

# Implementacija transformer modela u *PyTorch*-u za zadatak klasifikacije sentimenta

Student: Ana Petrović 3176/2024

Beograd, 2025.

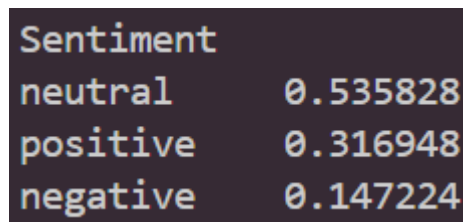
<b>1. Uvod</b>	<b>3</b>
<b>2. Izbor skupa podataka</b>	<b>3</b>
<b>3. Tehnološki stack</b>	<b>3</b>
<b>4. Metodologija</b>	<b>3</b>
a. Pipeline i organizacija projekta	3
b. Konfiguracija konstanti	5
c. Obrada i analiza podataka	6
d. Klasa Dataset	7
e. Skripta transformer.py	7
f. Pokretanje projekta i treniranje modela	7
<b>5. Evaluacija i rezultati</b>	<b>8</b>
<b>6. Zaključak</b>	<b>10</b>

# 1. Uvod

U ovom projektu primenjen je jednostavan transformer korišćenjem biblioteke *Pytorch* za problem analize sentimenta nad finansijskim skupom podataka. Analiza sentimenta je važna zbog predviđanja stanja tržišta, analize izveštaja kompanija, i slično. U ovom zadatku eksperimentisano je sa transformer arhitekturom u odnosu na klasične metode, koja bi trebalo da omogući modelu da razume dugačke rečenice, kao i da bolje razume kontekst u složenom jeziku finansijskih dokumenata. Cilj zadatka je klasifikovani sentiment rečenice na pozitivne, neutralne ili negativne, korišćenjem pažljivo pripremljenog *pipeline*-a za obradu podataka i treniranje modela.

## 2. Izbor skupa podataka

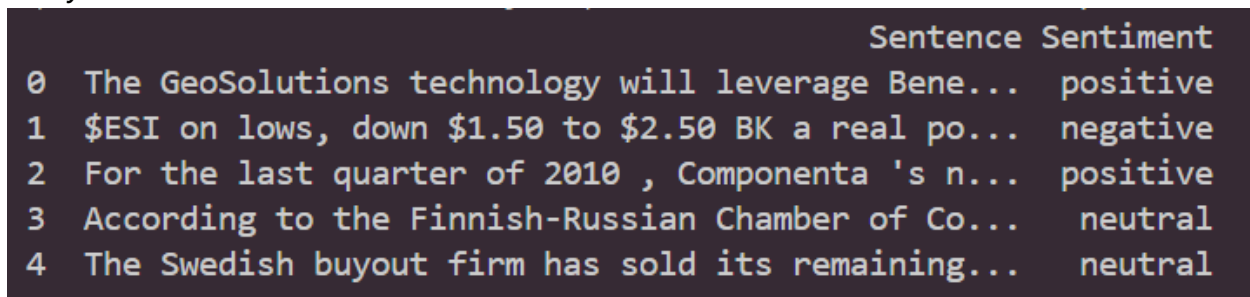
Odabran je *Financial Sentiment Analysis* dataset. Sadrži dve kolone: *Sentence* i *Sentiment*. Ima 5842 reda. Kolona *Sentence* opisuje različita finansijska stanja kompanija, a kolona *Sentiment* ima 3 različite vrednosti koje opisuju sentiment rečenice iz kolone *Sentence*: *positive*, *neutral*, *negative*. Ovaj dataset može se pronaći na *Kaggle*-u: [link](#). Format skupa podataka je CSV. Dodatnom analizom distribucije izlazne varijable zaključeno je da je ovaj skup podataka prilično nebalansiran, gde klasa *neutral* dominira sa 45% primera, a klasa *negative* je veoma slabo zastupljena, sa samo 15% primera, dok klasa *positive* ima oko 31% primera.



Sentiment	
neutral	0.535828
positive	0.316948
negative	0.147224

Slika: Distribucija klasa

Na fotografiji ispod može se videti Izgled prvih pet redova skupa podataka *Financial Sentiment Analysis*:



	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	positive
1	\$ESI on lows, down \$1.50 to \$2.50 BK a real po...	negative
2	For the last quarter of 2010 , Componenta 's n...	positive
3	According to the Finnish-Russian Chamber of Co...	neutral
4	The Swedish buyout firm has sold its remaining...	neutral

Slika: Prvih pet redova skupa podataka data.csv

### 3. Tehnološki *stack*

U sledećoj tabeli prikazan je osnovni tehnološki *stack* korišćen prilikom izrade projekta.

<b>Razvojno okruženje</b>	<i>VSCode</i>
<b>Programski jezik</b>	<i>Python</i>
<b>Biblioteke i paketi</b>	Standard <i>Python</i> library, <i>PyTorch</i> , <i>pandas</i> , <i>numpy</i> , <i>scikit-learn</i> , <i>pickle</i> , <i>matplotlib</i> , <i>seaborn</i>

Tabela: *Tehnološki stack*

### 4. Metodologija

U narednom poglavlju biće opisan način organizacije projekta i tok rada.

#### a. Pipeline i organizacija projekta

U ovom projektnom radu ispoštovani su osnovni principi modularnosti prilikom projektovanja modela mašinskog učenja. Svaki deo *pipeline*-a je razdvojen u posebnu skriptu, čime se postiže bolja čitljivost i lakše održavanje.

- `data/`: sadrži originalne i obrađene podatke
- `imgs/`: sadrži slike sačuvanih metrika
- `config.py`: definiše sve konstante
- `dataset.py`: sadrži klasu za *Dataset*
- `evaluate.py`: vrši evaluaciju modela
- `preprocessing.py`: obrađuje sirove podatke
- `run.py`: centralno mesto pokretanja *pipeline*-a
- `train.py`: trenira model
- `transformer.py`: definiše arhitekturu modela

Projekat se pokreće iz dva koraka:

- 1) Pokretanjem `preprocessing.py` skripte
- 2) Pokretanjem `run.py` skripte

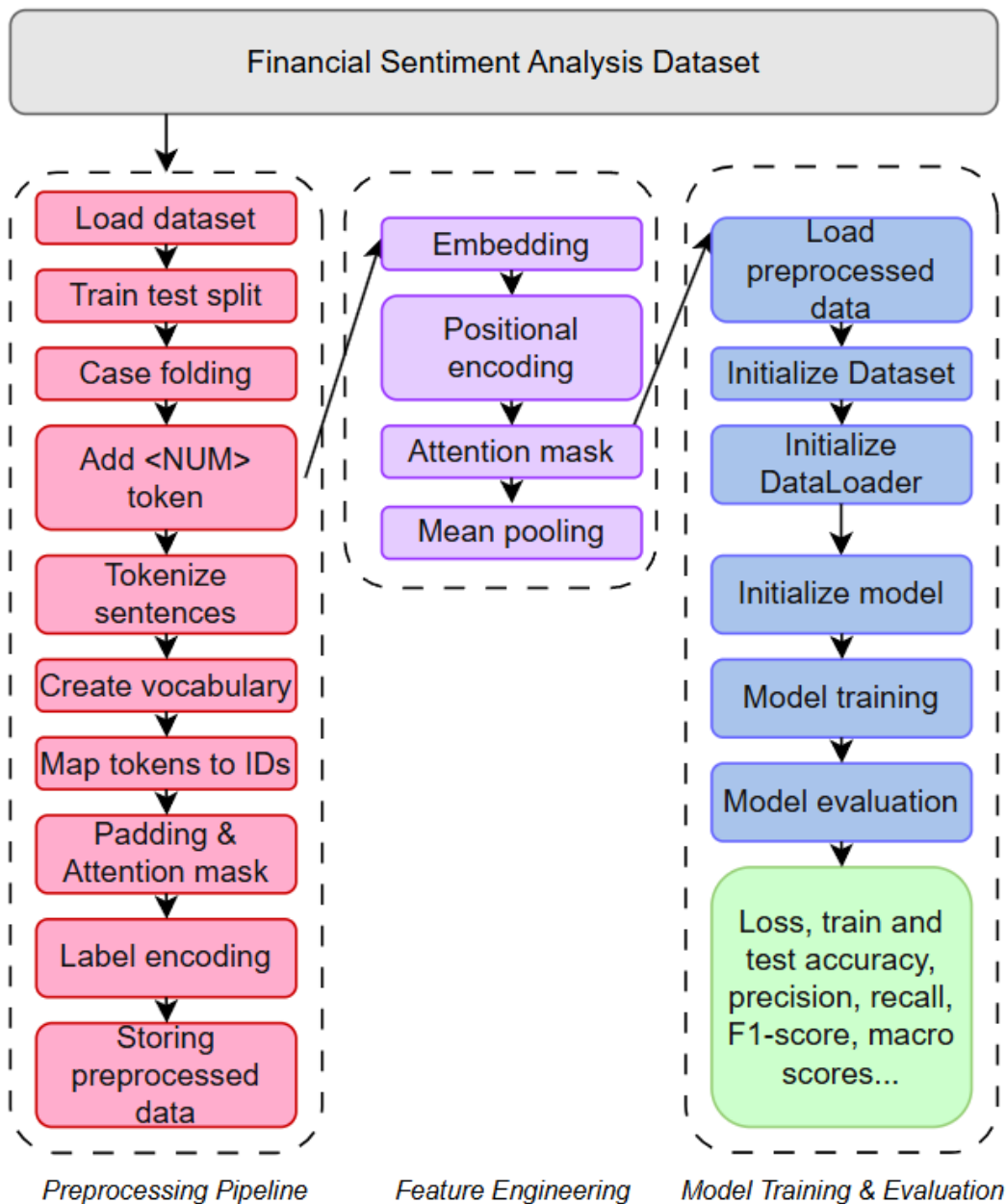
Izlaz iz `preprocessing.py` skripte su obrađeni *train* i *test* skupovi originalnog datasea, *padding\_id* i *vokabular* u *pickle* formatu, koji je koristan za čuvanje velikih formata u vidu binarnih *byte stream*-ova i svojstven je isključivo *Python* programskom jeziku.

Skripta `run.py` objedinjuje sve ostale korake u projektnom radu i njenim pokretanjem izvršava se učitavanje prethodno sačuvanih podataka, inicijalizovanje klasa *Dataset* i *DataLoader*, inicijalizovanje modela i pokretanje *training* i *test* pelje.

Na slici ispod vizuelno je prikazan redosled osnovnih koraka u ovom projektu. Projekat se može izdeliti na tri faze - u prvoj fazi se podaci pripremaju za ulaz u model, a čine ga: podela na učitavanje podataka, trening i test, prebacivanje na mala slova, dodavanje tokena koji označava brojeve, deljenje rečenice na reči na osnovu praznine, kreiranje vokabulara na osnovu postojećih rečenica, mapiranje tokena ka ID-jevima iz vokabulara, dodavanje *padding*-a i *attention* maske, enkodiranje izlazne varijable i čuvanje obrađenih podataka, vokabulara i *padding id*-a.

U drugoj fazi vrši se *embedding* tokena i dodavanje informacije o redosledu reči putem pozicionog enkodiranja. Zatim se primenjuje *attention* maska, a reprezentacija rečenice izvlači se metodom *mean pooling*-a. *Embedding* podrazumeva izvlačenje *ID*-jeva iz *tokena* i konvertovanje u vektore značenja koji se koriste za dalje učenje. *Mean pooling* je metoda kojom se svi vektori u jednoj rečenici kombinuju u jedan vektor, tako što se računa srednja vrednost svih vektora u reči.

U poslednjoj fazi podaci se pripremaju za treniranje modela - učitavaju se u *DataLoader*, prosleđuju modelu, optimizuju se parametri i na kraju model se evaluira.



Slika: Pipeline

## b. Konfiguracija konstanti

U skripti [config.py](#) definisane su konstante koje su dostupne u celom projektu i korišćene od strane ostalih skriptata, kao što su: [run.py](#), [evaluate.py](#), [train.py](#). `MIN_FREQ = 2` služi da otkloni token koji se retko pojavljuju i smanjuje šum i štedi memoriju. `MAX_LEN = 64` je postavljen na osnovu analize dužina rečenica u skupu podataka, i zaključeno je da ova dužina obuhvata najveći broj rečenica i da ne predstavlja previše računarski zahtevan problem.

```

MAX_LEN = 64
PAD_TOKEN = "<PAD>"
UNK_TOKEN = "<UNK>"
MIN_FREQ = 2
LABEL2ID = {"positive": 0, "neutral": 1, "negative": 2}
ID2LABEL = {v: k for k, v in LABEL2ID.items()}

```

Slika 1: Skripta *config.py*

### c. Obrada i analiza podataka

Obrada podataka započinje učitavanjem CSV fajla.

Nakon toga, skup podataka deli se na trening i test skup. Trening skup čini 70% originalnog skupa, a test skup 30%. Korišćen je *stratify* parametar za varijablu *Sentiment* koji čuva odnos zastupljenosti klasa u trening i test skupu, s obzirom da vrednosti nisu ujednačeno balansirane. Sledeći korak odnosi se na prevođenje celog tekstualnog sadržaja skupova podataka na mala slova, što dovodi do uniformnijeg rešenja.

Tokenizacija podrazumeva odvajanje reči po razmaku. Tokenizacija može biti i složenija, ali u ovom slučaju primenjen je najjednostavniji slučaj, koji odvaja reči samo po belinama.

Train set tokenized		
	Sentence	Sentiment
26	[costco:, a, premier, retail, dividend, play, ...	positive
5197	[the, diameter, protocol, is, developed, accor...	neutral
4070	[glaston, 's, share, gla1v, is, listed, on, th...	neutral
5704	[operating, result, ,, excluding, one-off, ite...	neutral
2765	[the, company, is, presently, examining, wheth...	neutral

Slika: *Tokenizacija*

Kreiranje vokabulara je neizostavan proces za zadatak analize sentimenta korišćenjem *transformer* arhitekture. Vokabular predstavlja spisak svih jedinstvenih reči u skupu podataka. U ovom primeru izostavljene su sve reči koje se pojavljuju manje od dva puta jer su previše specifične ili ne bi doprinele kvalitetu modela.

Nakon kreiranja vokabulara, neophodno je mapirati tokene sa *ID*-jevima.

Train set with mapped tokens			
	Sentence	Sentiment	input_ids
26	[costco:, a, premier, retail, dividend, play, ...	positive	[2, 3, 4, 5, 6, 7, 1, 8]
5197	[the, diameter, protocol, is, developed, accor...	neutral	[9, 1, 10, 11, 12, 13, 14, 9, 15, 16, 17, 18, ...
4070	[glaston, 's, share, gla1v, is, listed, on, th...	neutral	[21, 22, 23, 24, 11, 25, 26, 9, 27, 28, 29, 30...
5704	[operating, result, ,, excluding, one-off, ite...	neutral	[34, 35, 30, 36, 37, 38, 30, 39, 40, 18, 41, 4...
2765	[the, company, is, presently, examining, wheth...	neutral	[9, 46, 11, 47, 48, 49, 9, 50, 51, 52, 53, 1, 20]

Slika: Izgled tokenizovanih rečenica sa broječanim vrednostima (tokena) za svaku reč

Poslednji korak u pripremi podataka za treniranje modela odnosi se na dodavanje *padding*-a i *attention mask*-e. Za maksimalnu dužinu sekvence treba pažljivo odabrati broj na osnovu analize maksimalne, minimalne i prosečne dužine tokena za svaki red u skupu podataka. Manja dužina sekvence dovodi do gubitka informacija, a veoma dugačke sekvence mogu biti računarski skupe. Eksperimentisanjem različitim vrednostima ovog parametra može se doći do optimizovanih rešenja.

```
Min length: 2
Max length: 60
Average length: 20.94
Median length: 19.0
```

Slika: Analiza dužina rečenica

Sentence	[costco:, a, premier, retail, dividend, play, ...
Sentiment	positive
input_ids	[2, 3, 4, 5, 6, 7, 1, 8]
padded_input_ids	[2, 3, 4, 5, 6, 7, 1, 8, 0, 0, 0, 0, 0, 0, 0, ...
attention_mask	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, ...
label	0

Slika: Primer tokenizovane rečenice, sentimenta, ulaznih i padded id-jeva, attention maske i izlaza

Na samom kraju, prevode se kategoričke ocene rečenica u numeričke.

U pickle formatu čuvaju se prerađeni skupovi podataka, vokabular i PAD\_ID.

## d. Klasa Dataset

Ova klasa prima gotov *DataFrame*, čije podatke konvertuje u liste radi lakšeg korišćenja. Omogućava indeksiran pristup uzorcima u obliku tenzora i upotrebu *DataLoader*-a prilikom treniranja, što čini projekat modularnim.



```

class FinancialDataset(Dataset):
    def __init__(self, df):
        self.input_ids = df['padded_input_ids'].tolist() # Lists of padded tokens until MAX_LEN
        self.attention_masks = df['attention_mask'].tolist() # Masks that show if tokens are real (1) or masks (0)
        self.labels = df['label'].tolist() # Numerical IDs for sentiment

    def __len__(self):
        return len(self.labels) # Get the length of a sample

    def __getitem__(self, idx): # Load a sample by index in a dictionary form
        return {
            'input_ids': torch.tensor(self.input_ids[idx], dtype=torch.long), # Input ID tokens; PyTorch expects type Long for NLP models
            'attention_mask': torch.tensor(self.attention_masks[idx], dtype=torch.long), # Attention mask
            'label': torch.tensor(self.labels[idx], dtype=torch.long) # Label
        }

```

Slika: Klasa Dataset

## e. Skripta transformer.py

Ovde se definiše transformer model za klasifikaciju teksta. Učitava se *ID* za *padding* token, koji se koristi za zanemarivanje praznih mesta u sekvencama. Klasa *PositionalEncoding* dodaje informacije o položaju reči u sekvenci tako što računa sinusne i kosinusne funkcije, što omogućava modelu da prepozna redosled reči. Glavna klasa *TransformerClassifier* implementira sloj za učenje ugrađenih vektora reči. Izlaz transformera se agregira sa srednjom vrednošću vektora, prolazi kroz *dropout* za regularizaciju, i kroz linearni sloj za dobijanje konačnih rezultata klasifikacije. Model je treniran na 50 epoha, learning rate je  $1e-4$ , a korišćen optimizator je Adam, dropout je 0.3, a veličina test seta je sa 20% povećana na 30%, kako bi se dobili što bolji rezultati.

Broj epoha	50
Learning rate	$1e-4$
Optimizator	Adam
Dropout	0.3

Tabela: Vrednosti hiperparametara

## f. Pokretanje projekta i treniranje modela

U training petlji prolazi se kroz epohe, računa se gubitak, backpropagation i optimizacija, prate se metrike i štampaju njihove vrednosti na trening setu. Definiše se funkcija za treniranje modela koja koristi balansirane težine klasa kako bi se ublažila neravnoteža u podacima. Na osnovu oznaka iz trening skupa računa se težina za svaku klasu, koje se prosleđuju funkciji gubitka kako bi model strožije kažnjavao greške na ređim klasama. Evaluacija i rezultati u ovom delu rada izračunate su i prikazane sve relevantne klasifikacione metrike. Rezultati su sačuvani u grafovima.

Model je odlično naučio podatke na trening skupu, ali je predvideo tačno oko 66.5% primera iz test skupa, što je solidno, ali ima prostora za poboljšanje. S obzirom na distribuciju klasa (*support*), može se videti da je dominantna klasa *neutral*, a minorna klasa *negative*.

Za klasu *positive*, *precision* je 72%, što znači da je od svih primera koji su klasifikovani kao pozitivni, malo više od dve trećine je zaista pozitivno. *Recall* iznosi 65%, što znači da je od svih *positive* primera toliko ispravno prepoznato. *F1-score* je 68%, što predstavlja solidnu ravnotežu između preciznosti i odziva.

Klasa *neutral* je najbolje predviđena klasa, što je i očekivano, s obzirom na to da je najzastupljenija. *Precision* je 76%, *Recall* 73% i *F1-score* 75%.

Što se tiče klase *negative*, primećuje se da model ima znatno lošije performanse. *Precision* je 34%, *Recall* 45% i *F1-score* 38%. Najveći problem je nabalansirano originalnog skupa podataka.

<b>Epoch 50/50</b>	<b>Loss: 0.1638</b>	<b>Accuracy: 0.9137</b>
<b>Test Accuracy: 0.6657</b>		

Tabela: Broj epoha, gubitak, train i test accuracy

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Positive</b>	0.72	0.65	0.68	556
<b>Neutral</b>	0.76	0.73	0.75	939
<b>Negative</b>	0.34	0.45	0.38	258

Tabela: Izveštaj klasifikacije po klasi

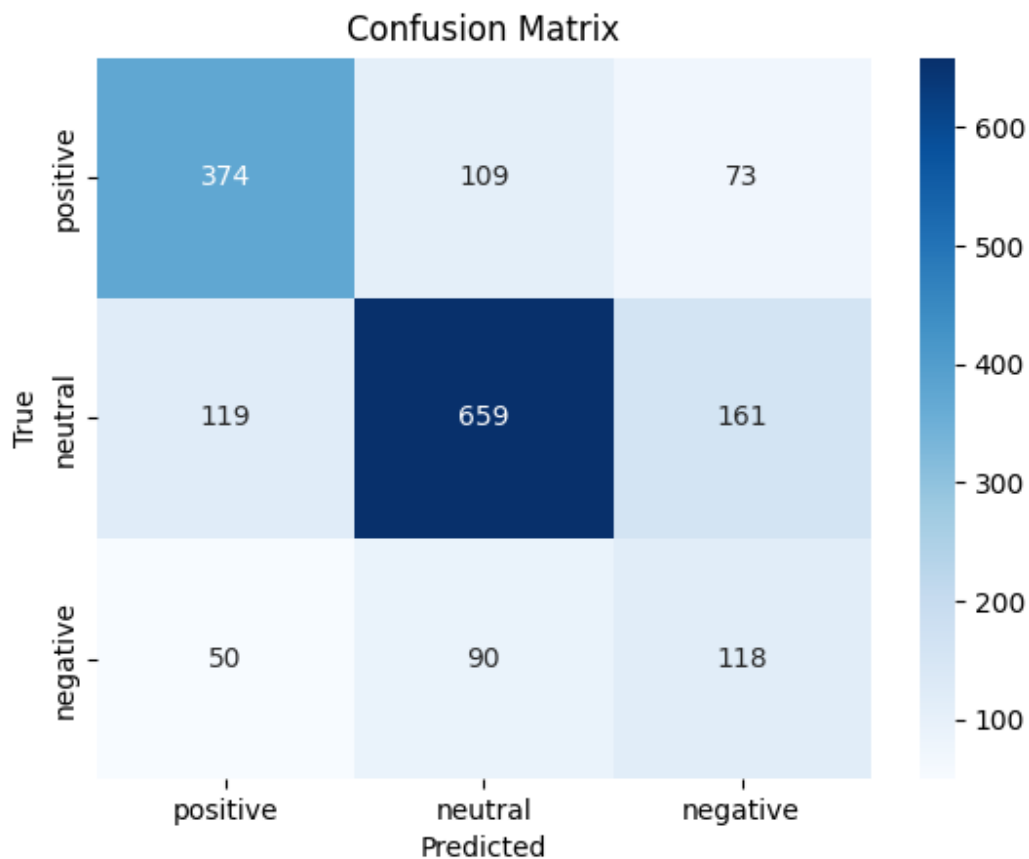
*Averaged* metrike posmatraju prosek po klasama, bez obzira na broj primera. Sve metrike su u opsegu između 60 i 61%, što nije idealno, najviše zbog problema sa negativnom klasom.

*Weighted* metrike uzimaju u obzir težine klasa, i one su nešto više od prosečnih metrika, s obzirom da dominira *neutral* klasa koja izvlači prosek.

	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Support</b>
<b>Accuracy</b>			0.67	1753
<b>Macro average</b>	0.61	0.61	0.60	1753
<b>Weighted average</b>	0.68	0.67	0.67	1753

Tabela: Izveštaj klasifikacije po svim klasama; bez težina i sa težinama klasa

Na slici ispod nalazi se matrica konfuzije koja ukazuje na to da model najbolje pogađa neutralne sentimente, što je i logično, s obzirom da ih ima najviše. Zatim, model solidno pogađa i pozitivne sentimente, jer su oni drugi po zastupljenosti u skupu podataka. Iako je treniran na 50 epoha, model se najviše muči sa negativnim sentimentima, jer ih i ima najmanje u skupu podataka. Može se videti i da model čak 161 put pogrešno klasifikuje neutralne sentimente kao negativne. Moguće je i da reprezentacija tokena nike dovoljno informativna.



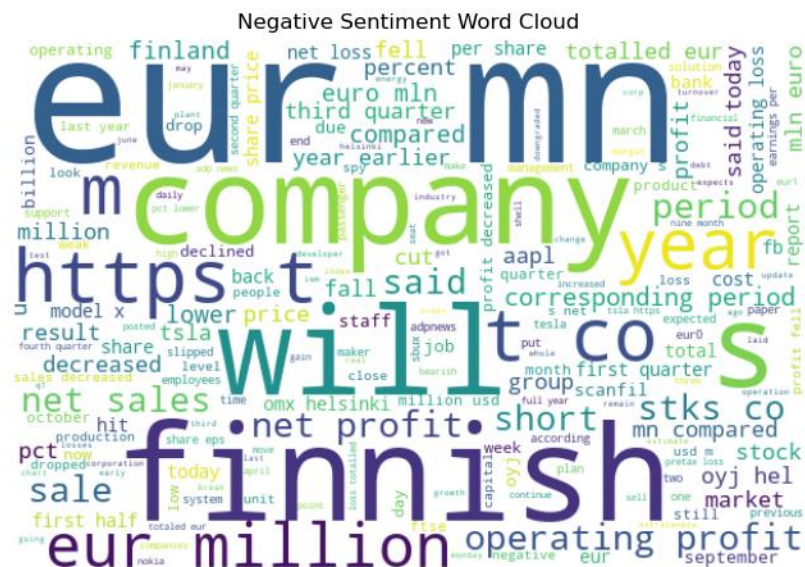
Slika: *Matrica konfuzije*

[illegible]

Najzastupljenije reči u pozitivnoj klasi su *company, finnish, eur, year, will, net sales, profit...*

Slika: Word Cloud - neutralni sentimenti

Najzastupljenije reči u neutralnom sentimentu su takođe *company, finnish, eur, year i net sales*. Specifičnije reči su *group, service, business, million*, itd.



Slika: Word Cloud - negativni sentiment

Analizom najzastupljenijih reči i u negativnoj klasi, može se videti da se reči ponavljaju, i da su vrlo slične između svih klasa, a ponajviše između negativne i neutralne klase, što ukazuje i na slabu heterogenost između klasa i potencijalno loš kvalitet skupa podataka, zbog čega su i sami rezultati analize sentimenta lošiji.

## 5. Zaključak

Problem nebalansiranog skupa podataka za klasu negative može se rešiti pribavljanjem novih podataka, kreiranjem veštačkih podataka ove klase (*oversampling*), ili smanjivanjem zastupljenosti dominantnih klasa (*undersampling*). Takođe, može se posvetiti više vremena za kreiranje specifičnih tokena koji bolje reprezentuju semantiku iz finansijske industrije. Stoga je potrebno dobro istražiti termine, značenja i najčešće probleme koji se javljaju u industrijama iz kojih je potrebno uraditi klasifikaciju sentimenta. Čak ni uvećanje test seta i korišćenje težina prilikom računanja gubitka nisu značajno doprineli poboljšanju rezultata. Dalji koraci bi, takođe, mogli da uključuju i eksperimentisanje sa višim vrednostima *MAX\_LEN*, i integraciju unapred treniranih modela kao što je *FinBERT*, koji je upravo obučen nad finansijskim skupovima podataka. Takođe, moglo je i koristiti drugi *embedding*, kao što je *GloVe* ili *BERT*, ali zbog pojednostavljenja zadatka napravljen je jednostavan *custom embedding*. S obzirom da se radi o

finansijskom skupu podataka, dosta teksta sadrži brojeve, koji bi se tretirali kao jedinstveni tokeni, što može uticati loše na performanse modela. Iz tog razloga, potrebno je razmotriti dodavanje *<NUM>* tokena, u koji bi bili prebačeni svi brojevi, što može doprineti boljoj generalizaciji.