[10] Sentiment analysis, https://en.wikipedia.org/wiki/Sentiment_analysis, dostupno dana: 12.02.2024.

[11] Sentiment analysis, https://monkeylearn.com/sentiment-analysis/, dostupno dana: 12.02.2024.

[12] Natural language processing applications, https://datasciencedojo.com/blog/natural-language-processing-applications/#, dostupno dana: 12.02.2024

[13] Python intro, https://www.w3schools.com/python/python_intro.asp, dostupno dana: 12.02.2024.

[14] Numpy intro, https://www.w3schools.com/python/numpy/numpy_intro.asp, dostupno dana: 12.02.2024.

[15] Matplotlib, https://matplotlib.org/, dostupno dana: 12.02.2024.

[16] Natural Language Toolkit, https://en.wikipedia.org/wiki/Natural_Language_Toolkit, dostupno dana: 12.02.2024.

[17] Python lemmatization with NLTK, https://www.geeksforgeeks.org/python-lemmatization-with-nltk/, dostupno dana: 12.02.2024.

[18] Stemming vs lemmatization in NLP, https://nirajbhoi.medium.com/stemming-vs-lemmatization-in-nlp-efc280d4e845, dostupno dana: 12.02.2024.

[19] What is tokenization, https://www.datacamp.com/blog/what-is-tokenization, dostupno dana: 12.02.2024.

[20] Tokenization techniques in natural language processing, https://medium.com/@ajay_khanna/tokenization-techniques-in-natural-language-processing-67bb22088c75, dostupno dana: 12.02.2024.

[21] Word cloud or tag cloud in Python, https://www.analyticsvidhya.com/blog/2020/10/word-cloud-or-tag-cloud-in-python/, dostupno dana: 12.02.2024.

[22] Building Naive Bayes Classifier from scratch to perfrom sentiment analysis, https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/, dostupno dana: 12.02.2024.

# PRILOZI

I. Python kod

```python
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import twitter_samples, stopwords
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
from nltk import FreqDist, classify, NaiveBayesClassifier
import re,string,random
import matplotlib.pyplot as plt
import wordcloud
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
# micemo stop words
2 usages
def remove_noise(tweet_tokens, stop_words = ()):
    cleaned_tokens = []
    for token, tag in pos_tag(tweet_tokens):
        token = re.sub( pattern: 'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'\
                        '(?:%[0-9a-fA-F][0-9a-fA-F]))+', repl: '', token) # makni URL's
        token = re.sub( pattern: "(@[A-Za-z0-9_]+)", repl: "", token) # makni twitter handles

        if tag.startswith('NN'):
            pos = 'n' #noun
        elif tag.startswith('VB'):
            pos = 'v' #verb
        else:
            pos = 'a' #adj

        lemmatizer = WordNetLemmatizer()
        token = lemmatizer.lemmatize(token, pos)

        if len(token) > 0 and token not in string.punctuation and token.lower() not in stop_words:
```

```
            cleaned_tokens.append(token.lower()) # makni interpunkcije
    return cleaned_tokens
# word density
2 usages
def get_all_words(cleaned_tokens_list):
    for tokens in cleaned_tokens_list:
        for token in tokens:
            yield token

# converting tokens to a dictionary
def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
         yield dict([token, True] for token in tweet_tokens)

positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
stop_words = stopwords.words('english')
positive_tweet_tokens = twitter_samples.tokenized('positive_tweets.json')
negative_tweet_tokens = twitter_samples.tokenized('negative_tweets.json')
# print(positive_tweet_tokens)
print(pos_tag(positive_tweet_tokens[0]))
#wordcloud positive (before cleaning out noise)
positive_tweets_string = (' ').join(positive_tweets)
positive_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'white', colormap = 'Greens').generate(positive_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(positive_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()

negative_tweets_string = (' ').join(negative_tweets)
negative_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Reds').generate(negative_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(negative_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
# output su negativne rijeci s noise
# output su negativne rijeci s noise

positive_cleaned_tokens_list = []
negative_cleaned_tokens_list = []
for tokens in positive_tweet_tokens:
 positive_cleaned_tokens_list.append(remove_noise(tokens, stop_words))

for tokens in negative_tweet_tokens:
 negative_cleaned_tokens_list.append(remove_noise(tokens, stop_words))

#print(positive_cleaned_tokens_list)

#wordcloud positive (post cleaning out noise)
flattened_list = [item for sublist in positive_cleaned_tokens_list for item in sublist]
positive_tweets_string = ' '.join(flattened_list)
positive_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Greens').generate(positive_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(positive_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
```

```
#wordcloud negative (post cleaning out noise)
flattened_list = [item for sublist in negative_cleaned_tokens_list for item in sublist]
negative_tweets_string = ' '.join(flattened_list)
positive_tweets_cloud = WordCloud(width = 520, height = 260, stopwords = stop_words, max_font_size = 50,
                                  background_color = 'black', colormap = 'Reds').generate(negative_tweets_string)
plt.figure(figsize=(16,10))
plt.imshow(positive_tweets_cloud, interpolation = 'bilinear')
plt.axis('off')
plt.show()
#output je negativne rijeci bez noise

all_pos_words = get_all_words(positive_cleaned_tokens_list)
freq_dist_pos = FreqDist(all_pos_words)
print(freq_dist_pos)
print(freq_dist_pos.most_common(10))


all_pos_words = get_all_words(negative_cleaned_tokens_list)
freq_dist_pos = FreqDist(all_pos_words)
print(freq_dist_pos)
print(freq_dist_pos.most_common(10))
2 usages
def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
        yield dict([token, True] for token in tweet_tokens)

positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)
positive_dataset = [(tweet_dict, "Positive")
for tweet_dict in positive_tokens_for_model]
    negative_dataset = [(tweet_dict, "Negative")
    for tweet_dict in negative_tokens_for_model]
    #print(positive_dataset)
    #print(negative_dataset)
    dataset = positive_dataset + negative_dataset
    random.shuffle(dataset)
    train_data = dataset[:7000]
    test_data = dataset[7000:]
    classifier = NaiveBayesClassifier.train(train_data)
    print("Accuracy is:", classify.accuracy(classifier, test_data))
    print(classifier.show_most_informative_features(10))
```