SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij

# KLASIFIKACIJA SENTIMENTA FILMSKIH RECENZIJA

Projektni zadatak iz kolegija

*Raspoznavanje uzoraka i strojno učenje*

Luka Pivk

Osijek, 2024.

# Sadržaj

# 1. UVOD

Jedan od najpopularnijih primjena strojnog učenja je klasifikacija sentimenta. To je proces koji uključuje analizu tekstualnih podataka s ciljem utvrđivanja emocionalnog tona ili stava autora. Posebno često se koristi na društvenim mrežama te na sličnim platformama gdje korisnici imaju priliku slobodno iznositi mišljenja o danim temama. U kontekstu filmskih recenzija, klasifikacija sentimenta automatizira zamoran proces ručnog čitanja i određivanja je li neka recenzija pozitivna, negativna ili neutralna i time štedi vrijeme potencijalnim gledateljima, ali i filmskim studijima.

S tehničke strane, postoji nekoliko pristupa za izradu modela koji može klasificirati sentiment tekstualnih podataka, a mogu se podijeliti u tri kategorije:

- ručno izrađeni leksički sustavi poput rječnika pozitivnih i negativnih riječi
- tradicionalne metode strojnog učenja poput logističke regresije
- napredne metode dubokog učenja, odnosno neuronske mreže.

U ovome radu za klasifikaciju filmskih recenzija koriste se LSTM (*eng. Long Short-Term Memory*) neuronske mreže, koje su posebno učinkovite za obradu sekvencijalnih podataka kao što su tekstovi. One dugoročno mogu zapamtiti zavisnosti unutar podataka zbog čega su idealno rješenje za zadatke gdje je kontekst riječi unutar rečenica ključan za preciznu klasifikaciju.

# 2. EKSPLORATIVNA ANALIZA I PREDOBRADBA PODATAKA

## 2.1. Inicijalni skup podataka

Skup podataka koji će se koristiti u ovome radu preuzet je sa stranice *Kaggle.com,* a na njemu su nakon prikupljanja već odrađeni određeni koraci pripreme podataka za modeliranje koji su objašnjeni u nastavku. Naime, riječ je o *.tsv* (*eng. tab-separated*) datoteci koja sadrži rečenice na engleskom jeziku iz recenzija raznih filmova sa stranice Rotten Tomatoes.

Svaka rečenica je podijeljenja na nekolicinu manjih fraza koji mogu biti dužine samo jedne riječi. Česte i kratke riječi poput veznika i čestica filtrirane su tako da se pojavljuju samo jednom duž skupa. Svaka rečenica ima svoj *SentenceId,* a svaka fraza ima svoj *PhraseId, ali i SentenceId* kako bi se lakše prepoznalo kojoj rečenici fraza pripada. Iako *PhraseId* i *SentenceId* nemaju ulogu u samom treniranju modela, mogu biti korisni pri drugačijim tipovima obrade podataka.

Usitnjavanjem rečenica na kraće fraze, neuronska mreža će lakše prepoznati riječi koje odaju sentiment. Primjerice, opisni pridjevi poput *awesome* ili *terrible* imaju puno veće značenje za sentiment recenzije od veznika ili vlastitih imenica, a bez usitnjavanja lako bi se izgubili unutar dugih rečenica te bi model zbog toga bio neprecizniji.

Skup za treniranje modela za svaku frazu sadrži odgovarajući sentiment koji je označen brojevima od 0 do 4, pri čemu je:

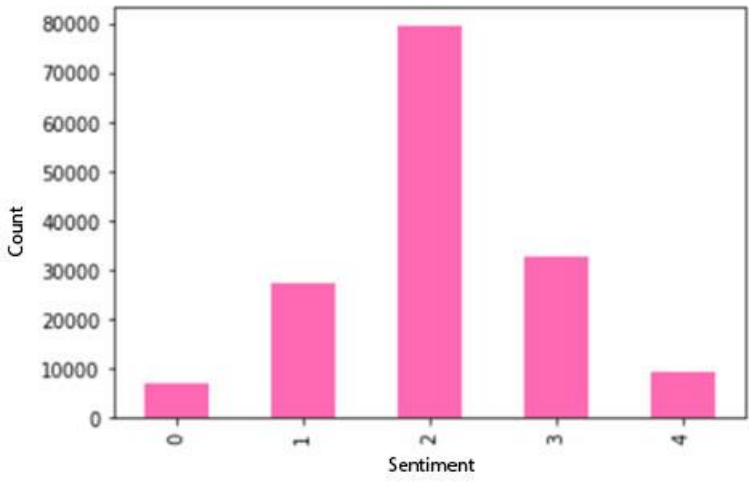
- 0 – negativan sentiment
- 1 – donekle negativan sentiment
- 2 – neutralan sentiment
- 3 – donekle pozitivan sentiment
- 4 – pozitivan sentiment.

| | PhraseId | SentenceId | Phrase | Sentiment |
|---|---|---|---|---|
| 0 | 1 | 1 | A series of escapades demonstrating the adage ... | 1 |
| 1 | 2 | 1 | A series of escapades demonstrating the adage ... | 2 |
| 2 | 3 | 1 | A series | 2 |
| 3 | 4 | 1 | A | 2 |
| 4 | 5 | 1 | series | 2 |
| ... | ... | ... | ... | ... |
| 156055 | 156056 | 8544 | Hearst 's | 2 |
| 156056 | 156057 | 8544 | forced avuncular chortles | 1 |
| 156057 | 156058 | 8544 | avuncular chortles | 3 |
| 156058 | 156059 | 8544 | avuncular | 2 |
| 156059 | 156060 | 8544 | chortles | 2 |

156060 rows × 4 columns

Slika 2.1. Sažeti tablični prikaz inicijalnog skupa za treniranje modela

U skupu je 156059 fraza i svaki redak skupa je popunjen odgovarajućim podacima što znači da nije potrebno dodatno čišćenje skupa. Također, skup je pravilno balansiran s obzirom na sentiment - distribucija nalikuje normalnoj (Slika 2.2.).




Slika 2.2. Distribucija fraza ovisno o pridruženom sentimentu

Skup koji je priložen za testiranje modela (Slika 2.3.) sadrži 66291 fraze kojima treba dodijeliti sentiment, no nije nužno isključivo na njemu testirati model.

| | PhraseId | SentenceId | Phrase |
|---|---|---|---|
| 0 | 156061 | 8545 | An intermittently pleasing but mostly routine ... |
| 1 | 156062 | 8545 | An intermittently pleasing but mostly routine ... |
| 2 | 156063 | 8545 | An |
| 3 | 156064 | 8545 | intermittently pleasing but mostly routine effort |
| 4 | 156065 | 8545 | intermittently pleasing but mostly routine |
| ... | ... | ... | ... |
| 66287 | 222348 | 11855 | A long-winded , predictable scenario . |
| 66288 | 222349 | 11855 | A long-winded , predictable scenario |
| 66289 | 222350 | 11855 | A long-winded , |
| 66290 | 222351 | 11855 | A long-winded |
| 66291 | 222352 | 11855 | predictable scenario |

66292 rows × 3 columns

Slika 2.3. Sažeti tablični prikaz inicijalnog skupa za testiranje modela

## 2.2. Dodatna predobradba

Iako su podaci već prošli jednu razinu predobradbe, potrebno ih je dodatno prilagoditi kako bi bili u optimalnom formatu za uporabu u treniranju modela. Na slici 2.4. prikazana je metoda *clean_sentences* kroz koju će svaki redak u skupu podataka, odnosno svaka fraza, proći. Metoda će iz svake fraze pomoću regularnih izraza maknuti sve znakove koji nisu iz engleske abecede – npr. interpunkcijski znakovi. Onda će odvojiti svaku riječ iz fraze praveći od jednog niza znakova (*eng. stringa*) polje (*eng. array*) *stringova* – tokenizirat će ju. Na kraju, svaku riječ će unutar polja zamijeniti njenom lemom, tj. korijenskom riječju. Primjerice, imenice u množini će pretvoriti u jedninu, glagolske imenice u glagole i sl. Cilj lematizacije je smanjivanje broja različitih riječi kako bi ih model lakše obrađivao, istovremeno zadržavajući njihovu semantičko značenje.

```
def clean_sentences(df):
    reviews = []

    for sentence in tqdm(df['Phrase']):
        if type(sentence) is not str:
            sentence = str(sentence)

        #remove html content
        review_text = BeautifulSoup(sentence, features="html.parser").get_text()

        #remove non-alphabetic characters
        review_text = re.sub("[^a-zA-Z]"," ", review_text)

        #tokenize the sentences
        words = word_tokenize(review_text.lower())

        #lemmatize each word to its lemma
        lemma_words = [lemmatizer.lemmatize(i) for i in words]

        reviews.append(lemma_words)

    return(reviews)
```

Slika 2.4. Metoda *clean_sentences()*

Međutim, još jedan korak je potreban kako bi ovaj skup bio spreman za neuronske mreže - vektorizacija teksta. To je proces koji se koristi za pretvorbu teksta u vektore cijelih brojeva, odnosno format s kojim je računalima puno lakše raditi nego s riječima. Vektorizacija, ovisno o implementaciji, može obuhvaćati neke procese koji su već ranije spomenuti – tokenizaciju i lematizaciju – no u slučaju ovoga rada, vektorizacija je proces koji obuhvaća ova tri vrlo važna koraka:

- izgradnju vokabulara
- kodiranje
- standardizaciju.

U prvome koraku sagradit će se niz svih jedinstvenih riječi iz ranije obrađenog skupa podataka, a svaka jedinstvena riječ postat će jedinstveni cijeli broj. Zatim će se svaka riječ u skupu podataka kodirati, odnosno zamijeniti odgovarajućim cijelim brojem. Na kraju, pronaći će se najduža fraza unutar skupa – ovom skupu najduža fraza je imala 48 riječi – te će sve ostale fraze proširiti na njenu dužinu dodavanjem broja 0 za svaku riječ koja frazi nedostaje da bi imala 48 riječi. Time će svaka fraza biti jednake dužine što čini ovaj skup uniformnim, a nule će model prepoznati kao prazne riječi ili riječi bez značenja pa je zbog toga ova metoda standardizacije idealno rješenje. Tako obrađeni skup sada je spreman za treniranje modela. Na slici 2.5. primjer je takvog skupa. Dok čovjeku nije od velike koristi, idealan je za računalnu obradu.  Na slici 2.6. prikazana jednostavna implementacija vektorizacije koja uključuje sva tri navedena koraka.

```
[[ 186  148   37 ...    0    0    0]
 [ 273  772    0 ...    0    0    0]
 [   2   41   10 ...    0    0    0]
 ...
 [3536    0    0 ...    0    0    0]
 [  10   73   59 ...    0    0    0]
 [  28    6 1633 ...    0    0    0]], shape=(124848, 48)
```

Slika 2.5. Primjer vektoriziranog skupa spremnog za treniranje modela

```
unique_words = set()
len_max = 0

for sentence in X_train:
    unique_words.update(sentence)

    if(len_max<len(sentence)):
        len_max = len(sentence)

print("Unique words count: ", len(list(unique_words)))
print("Max review length: ", len_max)
```

```
Unique words count:  13736
Max review length:  48
```

```
X_train_s = [' '.join(word_list) for word_list in X_train]
X_val_s = [' '.join(word_list) for word_list in X_val]

vectorize_layer = keras.layers.TextVectorization(
    max_tokens=len(list(unique_words))+2,
    pad_to_max_tokens=True,
    output_mode='int',
    output_sequence_length=len_max)

vectorize_layer.adapt(X_train_s)
```

Slika 2.6. Implementacija vektorizacija u *Pythonu*

# 3. IZRADA I TRENIRANJE MODELA

Za izradu modela koji će predviđati sentiment recenzije korištena je Python biblioteka *Keras*. *Keras* je API visoke razine koji pruža veliki broj metoda strojnog učenja, dubokog učenja i drugih srodnih područja. Ključna značajka *Kerasa* je dostupnost. Osim što je napisana u *Python-u* – jednom od najpopularnijih programskih jezika – čak i programer početnik može složiti nekakav jednostavan model i zatim postupno nadograđivati znanje. Najčešća implementacija *Kerasa* su neki od mnogih oblika neuronskih mreža, a jedna od njih – *Long Short-Term Memory* (LSTM) – će se koristiti pri izradi modela u nastavku.

LSTM je tip rekurentne neuronske mreže dizajnirane za rad sa sekvencijalnim podacima kao što su, na primjer nizovi znakova, jer sadrže „skriveno“ stanje koje se prenosi iz koraka u korak. Time rekurentne neuronske mreže mogu „pamtiti“ informacije iz raniji dijelova nizova te tako bolje razumjeti kontekst niza. Ako je niz, primjerice, rečenica, takva neuronska mreža bolje će razumjeti o čemu se govori u njoj.

Na slici 3.1. prikazana je implementacija jedne takve neuronske mreže, a ona će u nastavku biti trenirana. Prije LSTM sloja koristi se *Embedding* sloj koji dodatno obrađuje proslijeđeni predobrađeni skup za treniranje i pretvara ga u vektore veličine 300 brojeva. Poslije LSTM sloja dodaju se *Dense*, *Dropout* te još jedan *Dense sloj*. *Dense* slojevi smanjuju dimenzije vlastitih izlaznih podataka koji će u posljednjem sloju biti vektori veličine od pet decimalnih brojeva između 0 i 1. Svaki broj označavat će vjerojatnost kojoj klasi (sentimentu) ulazni podaci (fraze) pripadaju. Sloj *Dropout* koristi se za sprječavanje *overfittinga*.

```
early_stopping = keras.callbacks.EarlyStopping(min_delta = 0.001, mode = 'max', monitor='val_acc', patience = 2)
callback = [early_stopping]
```

```
model=keras.models.Sequential()
model.add(keras.layers.Embedding(len(list(unique_words))+2,300,input_length=len_max))
model.add(keras.layers.LSTM(128,dropout=0.5, recurrent_dropout=0.5,return_sequences=True))
model.add(keras.layers.LSTM(64,dropout=0.5, recurrent_dropout=0.5,return_sequences=False))
model.add(keras.layers.Dense(100,activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(num_classes,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer=keras.optimizers.Adam(learning_rate=0.005),metrics=['accuracy'])
model.summary()
```

Slika 3.1. Implementacija jednostavne LSTM rekurentne neuronske mreže

Metoda *model.compile()* konfigurira proces učenja prije početka treniranja modela, a prihvaća tri ključna argumenta:

- funkciju gubitka – razliku između predviđenog i stvarnog rezultata koji će model pokušati minimizirati, postoji mnogo načina kojima se razlika može računati te ovisi o problemu kojeg neuronska mreža rješava, a *categorical_crossentropy* se preporuča za metode višeklasne klasifikacije kao što je ovaj

- optimizator – algoritam koji model korisni za minimiziranje funkcije gubitka, prilagođava težine modela kako bi s vremenom smanjio gubitak
- metrike – funkcije koje se koriste za ocjenjivanje performansi modela.

Osim *Dropout* sloja, za sprječavanje *overfittinga* korištena je i *EarlyStopping* metoda koja zaustavlja treniranje modela ako se preciznost modela na validacijskom skupu prestane povećavati barem dvije epohe za redom.

Prije konačnog treniranja modela, skup za treniranje podijeljen je u omjeru 80:20 pri čemu će se 20% skupa koristiti za validaciju modela.

Na slici 3.2. prikazan je tijek treniranja modela. Poželjno je nekoliko puta istrenirati model s različitim zadanim parametrima za broj epoha treniranja i veličinu uzorka (*eng. batch_size*). Nakon treniranja s različitim parametrima, optimalan omjer trajanja treniranja i preciznosti koja se u nijednom slučaju ne penje iznad 72% za trening skup, te 67% za validacijski skup, dobiva se s 10 epoha i veličinom uzorka od 256.

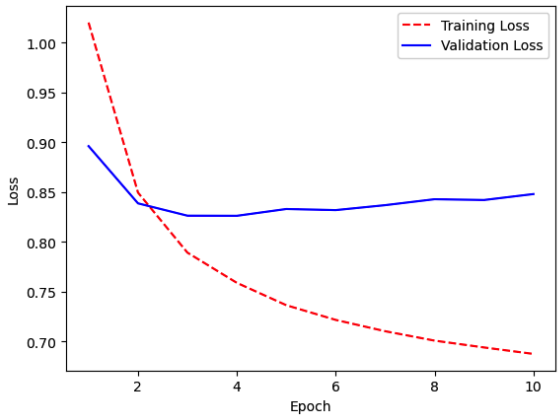
```
X_train_v = vectorize_layer(np.array(X_train_s))
X_val_v = vectorize_layer(np.array(X_val_s))

history=model.fit(X_train_v, y_train, validation_data=(X_val_v, y_val),epochs=10, batch_size=256, verbose=1, callbacks=callback)
```

```
Epoch 1/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 90s 185ms/step - accuracy: 0.5674 - loss: 1.0744 - val_accuracy: 0.6355 - val_loss: 0.8963
Epoch 2/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 94s 194ms/step - accuracy: 0.6595 - loss: 0.8498 - val_accuracy: 0.6511 - val_loss: 0.8388
Epoch 3/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 97s 198ms/step - accuracy: 0.6780 - loss: 0.7835 - val_accuracy: 0.6588 - val_loss: 0.8264
Epoch 4/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 95s 194ms/step - accuracy: 0.6882 - loss: 0.7492 - val_accuracy: 0.6636 - val_loss: 0.8263
Epoch 5/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 96s 197ms/step - accuracy: 0.6976 - loss: 0.7310 - val_accuracy: 0.6613 - val_loss: 0.8331
Epoch 6/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 97s 199ms/step - accuracy: 0.7033 - loss: 0.7134 - val_accuracy: 0.6664 - val_loss: 0.8320
Epoch 7/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 97s 198ms/step - accuracy: 0.7050 - loss: 0.7048 - val_accuracy: 0.6682 - val_loss: 0.8370
Epoch 8/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 99s 203ms/step - accuracy: 0.7101 - loss: 0.6952 - val_accuracy: 0.6670 - val_loss: 0.8430
Epoch 9/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 97s 198ms/step - accuracy: 0.7166 - loss: 0.6850 - val_accuracy: 0.6670 - val_loss: 0.8422
Epoch 10/10
488/488 ━━━━━━━━━━━━━━━━━━━━ 96s 196ms/step - accuracy: 0.7211 - loss: 0.6758 - val_accuracy: 0.6695 - val_loss: 0.8481
```

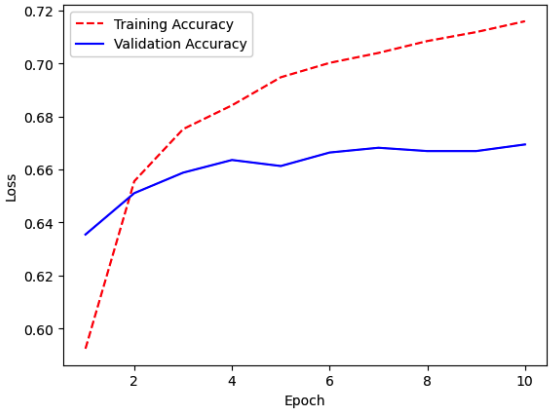
Slika 3.2. Tijek treniranja modela po epohama

# 4. EVALUACIJA MODELA

Na slikama 4.1. i 4.2. prikazano je kretanje gubitka i kretanje preciznosti za trening i validacijski skup podataka, a iz grafova je jasno da su te dvije vrijednosti obrnuto proporcionalne. Preciznost modela na trening bi s dodatnim epohama nastavila polagano rasti, odnosno model bi učio trening skup sve bolje i bolje, no to ne znači da bi dobro reagirao na nepoznatom skupu podataka. Preciznost modela za validacijski skup stoga je bolji indikator jer pokazuje kako model reagira na nepoznati skup.



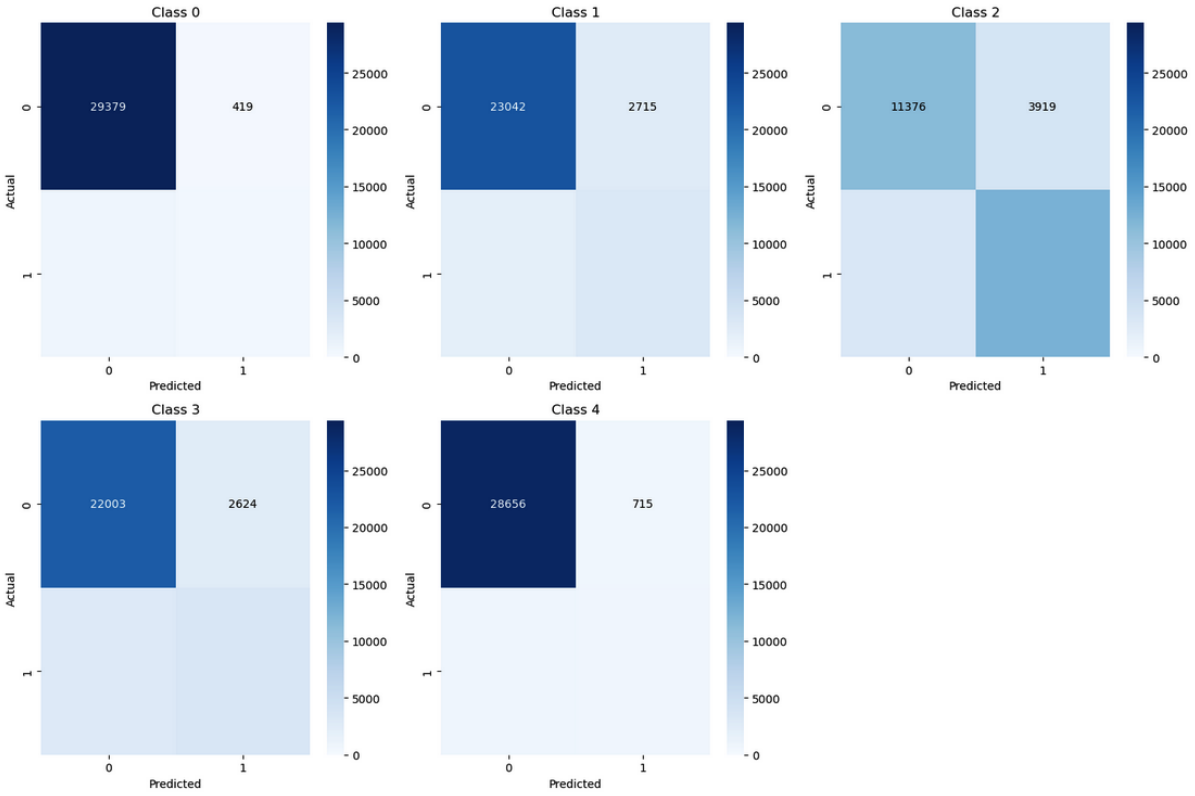

Slika 4.1. Kretanje gubitka




Slika 4.2. Kretanje preciznosti

Kako to pokazuje graf preciznosti za validacijski skup, model dosegne maksimum preciznosti već oko treće epohe te iznosi oko 66% što je s obzirom na kompleksnost implementacije i važnost ovakvog modela prihvatljivo, a kako su obje preciznosti usporedive, jasno je da nema *overfittinga*.

Na slici 4.3. prikazane su matrice konfuzije za svaku klasu, odnosno sentiment. Matrica konfuzije su tablice koje vizualiziraju koliko dobro algoritam klasificira podatke u određenu klasu. Iz ovih matrica vidljivo je da algoritam najbolje klasificira dvije ekstremne klase 0 i 4, odnosno negativan i pozitivan sentiment, dok se najviše muči s klasom 2, odnosno neutralnim sentimentom. Takav ishod je, doduše, potpuno očekivan. Negativan sentiment je računalu najteže razumjeti jer unutar neutralnih fraza, tj. recenzija, nema jakih indikatora o sentimentu poput riječi *good, super, brilliant, bad, awful, horrible* ili su takve recenzije preduge, prekompliciranog konteksta, koriste sarkazam i sl.



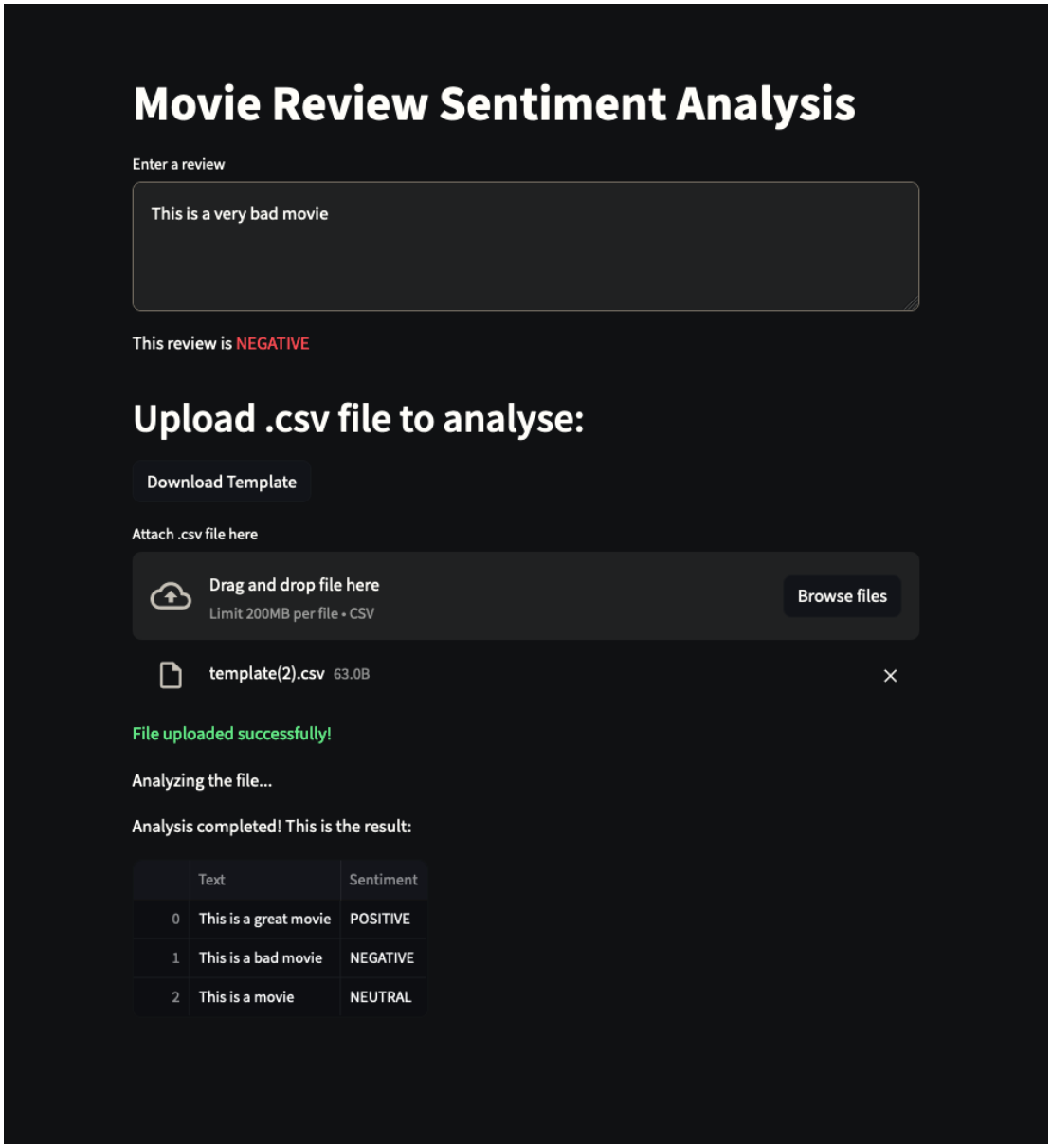

Slika 4.3. Matrice konfuzije za svaku klasu

# 5. STREAMLIT APLIKACIJA

*Streamlit* je *open-source Python* biblioteka koja se koristi za brzo i jednostavno kreiranje web aplikacija za obradu podataka i strojno učenje. Omogućava korisnicima da kreiraju interaktive web aplikacije koristeći samo *Python* skripte, bez potrebe za poznavanjem *front-end* tehnologija kao što su *HTML*, *CSS* i *JavaScript*.

Za potrebu ručnog testiranja i demonstracije ranije istreniranog modela kreirana je jedna takva *Streamlit* aplikacija koja pruža korisniku dvije različite opcije upisa podataka za testiranje: običan tekst, odnosno nekoliko riječi ili rečenica, ili prijenos *.csv* datoteke koja u sebi ima stupac upisanih recenzija.

Unutar *Python* skripte, oba ulaza su obrađena na isti način kao i podaci za treniranje modela te su zatim proslijeđena u metodu *model.predict()*. Vraćeni rezultat metode, tj. rezultat predviđanja sentimenta prilagođen je za korisničko sučelje te je prikazan ispod svakog ulaznog elementa. U slučaju tekstualnog upisa, to je obojani naziv klase sentimenta, dok je u slučaju prenesene *.csv* datoteke prikazana tablica cijele datoteke s dodatnim stupcem – sentimentom za svaki odgovarajući redak.




Slika 5.1. Sučelje *Streamlit* aplikacije

# 6. ZAKLJUČAK

Ovaj model za analizu sentimenta filmskih recenzija postigao je preciznost od 66%. Iako takva preciznost, s obzirom na kompleksnost implementacije modela, može biti prihvatljiva jer je riječ o savladavanju osnova strojnog učenja i dubokog učenja, za korištenje ovakvog modela u stvarnim aplikacijama i sustavima, potrebna su značajna poboljšanja.

Kao što je već spomenuto, model se najviše muči s klasifikacijom neutralnog sentimenta. Potencijalno unaprjeđenje za taj problem je kvalitetnija predobradba podataka. Konkretno, korisno bi bilo potpuno izbaciti sve riječi iz skupa koje nemaju semantičko značenje koje utječe na sentiment - veznici, vlastite imenice i sl.

Osim toga, veći skupovi podataka za treniranje, dodavanje i podešavanje slojeva modela te optimizacija njihovih parametara također bi poboljšali preciznost modela.

Uz sve nedostatke, i ovako jednostavan model svejedno vjerno dočarava njegove snage i ogroman potencijal za upotrebu u raznim industrijama.