Univerzitet u Beogradu - Elektrotehnički fakultet

PROJEKTNI ZADATAK

Quora parovi pitanja:
Možete li da identifikujete parove pitanja koji imaju istu nameru?

Student:

Antonija Vasiljević 2023/3301

Beograd, *januar* 2024. godine

1. Uvod

Quora je mesto za sticanje i deljenje znanja. To je platforma za postavljanje **pitanja** i povezivanje sa ljudima koji daju jedinstvene uvide i kvalitetne odgovore. Sa preko **100 miliona posetilaca mesečno** koji traže i dele znanje o različitim temama, Quora napreduje kao zajednica koja podstiče intelektualnu razmenu.

Izazov sa kojim se platforma suočava je ukorenjen u ogromnom angažovanju njenih korisnika. Kako pitanja pristižu, pojava sličnih upita je neizbežna. Ovaj višak predstavlja značajnu prepreku, primoravajući korisnike da se kreću kroz više verzija suštinski istog pitanja i opterećujući pisce ponavljanjem odgovora.

Quora, prepoznajući važnost rešavanja ovog problema, trenutno koristi **Random forest model** da identifikuje pitanja koja su duplikati. Međutim, dinamična priroda jezika i sadržaja koji generišu korisnici zahtevaju sofisticiranija rešenja. U ovom Kaggle takmičenju, pozvani smo da primenimo napredne tehnike obrade prirodnog jezika kako bismo klasifikovali da li su parovi pitanja duplikati.

U ovoj beležnici ćemo se upustiti u zamršenost problema, pažljivo analizirati dostavljene podatke i primeniti raznovrstan niz modela **mašinskog učenja i dubokog učenja**. Dok se krećemo kroz ovo istraživanje, koristićemo najsavremenija rešenja kao što su **Sijamske mreže** i **Word2Vec** da bismo pomerili granice razumevanja i klasifikacije prirodnog jezika.

2. Uvoz odgovarajućih paketa

```
!pip install seaborn wordcloud
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re # Koriste se regularni izrazi za čišćenje podataka
import warnings
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
pd.set_option('display.max_colwidth', None)

from wordcloud import WordCloud
import warnings
warnings.filterwarnings('ignore')

from sklearn.metrics import jaccard_score
from nltk.metrics import edit_distance
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore", category=UserWarning,
module="tensorflow")
```

3. Istraživačka analiza podataka

Ovaj odeljak će se baviti skupom podataka Quora da bi se bolje razumele karakteristike ulaznih podataka. Skup podataka se sastoji od četiri kolone: `id`, `qid1`, `qid2` i `is_duplicated`, gde `is_duplicated` označava da li su dva pitanja povezana ili ne.

Istraživačka analiza podataka se sastoji iz više faza, uključujući:

1. Pregled podataka: Za početak, ispitivaće se osnovne strukture skupa podataka, istražujući prvih nekoliko redova radi sticanja uvida u podatke.

2. Distribucija podataka: Istražiće se distribucija ciljne promenljive, `is_duplicated`, da bi se razumela ravnoteža između dupliranih i neduplikanih parova pitanja.

3. Analiza teksta: Izvršiće se tekstualne analize pitanja 1 i pitanja 2, istražujući učestalost reči, uobičajene fraze i sve značajne obrasce.

4. Čišćenje podataka: Ako je potrebno, baviće se svim nedostajućim vrednostima ili odstupnicima, obezbeđujući da su podaci pripremljeni za modeliranje.

Do kraja ovog odeljka, cilj je da se stekne kompletno razumevanje skupa podataka Quora, postavljajući osnovu za naredne korake u analizi i modelovanju.

3.1. Pregled podataka

Učitavanje podataka iz CSV datoteke:

```
Quora = pd.read_csv('/kaggle/input/quora-question-
pairs/train.csv.zip')
```

Provera nedostajućih vrednosti za svaku kolonu DataFrame-a "Quora" korišćenjem metode isna() i potom sabiranje broja nedostajućih vrednosti u svakoj koloni pomoću sum() funkcije, pružajući informaciju o tome koliko nedostajućih vrednosti postoji u svakoj koloni, zatim uklanjanje redova koji sadrže nedostajuće vrednosti.

```
print(Quora.isna().sum())
Quora.dropna(inplace=True)
Quora
```

Ispis izgleda ovako:

```
id              0
qid1            0
qid2            0
question1       1
question2       2
is_duplicate    0
dtype: int64
```

`Tabela sa podacima [404287 redova x 6 kolona]:`

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24} [/math] is divided by 24,23?	0
4	4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
...	...	...	...	...	...	...
404285	404285	433578	379845	How many keywords are there in the Racket programming language of the latest version?	How many keywords are there in PERL Programming Language in the latest version?	0
404286	404286	18840	155606	Do you believe there is life after death?	Is it true that there is life after death?	1
404287	404287	537928	537929	What is one coin?	What's this coin?	0
404288	404288	537930	537931	What is the approx annual cost of living while studying in UIC Chicago, for an Indian student?	I am having little hairfall problem but I want to use hair styling product. Which one should I prefer out of gel, wax and clay?	0

1. Tabela sa podacima

Quora skup podataka se sastoji od **404.289** redova i **6** kolona, pružajući bogat izvor informacija za zadatke obrade prirodnog jezika.

Kratak opis svake kolone:

1. **`id`**: Jedinstveni identifikator za svaki red u skupu podataka.

2. **`qid1`**: Identifikator za prvo pitanje u paru pitanja.

3. **`qid2`**: Identifikator za drugo pitanje u paru pitanja.

4. **`question1`**: Tekst prvog pitanja u paru.

5. **`question2`**: Tekst drugog pitanja u paru.

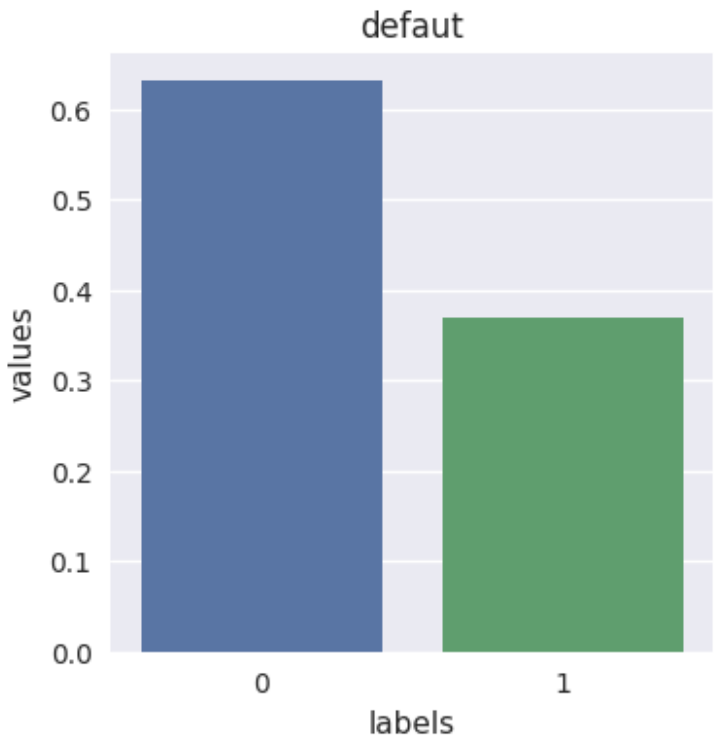
6. **`is_duplicate`**: Binarni indikator (0 ili 1) koji označava da li se dva pitanja u paru smatraju duplikatima (1) ili ne (0).

Ovaj ogroman skup podataka je posebno koristan za zadatke koji se odnose na identifikaciju duplikata pitanja, što ga čini vrednim za istraživanje tehnika obrade prirodnog jezika i modela mašinskog učenja.

3.2. Distribucija podataka

U nastavku će distribuiranost podataka biti grafički prikazana na osnovu broja pitanja koja su duplikati.

```
temp = Quora.is_duplicate.value_counts()
df_class = pd.DataFrame({'labels': temp.index,
                   'values': temp.values/len(Quora)})
plt.figure(figsize = (4,4))
plt.title('defaut')
sns.set_color_codes("pastel")
sns.barplot(x = 'labels', y="values", data=df_class)
locs, labels = plt.xticks()
plt.show()
```



2. Grafik distribuiranosti podataka na osnovu toga da li su pitanja duplikati

Podaci su dobro izbalansirani, sa procentom od **64**% pitanja koja se ne dupliraju i **36**% duplikata.

3.2. Analiza teksta

3.2.1. Word Cloud

Pre nego što se obavi analiza, na neka pitanja su potrebni odgovori:

1. Koje su reči koje se najčešće pojavljuju u svakom pitanju? Da li se razlikuju u odnosu na promenljivu `is_duplicate`?
2. Da li postoje specifični termini koji se ističu u duplim oblacima reči?
3. Da li postoje razlike u istaknutim terminima između dupliranih i neduplikanih oblaka?

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

    # Funkcija za generisanje i iscrtavanje oblaka reči
def generate_wordcloud(data_q1_duplicate, data_q1_non_duplicate,
data_q2_duplicate, data_q2_non_duplicate):
    # Generisanje oblaka reči za pitanje 1 i pitanje 2
    wordcloud_q1_duplicate = WordCloud(width=400, height=200,
background_color='white').generate('
'.join(data_q1_duplicate.astype(str)))
    wordcloud_q1_non_duplicate = WordCloud(width=400, height=200,
background_color='white').generate('
'.join(data_q1_non_duplicate.astype(str)))
    wordcloud_q2_duplicate = WordCloud(width=400, height=200,
background_color='white').generate('
'.join(data_q2_duplicate.astype(str)))
    wordcloud_q2_non_duplicate = WordCloud(width=400, height=200,
background_color='white').generate('
'.join(data_q2_non_duplicate.astype(str)))

    # Crtanje oblaka reči u 2x2 plotu
    plt.figure(figsize=(15, 10))

    # Crtanje pitanja 1 gde je is_duplicate = 1
    plt.subplot(2, 2, 1)
    plt.imshow(wordcloud_q1_duplicate, interpolation='bilinear')
    plt.title('Question1 - Duplicate')
    plt.axis('off')

    # Crtanje pitanja 1 gde je is_duplicate = 0
    plt.subplot(2, 2, 2)
    plt.imshow(wordcloud_q1_non_duplicate, interpolation='bilinear')
    plt.title('Question1 - Non-duplicate')
    plt.axis('off')

    # Crtanje pitanja 2 gde je is_duplicate = 1
    plt.subplot(2, 2, 3)
    plt.imshow(wordcloud_q2_duplicate, interpolation='bilinear')
    plt.title('Question2 - Duplicate')
    plt.axis('off')

    # Crtanje pitanja 2 gde je is_duplicate = 0
    plt.subplot(2, 2, 4)
    plt.imshow(wordcloud_q2_non_duplicate, interpolation='bilinear')
    plt.title('Question2 - Non-duplicate')
    plt.axis('off')

    plt.tight_layout()
    plt.show()
```

```
# Odvojiti podatke za duplirana i neduplirana pitanja
duplicate_data_q1 = Quora[Quora['is_duplicate'] == 1]['question1']
non_duplicate_data_q1 = Quora[Quora['is_duplicate'] ==
0]['question1']
duplicate_data_q2 = Quora[Quora['is_duplicate'] == 1]['question2']
non_duplicate_data_q2 = Quora[Quora['is_duplicate'] ==
0]['question2']

# Generisanje i crtanja oblaka reči za 'question1' (pitanje 1)
i'question2' (pitanje 2)
generate_wordcloud(duplicate_data_q1, non_duplicate_data_q1,
duplicate_data_q2, non_duplicate_data_q2)
```

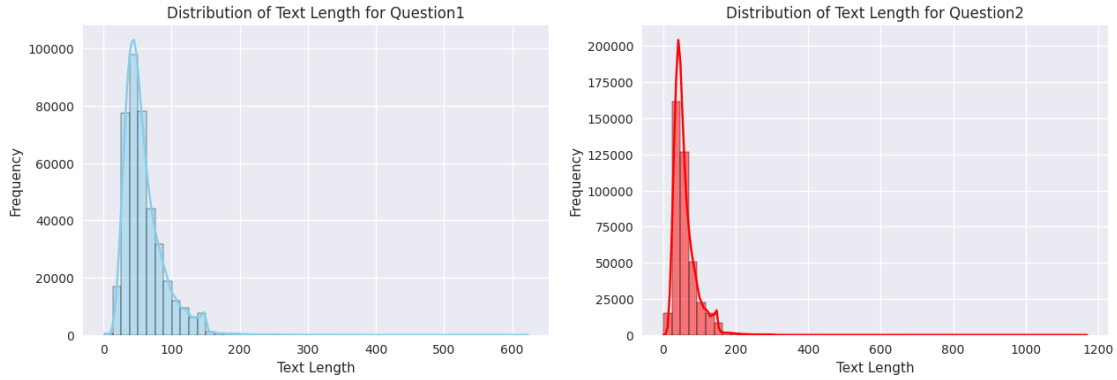
3.1. Oblaci reči

Iz ovih oblaka reči prepoznaje se mnogo sličnosti u rečniku koji se koristi u dva pitanja. Najčešći termini u rečniku sadrže reči: India, Best, Best, Quora, Will, Question , difference , Donald Trump i Life. Žargon ostaje isti pri analizi duplikata i neduplikata (preko linije kolone). Ali iz najčešće korišćenih reči otkriveno je da **tekst deluje čisto i da mu nije potrebna obrada** (iako je manje neuredan, što nije problem jer je učestalost neurednih reči manja).

3.2.2. Distribucija dužine teksta

```
# Distribucija dužine teksta za question1
plt.figure(figsize=(15 ,10))
plt.subplot(2,2,1)
sns.histplot(data=Quora,
x=Quora['question1'].astype('str').apply(len), bins=50, kde=True,
color='skyblue')
plt.title('Distribution of Text Length for Question1')
plt.xlabel('Text Length')
```

```
plt.ylabel('Frequency')
# Text length distribution for question2
plt.subplot(2,2,2)
sns.histplot(data=Quora,
x=Quora['question2'].astype('str').apply(len), bins=50, kde=True,
color='red')
plt.title('Distribution of Text Length for Question2')
plt.xlabel('Text Length')
plt.ylabel('Frequency')
plt.show()
```

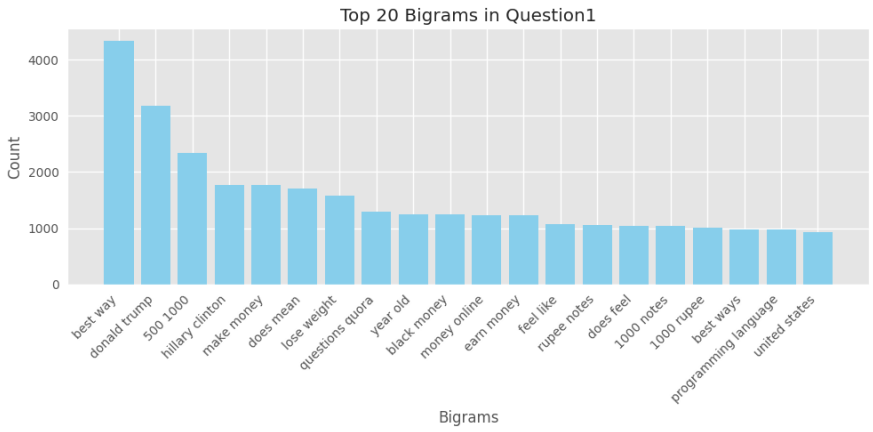


3. 2. Grafički prikaz distribucije dužine teksta za question1 i za question2

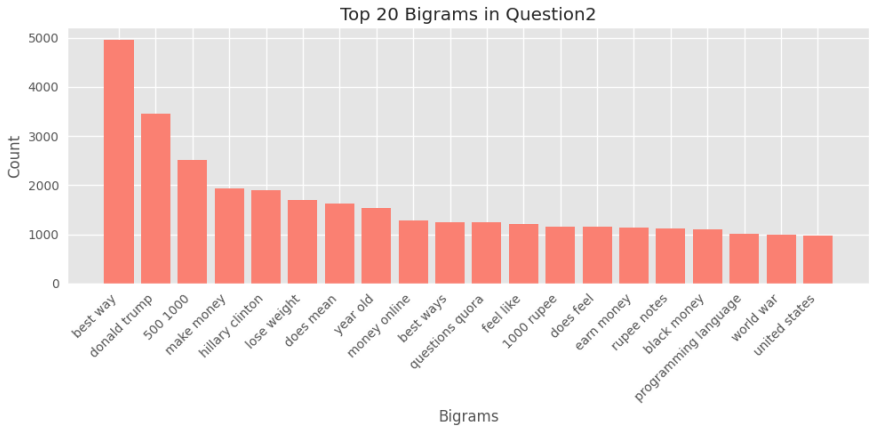
Prilikom ispitivanja KDE dijagrama za dužine question1 i question2, pojavljuje se upadljiva sličnost u distribuciji, pri čemu se obe kreću od 0 do 150 i 0 do 180, respektivno. Primetno je da oba grafika pokazuju jasan vrh unutar prva dva pravougaonika, što ukazuje na učestalost pitanja u relativno kraćem opsegu dužine. Ova tesna usklađenost u distribuciji sugeriše stepen uniformnosti u dužini pitanja, sa tendencijom ka konciznim upitima.

```
from sklearn.feature_extraction.text import CountVectorizer

# Kreiranje CountVectorizer za bigrame
vectorizer = CountVectorizer(ngram_range=(2, 2),
stop_words='english')
bigrams_q1 = vectorizer.fit_transform(Quora['question1'])

# Dohvatanje imena atributa (bigrami)
feature_names = vectorizer.get_feature_names_out()

top_bigrams = pd.DataFrame(bigrams_q1.sum(axis=0).tolist()[0],
index=feature_names, columns=['Count'])
top_bigrams_q1 = top_bigrams.sort_values(by='Count',
ascending=False).head(20)
bigrams_q2 = vectorizer.fit_transform(Quora['question2'])
feature_names = vectorizer.get_feature_names_out()
top_bigrams = pd.DataFrame(bigrams_q2.sum(axis=0).tolist()[0],
index=feature_names, columns=['Count'])
top_bigrams_q2 = top_bigrams.sort_values(by='Count',
ascending=False).head(20)
```

```
# Stilizovanje crteža
plt.style.use('ggplot')
# Crtanje histograma za najčešće bigrame u okviru question1
plt.figure(figsize=(10,5))
plt.bar(top_bigrams_q1.index, top_bigrams_q1['Count'],
color='skyblue')
plt.title('Top 20 Bigrams in Question1')
plt.xlabel('Bigrams')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
# Crtanje histograma za najčešće bigrame u okviru question2
plt.figure(figsize=(10,5))
plt.bar(top_bigrams_q2.index, top_bigrams_q2['Count'],
color='salmon')
plt.title('Top 20 Bigrams in Question2')
plt.xlabel('Bigrams')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



4. Najčešćih 20 bigrama u okviru question1



5. Najčešćih 20 bigrama u okviru question2

U istraživanju prvih 20 bigrama u question1 i question2, pojavljuje se konzistentnost sa prisustvom istih 5 bigrama koji se pojavljuju u identičnom redosledu. Ova zajednička sekvenca ukazuje na ponavljajuću jezičku strukturu zajedničku za oba skupa pitanja. Naročito, dok su ukupni obrasci bigrama blisko usklađeni, postoje suptilne razlike u frekvencijama. Neki bigrami pokazuju veće stope pojavljivanja, dostižući i do 5000 puta, dok se drugi kreću između 1000 i 5000. Ove varijacije u učestalosti označavaju razlike u izražavanju određenih fraza ili koncepata. Ovaj nalaz ne samo da naglašava visok stepen tekstualne sličnosti, već takođe pruža vredan uvid u ponavljajuće jezičke obrasce unutar skupa podataka.

3.3. Analiza korelacije

```
from nltk.metrics import jaccard_distance
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

# Pretpostavka da su'question1' and 'question2' kolone u okviru
DataFrame

# I atribut 'is_duplicate' je kolona koja je indikator da li je
pitanje duplikat ili ne.

# Fitriranje DataFrame da uključuje samo redove gde je is_duplicate
1
duplicate_questions = Quora[Quora['is_duplicate'] ==
1].sample(frac=1/3, random_state=42)


# Kreiranje CountVectorizer
vectorizer = CountVectorizer()
vectorizer.fit(duplicate_questions['question1'] +
duplicate_questions['question2'])

# Vektorizacija pitanja
q1_vectorized =
vectorizer.transform(duplicate_questions['question1'])
q2_vectorized =
vectorizer.transform(duplicate_questions['question2'])

# Konvertovanje u binarni format
q1_binary = np.asarray(q1_vectorized.toarray(),
dtype=bool).astype(int)
q2_binary = np.asarray(q2_vectorized.toarray(),
dtype=bool).astype(int)

# Poravnanje nizova
q1_flat = q1_binary.ravel()
q2_flat = q2_binary.ravel()

# Računanje Jaccard similarity korišćenjem NLTK biblioteke
jaccard_similarity = 1 - jaccard_distance(set(q1_flat),
```

```
set(q2_flat))

print(f"Jaccard Similarity between a fraction of duplicate
questions: {jaccard_similarity}")
```

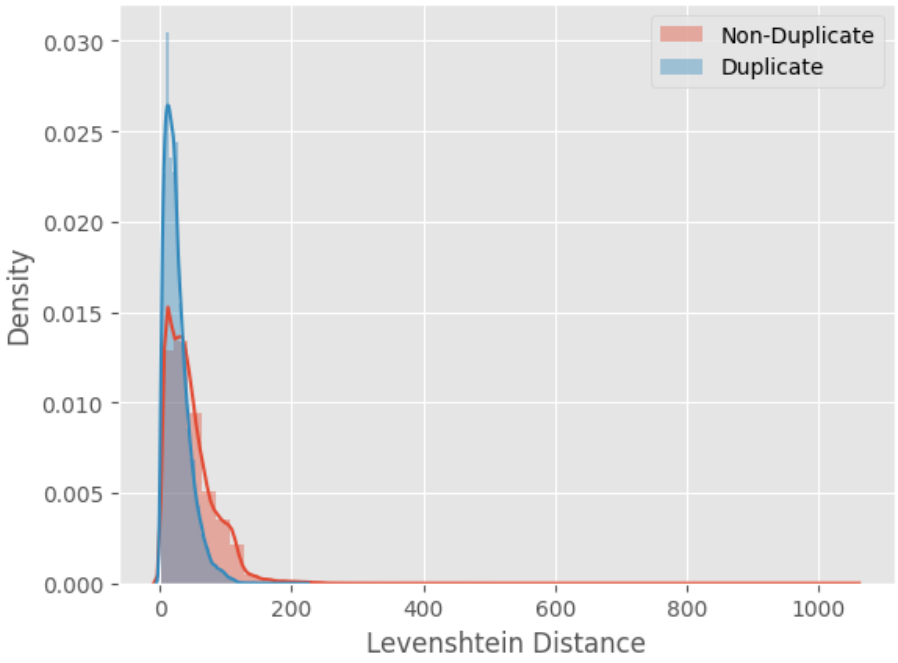
```
Jaccard Similarity between a fraction of duplicate questions: 1.0
```

#Jaccard Similarity score =1 pokazuje da se isti vokabular koristi u oba pitanja .

```
vectorizer = CountVectorizer()
duplicate_questions = Quora[Quora['is_duplicate'] ==
1].sample(frac=0.5)
vectorizer.fit(duplicate_questions['question1'] +
duplicate_questions['question2'])

# Vektorizacija svih pitanja
all_questions_vectorized =
vectorizer.transform(duplicate_questions['question1'] +
duplicate_questions['question2']).toarray().astype(bool)

# Resetovanje index-a za odgovarajuće poravnanje
duplicate_questions = duplicate_questions.reset_index(drop=True)
all_questions_vectorized_df = pd.DataFrame(all_questions_vectorized)

jaccard_similarities = []
levenshtein_distances = []

for i in range(len(duplicate_questions)):
    row_vectorized =
vectorizer.transform([duplicate_questions['question1'][i] +
duplicate_questions['question2'][i]]).toarray().astype(bool).ravel()

    # Računanje Levenshtein distance (Levenštajnovo rastojanje)
    levenshtein_distance =
edit_distance(duplicate_questions['question1'][i],
duplicate_questions['question2'][i])
    levenshtein_distances.append(levenshtein_distance)

  # Dodavanje Jaccard similarity-ja i Levenštajnovih rastojanja
DataFrame-u
duplicate_questions['levenshtein_distance'] = levenshtein_distances

duplicate_questions['levenshtein_distance'] = levenshtein_distances

from sklearn.metrics import jaccard_score
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

duplicate_questions = Quora[Quora['is_duplicate'] == 1]
# Kreiranje CountVectorizer
vectorizer = CountVectorizer()
vectorizer.fit(duplicate_questions['question1'] +
duplicate_questions['question2'])
```

```
# Vektorizacija svih pitanja
all_questions_vectorized =
vectorizer.transform(duplicate_questions['question1'] +
duplicate_questions['question2']).toarray().astype(bool)

# Resetovanje index-a za odgovarajuće poravnanje
duplicate_questions = duplicate_questions.reset_index(drop=True)
all_questions_vectorized_df = pd.DataFrame(all_questions_vectorized)

# Računanje Jaccard similarity između redova korišćenjem scikit-
learn
jaccard_similarities = []

for i in range(len(duplicate_questions)):
    row_vectorized =
vectorizer.transform([duplicate_questions['question1'][i] +
duplicate_questions['question2'][i]]).toarray().astype(bool).ravel()

    # Računanje Jaccard similarity
    jaccard_similarity = jaccard_score(row_vectorized,
all_questions_vectorized_df.loc[i], average='binary')
    jaccard_similarities.append(jaccard_similarity)

# Dodavanje Jaccard similarity-ja DataFrame-u
duplicate_questions['jaccard_similarity'] = jaccard_similarities

vectorizer = CountVectorizer()
non_duplicate_questions = Quora[Quora['is_duplicate'] ==
0].sample(frac=0.5,random_state=1)
vectorizer.fit(non_duplicate_questions['question1'] +
non_duplicate_questions['question2'])

# Vektorizacija svih pitanja
all_questions_vectorized =
vectorizer.transform(non_duplicate_questions['question1'] +
non_duplicate_questions['question2']).toarray().astype(bool)

# Resetovanje index-a za odgovarajuće poravnanje
non_duplicate_questions =
non_duplicate_questions.reset_index(drop=True)
all_questions_vectorized_df = pd.DataFrame(all_questions_vectorized)

jaccard_similarities = []
levenshtein_distances = []

for i in range(len(non_duplicate_questions)):
    row_vectorized =
vectorizer.transform([non_duplicate_questions['question1'][i] +
non_duplicate_questions['question2'][i]]).toarray().astype(bool).rav
el()
```

```
# Računanje Levenshtein distance (Levenštajnovo rastojanje)
    levenshtein_distance =
edit_distance(non_duplicate_questions['question1'][i],
non_duplicate_questions['question2'][i])
    levenshtein_distances.append(levenshtein_distance)

# Dodavanje Jaccard similarities i Levenshtein distances DataFrame-u
non_duplicate_questions['levenshtein_distance'] =
levenshtein_distances

df_non_duplicate =
non_duplicate_questions.groupby('is_duplicate')['levenshtein_distanc
e'].agg(['min', 'max', 'median', 'mean', 'std'])
df_duplicate =
duplicate_questions.groupby('is_duplicate')['levenshtein_distance'].
agg(['min', 'max', 'median', 'mean', 'std'])

# Kombinovanje rezultata
df_combined = df_non_duplicate.add(df_duplicate, fill_value=0)
df_combined
sns.distplot(non_duplicate_questions['levenshtein_distance'],
bins=50, label='Non-Duplicate')
sns.distplot(duplicate_questions['levenshtein_distance'], bins=50,
label='Duplicate')

plt.xlabel('Levenshtein Distance')
plt.ylabel('Density')  # Verovatnoca
plt.legend()

plt.show()
```



6. Grafik verovatnoće dupliranosti pitanja koristeći Levenštajnovo rastojanje

Rezime statistike za Levenštajnovo rastojanje između pitanja koja se ne dupliraju i duplikata je sledeća. Za pitanja bez duplikata, Levenštajnova distanca se kreće od **1** do **1051**, sa medijanom **38**, srednjom vrednošću od približno **46,46** i standardnom devijacijom od oko **35,98**. S druge strane, za dupla pitanja, Levenštajnova distanca se kreće od **1** do **258**, sa medijanom od **22**, srednjom vrednošću od oko **26,28** i standardnom devijacijom od približno **19,60**. Ove statistike pružaju uvid u distribuciju i centralnu tendenciju Levenštajnovih udaljenosti u obe kategorije. Primetno je da Levenštajnove udaljenosti za pitanja koja se ne dupliraju imaju tendenciju da imaju veću varijabilnost u poređenju sa dupliranim pitanjima, što pokazuje veća standardna devijacija.

4. Pristup mašinskog učenja

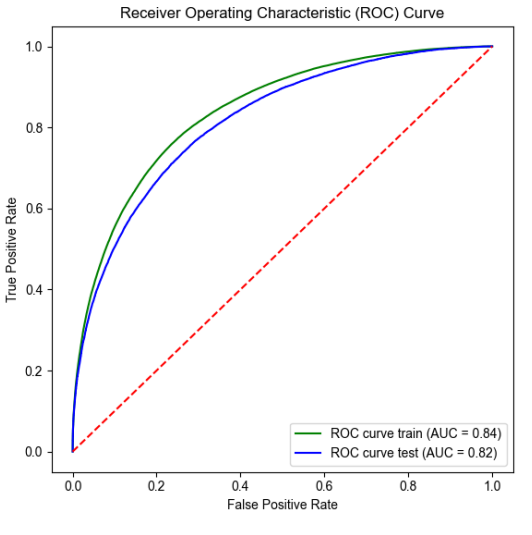
4.1. Logistička regresija

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

X = Quora[['question1', 'question2']]
y = Quora['is_duplicate']

# Kombinovanje 'question1' and 'question2' u jedinstven string za
svaki red

X['combined'] = X['question1'].astype(str) + ' ' +
X['question2'].astype(str)

# Za vrednosti koje ne postoje ubacuje se prazan string

X['combined'] = X['combined'].fillna('')

# Podela podataka za train funkciju
X_train, X_test, y_train, y_test = train_test_split(X['combined'],
y, test_size=0.2, random_state=42)

# TF-IDF vektorizacija (fit transform)
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Uveravanje da se broj primeraka poklapa
print(X_train_tfidf.shape, '  ', len(X_train))

# Logisticka regresija
classifier = LogisticRegression()
classifier.fit(X_train_tfidf, y_train)

# Predikcije
predictions = classifier.predict(X_test_tfidf)

# Evaluacija
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
print(confusion_matrix(y_test, predictions))
```

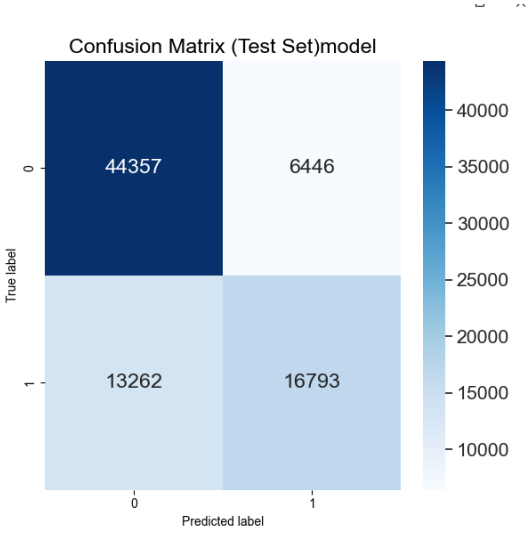
```
(323432, 78154)    323432
Accuracy: 0.756264067872072
[[44357  6446]
 [13262 16793]]
```

Prvo, ovaj izazov koristeći pristup mašinskog učenja, konkretno logističku regresiju na TF-IDF vektorizovanim podacima. Model logističke regresije pokazao je tačnost od **75,62**% na test setu, sa **16.793** istinitih pozitivnih

rezultata, **6.446** lažno pozitivnih, **44.357** pravih negativnih i **13.262** lažno negativnih. Dok tačnost pruža opštu meru tačnosti, druge metrike kao što su preciznost (**72,2**%) i pamćenje (**55,8**%) nude nijansiraniju procenu, posebno u neuravnoteženim skupovima podataka. Razmatranje ovih metrika i dalje istraživanje naprednih modela ili podešavanje hiperparametara može poboljšati performanse modela.



7. ROC kriva



8. Konfuziona matrica

4.2. XGBoost klasifikacija

```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix

X = Quora[['question1', 'question2']]
y = Quora['is_duplicate']

X['combined'] = X['question1'].astype(str) + ' ' +
X['question2'].astype(str)

X['combined'] = X['combined'].fillna('')
X_train, X_test, y_train, y_test = train_test_split(X['combined'],
y, test_size=0.2, random_state=42)


vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# XGBoost klasifikator
classifier = XGBClassifier(verbose=1)

# Fit modela
classifier.fit(X_train_tfidf, y_train)
```

```
# Predikcije nad test skupom podataka
predictions = classifier.predict(X_test_tfidf)

# Evaluacija
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
print(confusion_matrix(y_test, predictions))
```

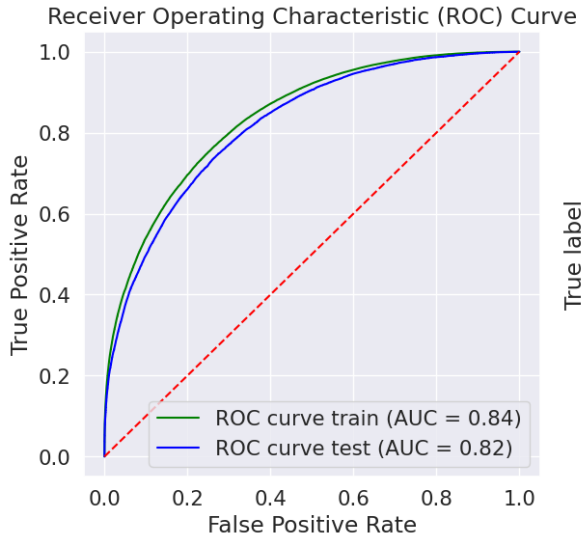
```
Accuracy: 0.7529248806549754
[[45405  5621]
 [14357 15475]]
```

U ovoj XGBoost evaluaciji, model je postigao tačnost od približno **75,3**%. Matrica konfuzije otkriva da je od ukupnog broja instanci u skupu testova, **45.405** tačno klasifikovano kao negativna klasa, dok je **15.475** tačno klasifikovano kao pozitivna klasa. Međutim, bilo je **5.621** slučajeva negativne klase pogrešno klasifikovane kao pozitivne i **14.357** slučajeva pozitivne klase pogrešno klasifikovane kao negativne.

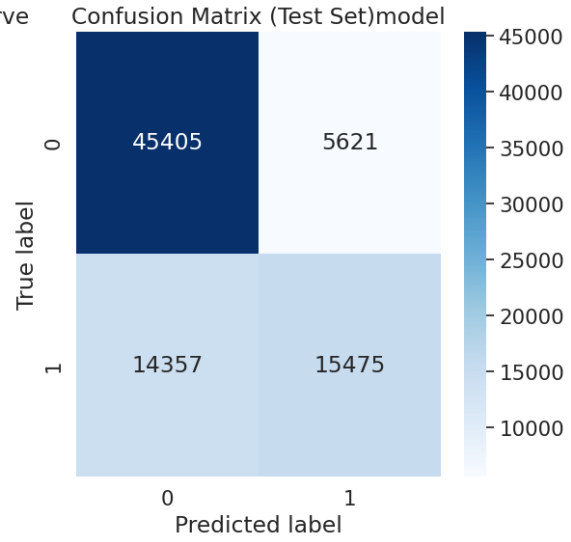
Dakle, otriveno je da statistički modeli mašinskog učenja ne uspevaju da otkriju pedeset procenata dupliranih pitanja, zato su na redu napredniji modeli sa pristupom dubokog učenja.

```
def roc_curve__confusion_matrix(model):
    from sklearn.metrics import roc_curve, auc
    y_prob_test = model.predict_proba(X_test_tfidf)[:, 1]
    y_prob_train  = model.predict_proba(X_train_tfidf)[:, 1]
    prediction=model.predict(X_test_tfidf)

    fpr_test , tpr_test , thresholds = roc_curve(y_test,
y_prob_test)
    fpr_train , tpr_train , thresholds1 = roc_curve(y_train,
y_prob_train)

    roc_auc_test  = auc(fpr_test, tpr_test)
    roc_auc_train  = auc(fpr_train ,tpr_train)


    # Racunanje matrice konfuzije za test set
    confusion_matrix_test = confusion_matrix(y_test, prediction)

    # Kreiranje subplotova
    fig, axs = plt.subplots(1, 2, figsize=(12, 6))

    # Crtanje ROC krivih na prvom subplotu
    axs[0].plot(fpr_train, tpr_train, color='green', label='ROC
curve train (AUC = %0.2f)' % roc_auc_train)
    axs[0].plot(fpr_test, tpr_test, color='blue', label='ROC curve
test (AUC = %0.2f)' % roc_auc_test)
    axs[0].plot([0, 1], [0, 1], color='red', linestyle='--')
    axs[0].set_xlabel('False Positive Rate')
    axs[0].set_ylabel('True Positive Rate')
```

```
    axs[0].set_title('Receiver Operating Characteristic (ROC)
Curve')
    axs[0].legend(loc="lower right")

    sns.set(font_scale=1.4)
    sns.heatmap(confusion_matrix_test, annot=True, fmt='g',
cmap='Blues', ax=axs[1])
    axs[1].set_xlabel('Predicted label')
    axs[1].set_ylabel('True label')
    axs[1].set_title('Confusion Matrix (Test Set)model');

    plt.tight_layout()
    plt.show()

roc_curve__confusion_matrix(classifier)
```



9. ROC kriva



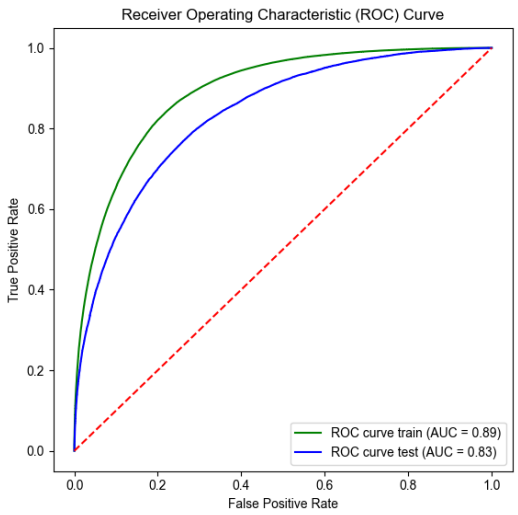
10. Konfuziona matrica

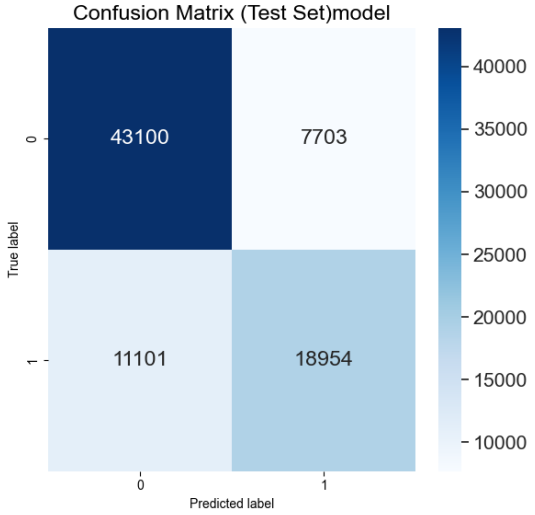
4.3. SVM (Support vector machine) model

```
Quora = pd.read_csv('./kaggle/input/quora-question-
pairs/train.csv.zip')
X = Quora[['question1', 'question2']]
y = Quora['is_duplicate']

X['combined'] = X['question1'].astype(str) + ' ' +
X['question2'].astype(str)

X['combined'] = X['combined'].fillna('')
X_train, X_test, y_train, y_test = train_test_split(X['combined'],
y, test_size=0.2, random_state=42)

vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# SVM model sa linearnim jezerom
```

```
svm_model = LinearSVC()


# Fit modela
svm_model.fit(X_train_tfidf, y_train)

# Predikcije nad test skupom podataka
predictions_svm = svm_model.predict(X_test_tfidf)

# Evaluacija
accuracy_svm = accuracy_score(y_test, predictions_svm)
print(f"Accuracy (SVM): {accuracy_svm}")
print(confusion_matrix(y_test, predictions_svm))

(323432, 78154)    323432
Accuracy: 0.756264067872072
[[44357  6446]
 [13262 16793]]
```

U ovoj evaluaciji SVM modela, postignuta je tačnost od približno **75,6**%. Analiza matrice konfuzije otkriva da je od ukupnog broja instanci u test skupu, **44.357**





sposobnosti prepoznavanja dupliranih pitanja. Prilikom prelaska na dublje modele, važno je pažljivo odabrati i pravilno podešavati arhitekturu modela kako bi se postigli optimalni rezultati.

11. ROC kriva

12. Konfuziona matrica

5. Pristup dubokog učenja

Obrada prirodnog jezika (NLP) je bila svedok transformativnog napretka, zahvaljujući integraciji neuralnih mreža (NN). Koristeći mogućnosti dubokog učenja, NN se pokazao kao ključni faktor u razumevanju i izdvajanju značenja iz ogromne količine tekstualnih podataka. U ovom odeljku ulazimo u prednosti koje NN donosi zadacima NLP-a:

Prednosti korišćenja neuralnih mreža u NLP:

- **Semantičko Razumevanje:** NN se ističe u razmevanju zamršenih semantičkih odnosa unutar teksta, omogućavajući modelima da shvate kontekst, nijanse i suptilnosti u jeziku.
- **Učenje Karakteristika:** Učenje hijerarhijskog predstavljanja u NN omogućava modelima da automatski otkriju relevantne karakteristike iz sirovog teksta, smanjujući potrebu za ručnim inženjeringom karakteristika.
- **Analiza Konteksta:** Za razliku od tradicionalnih metoda, NN modeli mogu uzeti u obzir ceo kontekst rečenice ili dokumenta, nudeći poboljšanu kontekstualnu analizu ključnu za razumevanje jezičkih zamršenosti.

Biće istraženi različiti pristupi zasnovani na NN prilagođenim zadacima NLP-a, od kojih svaki nudi jedinstvene prednosti. Sve počinje unapred obučenim ugrađivanjem reči, osnovnom tehnikom koja obuhvata semantičke odnose između reči. Zatim će biti reči o Sijamskim mrežama, specijalizovanoj arhitekturži dizajniranoj za upoređivanje i razumevanje odnosa u tekstualnim podacima. Y

5.1.Pretrained Word Embeddings (Unapred obučena ugradnja reči)

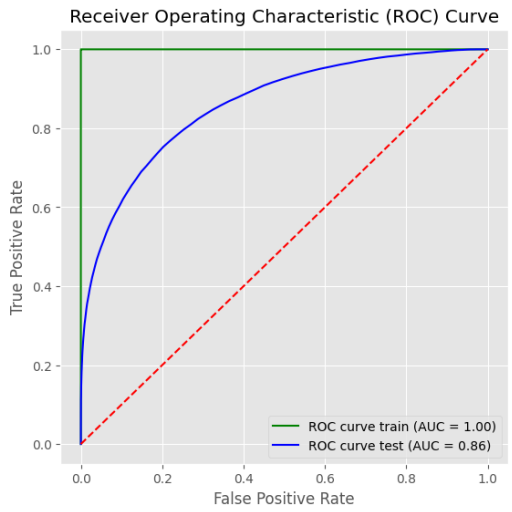
Kao osnova za NLP, biće generisane i iskorišćene unapred obučene ugradnje reči koristeći tehnike kao što su Word2Vec i GloVe.

```
from gensim.models import Word2Vec
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import numpy as np

X = Quora[['question1', 'question2']]
y = Quora['is_duplicate']

# Tokenizovanje rečenica
tokenized_sentences_q1 = X['question1'].apply(lambda x:
str(x).split())
tokenized_sentences_q2 = X['question2'].apply(lambda x:
str(x).split())

# Treniranje Word2Vec modela
word2vec_model_q1 = Word2Vec(sentences=tokenized_sentences_q1,
```

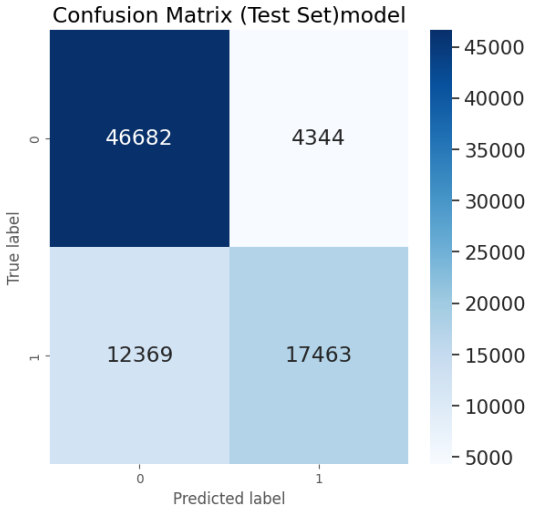
```
vector_size=100, window=5, min_count=1, workers=4)
word2vec_model_q2 = Word2Vec(sentences=tokenized_sentences_q2,
vector_size=100, window=5, min_count=1, workers=4)

# Funkcija za dobijanje vektorske predstave rečenice
def get_sentence_vector(sentence, model):
    vector = np.zeros(model.vector_size)
    count = 0
    for word in sentence:
        if word in model.wv:
            vector += model.wv[word]
            count += 1
    if count != 0:
        vector /= count
    return vector

# Kreiranje vektora karakteristika za svako pitanje
X_q1 = np.array([get_sentence_vector(sentence, word2vec_model_q1)
for sentence in tokenized_sentences_q1])
X_q2 = np.array([get_sentence_vector(sentence, word2vec_model_q2)
for sentence in tokenized_sentences_q2])

# Konkatenacija vektora karakteristika
X_combined = np.concatenate((X_q1, X_q2), axis=1)

# Raspodela podataka
X_train, X_test, y_train, y_test = train_test_split(X_combined, y,
test_size=0.2, random_state=42)

# Random Forest bez sredjivanja
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

# Predikcije
y_pred = rfc.predict(X_test)
# Evaluacija
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,
y_pred))
```

```
Accuracy: 0.7933043112617181
Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.91      0.85     51026
           1       0.80      0.59      0.68     29832

    accuracy                           0.79     80858
   macro avg       0.80      0.75      0.76     80858
weighted avg       0.79      0.79      0.78     80858
```

```
roc_curve__confusion_matrix(rfc)
```



13. ROC kriva



14. Konfuziona matrica

U ovoj evaluaciji korišćenjem modela Random Forest sa unapred obučenim Word2Vec ugrađenim elementima, ukupna postignuta preciznost bila je približno 79,3%. Izveštaj o klasifikaciji pruža detaljan pregled performansi modela za svaku klasu:

<u>Klasa 0 (pitanja bez duplikata):</u>

Preciznost: **79**%, Recall: **91**% , F1 score : **85**%

<u>Klasa 1 (Duplirana pitanja):</u>

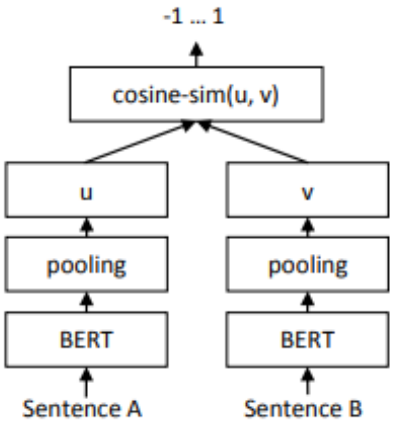
Preciznost: **80**%, Recall: **59**%, F1 score: **68**%

Prosečan F1-skor je oko **76**%, što ukazuje na uravnotežene performanse u svim klasama. Ponderisani prosečni F1 rezultat, s obzirom na neravnotežu u klasi, je približno **78**%. Iz ROC krive se čini da model preopterećuje podatke o vozu sa auc na train funkciji **1** i testu **0.86**.

Ali u poređenju sa XGBoost modelom na tfidf podacima, Word2Vec daje bolje rezultate.Ovi rezultati sugerišu da model ima dobre rezultate u identifikaciji pitanja koja se ne dupliraju, sa visokom preciznošću i pamćenjem. Međutim, suočava se sa izazovima u pravilnom klasifikovanju duplih pitanja, posebno u smislu *recall*-a. Dalja analiza i potencijalno fino podešavanje modela mogli bi da poboljšaju njegovu sposobnost da preciznije identifikuje duple instance (ali ovo bi moglo da potraje mnogo vremena) tako da će se koristiti drugi pristup dubokog učenja.

5.2.Sijamske mreže

Sijamske mreže su specijalizovana arhitektura neuralne mreže dizajnirana za poređenje i snimanje odnosa između parova ulaznih podataka. Biće pokazano kako primeniti sijamsku mrežu za analizu sličnosti teksta, omogućavajući merenje bliskosti između dva ulazna pitanja.



15. Sijamske mreže

```
import tensorflow as tf
from keras.models import Model
from keras.layers import Input, Embedding, LSTM, Lambda, Dense
from keras import backend as K
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

questions1 = Quora['question1']
questions2 = Quora['question2']
labels = Quora['is_duplicate']

questions1 = questions1.fillna('')
questions2 = questions2.fillna('')


max_sequence_length = 80
embedding_dim = 300
questions = Quora['question1'].astype(str) + ' ' +
Quora['question2'].astype(str)

tokens = [word for sentence in questions for word in
sentence.split()]

# Računanje dužine vokabulara
vocabulary_size = len(set(tokens))
```

```
questions = (questions1 + ' ' + questions2).astype(str)
tokenizer = Tokenizer(num_words=vocabulary_size)
tokenizer.fit_on_texts(questions)

sequences1 = tokenizer.texts_to_sequences(questions1)
sequences2 = tokenizer.texts_to_sequences(questions2)
padded_sequences1 = pad_sequences(sequences1,
maxlen=max_sequence_length)
padded_sequences2 = pad_sequences(sequences2,
maxlen=max_sequence_length)

input_layer1 = Input(shape=(max_sequence_length,))
input_layer2 = Input(shape=(max_sequence_length,))

embedding_layer = Embedding(input_dim=vocabulary_size,
output_dim=embedding_dim)

lstm_layer = LSTM(units=50)

x1 = embedding_layer(input_layer1)
x1 = lstm_layer(x1)

x2 = embedding_layer(input_layer2)
x2 = lstm_layer(x2)

distance_layer =  Lambda(lambda x: tf.keras.backend.abs(x[0] -
x[1]),
output_shape=lambda _: (1,))([x1, x2])

output_layer = Dense(units=1, activation='sigmoid')(distance_layer)

siamese_model = Model(inputs=[input_layer1, input_layer2],
outputs=output_layer)

siamese_model.compile(optimizer=Adam(0.0001),
loss='binary_crossentropy', metrics=['accuracy'])
callbacks = [
    EarlyStopping(monitor='val_loss', patience=2,
restore_best_weights=True),
    ModelCheckpoint(filepath='siamese_model_weights.h5',
save_best_only=True)
]

siamese_model.fit([padded_sequences1, padded_sequences2], labels,
epochs=5, batch_size=32, validation_split=0.2, callbacks=callbacks)


Epoch 1/5
10108/10108 [==============================] - 240s 23ms/step -
loss: 0.5587 - accuracy: 0.6945 - val_loss: 0.4899 - val_accuracy:
0.7654
Epoch 2/5
10108/10108 [==============================] - 197s 20ms/step -
```

```
loss: 0.4511 - accuracy: 0.7904 - val_loss: 0.4518 - val_accuracy:
0.7921
Epoch 3/5
10108/10108 [==============================] - 195s 19ms/step -
loss: 0.3914 - accuracy: 0.8254 - val_loss: 0.4308 - val_accuracy:
0.8038
Epoch 4/5
10108/10108 [==============================] - 194s 19ms/step -
loss: 0.3432 - accuracy: 0.8508 - val_loss: 0.4207 - val_accuracy:
0.8107
Epoch 5/5
10108/10108 [==============================] - 194s 19ms/step -
loss: 0.3008 - accuracy: 0.8725 - val_loss: 0.4201 - val_accuracy:
0.8136
```

Istorija obuke za sijamski LSTM model otkriva napredak u tačnosti tokom pet epoha. Počevši sa početnom tačnošću od **69,45**%, model je podvrgnut značajnom poboljšanju, što je kulminiralo primetno tačnošću od **87,25**% do poslednje epohe. Ovaj trend naglašava sposobnost modela da uoči semantičke odnose u parovima teksta. Tačnost validacije, koja odražava uzlaznu putanju, počinje od **76,54**% i konvergira na **81,36**%, potvrđujući robusnost sijamske LSTM arhitekture u hvatanju nijansiranih obrazaca unutar tekstualnih podataka. Smanjenje vrednosti gubitaka dodatno potkrepljuje sposobnost modela da optimizuje svoje prediktivne sposobnosti tokom procesa obuke. Ovi rezultati potvrđuju efikasnost sijamskog LSTM modela u predviđanju sličnosti teksta.

6. Zaključak

U okviru ovog projektnog zadatka, obrađena su dva pristupa: pristup mašinskog učenja i pristup dubokog učenja.

Predstavnici za mašinsko učenje bili su Logistička regresija, XGBoost klasifikacija, kao i SVM model sa zadovoljavajućim procentom tačnosti, ali i dalje gorim u odnosu na predstavnike dubokog učenja.

Što se tiče dubokog učenja, koje se pokazalo kao efektnije, poređeni su Sijamski LSTM model i Random Forest Model.

S druge strane, sijamski LSTM model koristi mreže dugotrajne kratkoročne memorije (LSTM) da bi uhvatio sekvencijalne zavisnosti u tekstu. Istorija obuke/treniranja pokazuje stalna poboljšanja. Komparativno, oba modela pokazuju pohvalne performanse, pri čemu sijamske mreže pokazuju nešto veću preciznost.

''Duboke mreže'' su pokazale visoku tačnost u mnogim zadacima obrade teksta, kao što su klasifikacija sentimenta, prepoznavanje entiteta ili mašinsko prevođenje. Međutim, važno je napomenuti da, iako duboko učenje može biti efikasno u obradi teksta, to ne znači nužno da je bolje u svim situacijama. Neki jednostavniji zadaci mogu biti rešeni klasičnim metodama mašinskog učenja bez potrebe za kompleksnim dubokim modelima.

7. Literatura

[1] Ian H. Witten, Eibe Frank, Mark A. Hall, Data Mining Practical Machine Learning Tools and Techniques, 2011.

[2] Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, 2019.

[3] Rodolfo Bonnin, Building Machine Learning Projects with TensorFlow, 2016.

[4] Sebastian Raschka, Vahid Mirjalili, "Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow" (2017)

[5] Pedro Domingos, "The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World" (2015)