УНИВЕРЗИТЕТ У НИШУ
ЕЛЕКТРОНСКИ ФАКУЛТЕТ

- ПРОЈЕКАТ -

АНАЛИЗА СЕНТИМЕНАТА

ПРЕДМЕТ: ОБРАДА ПРИРОДНИХ ЈЕЗИКА

Ментор: Сузана Стојковић **Студент:** Филип Трајковић 1574

Ниш, 2024.

Садржај

1. Увод

У савременом добу где се вештачка интелигенција широм примењује у различитим областима софтверског развоја, кључно је разумети њен начин функционисања и препознати потенцијалне могућности за њену примену. Да бисмо идентификовали те могућности, прво је битно стећи увид у различите технологије, савладати бројне концепте који су развијени у последње време, а такође и разумети како применити одређене технологије у конкретним сценаријима. Кроз ова разумевања, могуће је креирати систем вештачке интелигенције способан за решавање различитих проблема.

Системи за детекцију, разумевање и контекстуализацију текстуалних записа представљају широку област која се веома брзо развија почетком треће деценије 21. века. Убрзани развој и помама за оваквим системима креће крајем 2022. године са појавом до сада невиђеног система за обраду текстова под називом *eng. ChatGPT*. Широке корисничке масе које су се уз помоћ овог система упознале са могућности коју нуде системи за обраду текстова, додатно су убрзале даљи развој и напредак у обради природних језика стварајући нове случајеве коришћења овакве технологије.

Анализа сентимената представља један од основних случајева коришћења вештачке интелигенције у области обраде природних језика. Користе се процеси класичног машинског учења и процеси обраде природних језика како би се креирала једна симбиоза различитих области са циљем решавања специфичних проблема попут анализе сентименталних особина текстуалних записа. Анализа сентимената текстуалних записа представља област надгледаног учења у машинском учењу при чему је неопходно да се за сваки запис у скупу података наведе и припадност категорији којој припада наведени текст. Сви записи у скупу података требају припадати коначном скупу категорија. Решавање проблема анализе сентимената се своди на проблем класификације у машинском учењу.

Даљи текст овог рада је подељен на теоријски и практични део. У теоријском делу рада су обрађене технике обраде текстуалних података, док су у практичном делу рада приказани резултати ових обрада.

Теоријски део рада обухвата основну теоријску позадину иза техника обраде текстуалних података, док практични део рада обухвата примену наведених техника над специфичним скупом података са применом класификационих алгоритама машинског учења, као и уз помоћ претренираних језичких модела.

2. Анализа сентимената у обради природних језика

Анализа сентимената представља случај коришћења примене класификације у обради природних језика. Класификација текста се бави проучавањем означених скупова података при чему се скуп података састоји од текстуалног записа и категоричког атрибута који представља категорију припадности из скупа коначних категорија. Анализа сентимената је случај у којем текстуални подаци носе сентименталну контекстуалну вредност на основу којих је могуће одредити сентимент датог податка и категорисати га у неку од могућих сентиментално-квалитативних категорија.

2.1. Теоријске основе анализе сентимената

Концепт анализе сентимената као специфичног случаја обраде текстуалних података представља процес утврђивања квалитативних својстава емоционалног тонуса текстуалних записа како би се из датог скупа података извеле одређене законитости које важе између контекстуалних репрезентација посматраних емоционалних записа и категоричког атрибута који представља сентиментални показатељ припадности одређеном скупу.

Циљ овог процеса јесте утврђивање емоционалног контекста датих текстова и тежину коју речи природног језика имају у односу на стање сентимента који описују. Ставови тј. контекст који се преноси кроз текст се обично означавају у неколико категорија попут:

- Позитивно
- Негативно
- Неутрално

2.2. Примена анализе сентимената

Процес утврђивања сентемената у тексту са собом доноси нове случајеве примене оваквог типа технологија у различитим сферама. Примена анализе сентимената има највећих бенефита у сферама које се баве обрадом и анализом мишљења корисника, оцену одређених ентитета (брендова, производа, итд.), нивоима одушевљења или разочарења посматраног корисника и сл. Овакви случајеви коришћења могу додатно допринети анализи одређених појава јер до сада није било могуће аутоматизовати анализу и процесирање текстова на овај начин.

Овакав аутоматизован начин обраде великих текстуалних скупова података са собом доноси убрзани начин обраде и анализе различитих текстова.

Коначни резултати утврђивања контекста текстова имају за циљ презентовање јасније слике о стеченим мишљењима различитих корисника над датим објектима посматрања при чему се у анализу могу укључити и додатни параметри који могу додатно категоризовати типове личности корисника.

2.3. Технике обраде природног језика

Технике обраде природног језика представљају технике којима се обрађују подаци који се у изворном облику налазе у текстуалном формату. Оваквим техникама се врши обрада и припрема података за даље кораке у развоју процеса анализе сентимената попут примене математичких метода машинског учења или тренирања модела.

Обзиром да над подацима који се изворно налазе у текстуалном формату није могуће применити споменуте даље кораке, такве податке је најпре неопходно претворити у погодан формат који се може обрађивати на хардверу који се користи у машинском учењу. Погодан формат јесте векторски/тензорски простор у којем је неопходно представити текстуални контекст посматраног скупа података.

Обрађени текстуални подаци постају вектори који се у векторском/тензорском простору представљају као низови бројева одређене дужине. Вредности које ови вектори носе са собом треба да на високо-квалитативан начин опишу дати текстуални податак при чему треба одржати својства текстова попут редоследа речи у реченици, контекстуалне зависности између речи, дужина текстова, контекстуална сличности и сл.

2.3.1. Препроцесирање података

Препроцесирање података се бави основним чишћењем улазних текстуалних података и подразумева неке од следећи техника:

- Чишћење текста
- Нормализација
- Токенизација
- Лематизација и стемовање
- Уклањање стоп-речи

Чишћење текста

Чишћење текста је први корак у препроцесирању текстуалних података и подразумева уклањање специјалних карактера из текста попут HTML тагова, знакова интерпункције, бројева и осталих не-текстуалних карактера. На овај начин се врши елиминација сувишних симбола из текста и крера једноставан и чистији текст.

Нормализација

Нормализација представља корак у препроцесирању текстуалних података при чему се врши конверзија текста у јединствени тј. стандардни формат за све карактере текста. Основни процес у номализацији јесте *case* конверзија при чему се текст претвара у *lower* или *upper case.*

Токенизација

Токенизација представља корак у препроцесирању текстуалних података при чему се улазна реченица дели на листу краћих текстуалних јединица које могу бити појединачне речи или делови речи, чак и појединачна слова. Токени представљају смислену целину која представља заједнички контекст одређених делова речи, тј. целине токенизоване реченице. У највећем броју случајева се токенизација врши поделом улазне реченице на листу речи. Оваква листа је погодна за даље кораке у препроцесирању података.

Лематизација и стемовање

Ове технике представљају технике обраде текста које се користе за конверзију и скраћивање речи на њихов основни облик или корен речи. Технике се примењују на идентичан начин при чему се примењују различите трансформације.

Стемовање представља једноставнији процес обраде речи при чему се посматрана реч своди на корен речи изведен из посматране. Додатна битна карактеристика јесте и време извршавања самог процеса обраде података. Такође, стемовање се обавља над сваком речју појединачно без додатног познавања контекста осталих речи у околини.

Лематизација представља сложенији процес обраде текста у односу на стемизацију јер у процесу обраде података узима шири контекст и већи број речи како би креирао „лему“. Лема представља основну форму речи, тј. облик речи који представља њено лексично значење или основни облик. Лематизација је процес редукције речи при чему се све варијације одређене речи своде на идентичан лексички облик.

Уклањање стоп-речи

Стоп-речи представљају специфичан корпус речи сваког језика које са собом не носе високи значај у обради и анализи текста. Овакве речи је приликом обраде текста могуће извацити из скупа улазних реченица јер не доприносе квалитативним својствима посматраних текстова, док повећавају време процеса обраде текста. Ове речи се сматрају често коришћеним у језику и не доприносе контекстуалној репрезентацији улазних реченица.

Стоп-речи су углавном често коришћене речи, везници, предлози, речце и сл. Примери оваквих речи у енглеском речнику су следеће: “the”, “a”, “and”, “is”, “in”, “into”…

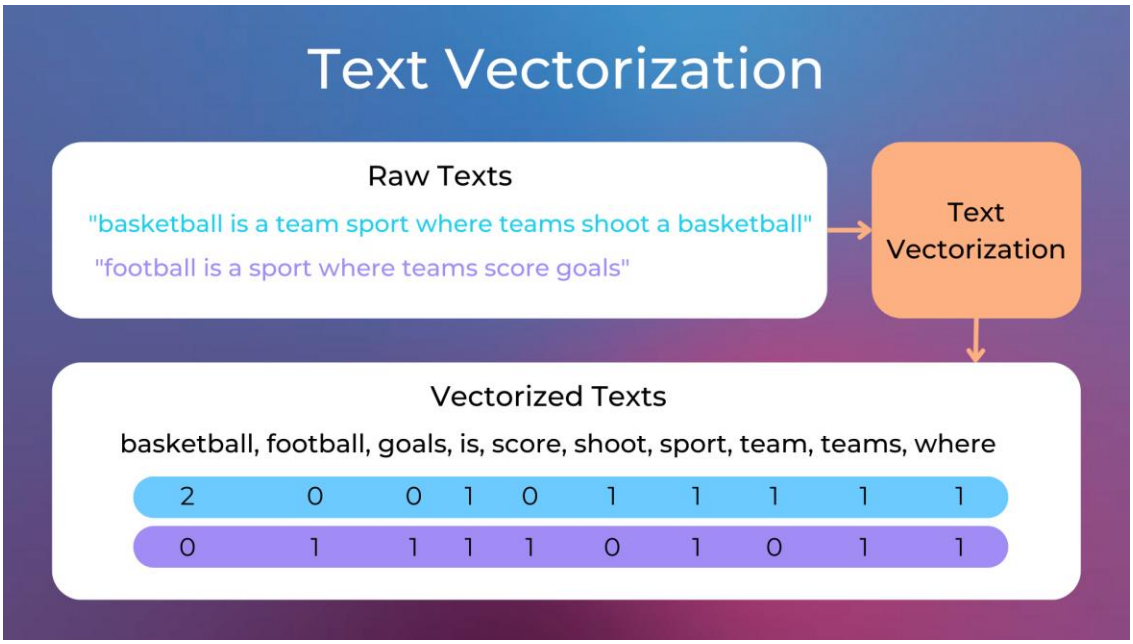
У овом одељку су наведене само неке од техника за препроцесирање улазног текста како би се прилагодио за даљу анализу и припрему података за тренирање.

2.3.2. Припрема података за тренирање

Процес припреме података за тренирање представља поступак креирања статистичких параметара о скупу реченица које се обрађују. Улазни подаци овог процеса представљају препроцесирани подаци из претходног поглавља. На улазу се добијају обрађени подаци у текстуалном формату који се даље прослеђују у процесу векторизације припреме података.

2.3.2.1. Векторизација текстова

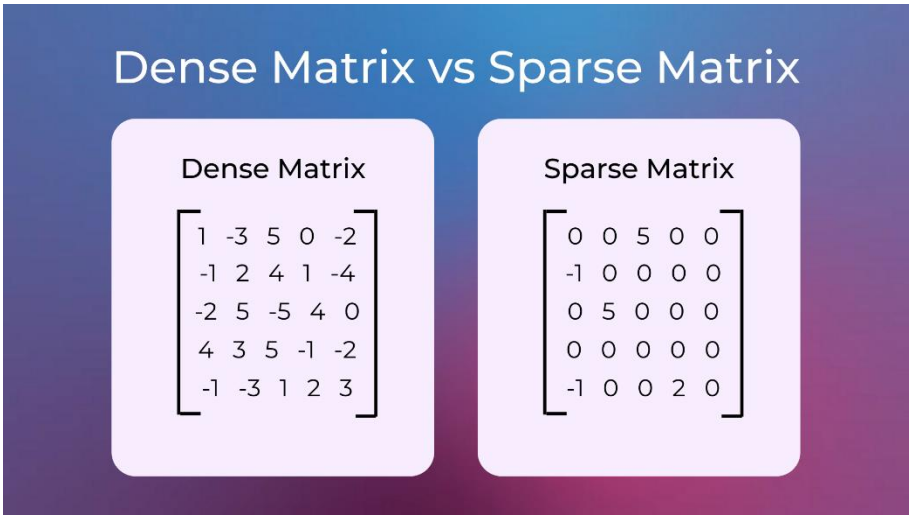
Овај процес подразумева конверзију улазних текстуалних података у нумеричке податке које може рачунар да разуме како би се даље вршило прилагођење (eng. *fitting)* . Нумерички подаци се представљају у облику векторизованих елемената матрице при чему сваки текст представља један ред матрице. Овакве матрице се називају ретко поседнутим (*eng. sparse matrix*).



Сл. 1 – Процес векторизације реченице

На слици изнад је приказан процес векторизације при чему свака реч у унији речи улазних текстова представља једну колону матрице. Сваки ред ове матрице представља појединачне улазне реченице, док вредности матрице по колонама представљају број понављања датих речи у свакој реченици понаособ. У случају када је корпус реченица изразито велики и различитог контекста, вокабулар корпуса ће бити већи што ће утицати на повећани број колона ове матрице. Уколико би текстови били различитих контекста, добили бисмо веома ретко поседнуту матрицу код које би велика већина вредности матрице имао вредност 0.

На слици испод се може видети како изгледа репрезентација ретко поседнуте матрице (*eng. sparse matrix*) у односу на густо поседнуту матрицу (*eng. dense matrix*).



Сл. 2 – Приказ репрезентације густо (лево) и ретко (десно) поседнутих матрица

Два основна начина за векторизацију текстова су:

- Bag of words
- Term frequency-Inverse document frequency

Bag of words

Bag of words (врећа речи) представља приступ формирања ретко поседнуте матрице описан у примеру претходног поглавља где се у матрици за сваку реч означава број понављања у свакој реченици појединачно. Овај начин обраде улазних текстова поседује вокабулар речи који се добија као скуп свих могућих речи из целокупног улазног скупа података. На основу вокабулара речи се затим креира мера понављања одређених речи из вокабулара.

Уз помоћ овог приступа се не могу запамтити комплексније зависности између речи у реченици попут структуре и редоследа, већ се памти само информација да ли одређена реч постоји или не у текстовима.

Главни проблем овог приступа је доминанта учесталост одређених речи унутар документа тј. улазног текста при чему такве речи не морају нужно да носе довољан информативни контекст. Контекст документа се заснива на речима које су специфичне домену који се обрађује, а такве речи могу имати мању учесталост у документима.

Term frequency-Inverse document frequency

Овај приступ представља унапређену верзију BOW приступа при чему се у првом делу обраде користи резултат добијен BOW методом. Термин *Eng. Term frequency* се односи на учесталост понављања токена унутар докумената и рачуна се као број понављања одређене речи подељено са бројем речи у посматраној реченици. Овај параметар предстљавља значајност дате речи у датом документу.

Термин *Eng. Inverse document frequency* се односи на учесталост појављивања датих речи у целокупном скупу података. Коначни резултат јесте производ две добијене вредности.

У следећим формулама је приказан детаљан начин израчунавања ових вредности.

$$TF = \frac{Number\ of\ times\ a\ word\ "X"\ appears\ in\ a\ Document}{Number\ of\ words\ present\ in\ a\ Document}$$

$$IDF = log\left(\frac{Number\ of\ Documents\ present\ in\ a\ Corpus}{Number\ of\ Documents\ where\ word\ "X"\ has\ appeared}\right)$$

$$\text{TF-IDF (x, Document, Corpus)=TF (x, Document)} \times \text{IDF (x, Corpus)}$$

Коначни резултат овог израчунавања је ретко поседнута матрица са идентичним колонама и врстама као у претходном случају, по колонама су документи/реченице док су по колонама појединачне речи.

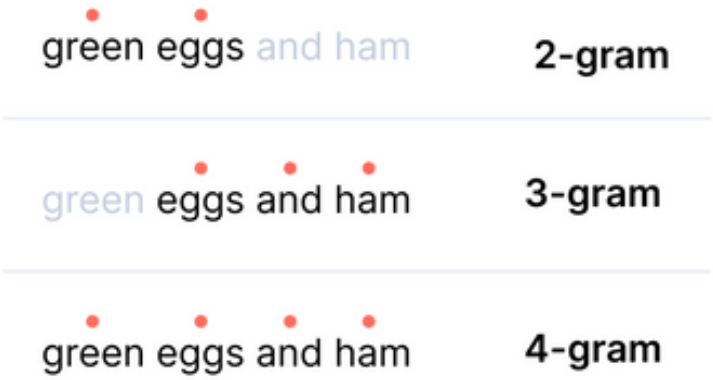
Вредност за сваку реч представља значајност речи у датом документу и у односу на целокупан корпус. Она омогућава да на тај начин дођу до изражаја речи специфичне за контекст посматраног документа.

2.3.2.2. Груписање речи помоћу N-грамова

Додатан напредак у домену контекстуализације докумената и хардверских захтева приликом припреме података за тренирање модела представља груписање речи у N-грамове. N-грамови представљају синтагме речи у реченици које се односе на сличан појам. У досадашњем тексту је због комплексности у обради тематике описан рад над 1-gram подацима, тј. код векторизације текстова се анализирала свака реч као засебна јединица (*eng. gram*).

Битно је напоменути да се могу креирати случајеви са 2-*gram* и 3-*gram* репрезентацијама при чему се скуп 2 или 3 речи посматра као засебна јединица у векторизацији текстова. Овакан корак у припреми података може додатно допринети контекстуализацији вектора, смањењу величине ретко поседнуте матрице и смањењу времена извршења самог процеса припреме података и тренирања.

Проблем са 2 или 3 речи у засебној јединици обраде јесте тај што се он примењује у случајевима када је корпус реченица изразито велики јер је тада учесталост таквих синтагми повећана у односу на мали корпус реченица.



Сл. 3 – Пример 2, 3 и 4-*gram* издвојених јединица за обраду

Наведене технике за обраду природног језика представљају основне кораке у припреми скупа података који се користи за даљи процес тренирања језичких модела. У случају анализе сентимената, неопходни формат је управо векторска матрица која се добија применом векторизације над препроцесираним скупом података. Скуп података припремљен на тај начин је спреман за процес тренирања. У даљем тексту извештаја у делу практичног дела је детаљније описан поступак тренирања модела.

3. Практични део

Практични део овог рада се састоји од практичне имплементације концепата које се баве прибављањем, обрадом и анализом података при чему се у завршном делу рада врши класификација скупа података са циљем добијања коначних резултата о анализи сентимената одабраног скупа података. Практична имплементација се састоји од следећих поглавља:

- Учитавање библиотека
- Учитавање и преглед основног скупа података
- Препроцесирање скупа података
- Експлоративнаа анализа података (*eng. EDA - Explanatory data analysis*)
- Припрема података за тренирање модела
- Тренирање модела

3.1. Учитавање библиотека

За рад на практичном делу пројекта неопходно је креирати виртуелно окружење у *Python* програмском језику и инсталирати све неопходне библиотеке за рад. Библиотеке које се користе су:

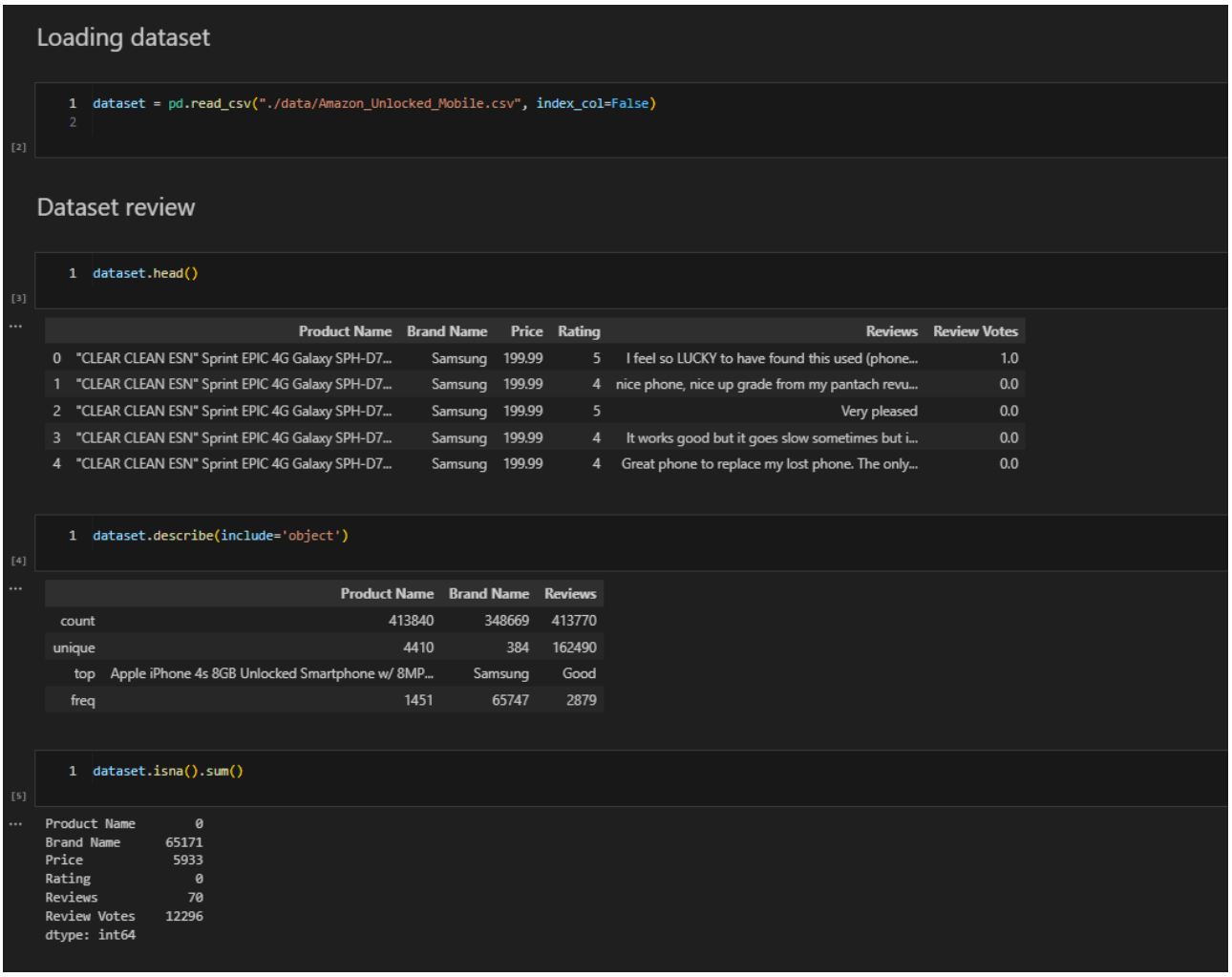
- ipykernel
- pandas
- matplotlib
- seaborn
- scikit-learn
- nltk
- transformers
- tensorflow

Наведене библиотеке је неопходо инсталирати командом pip install <назив библиотеке>, а затим све неопходне модуле ових библиотека укључити у пројекат на почетку *jupyter* радне свеске.

3.2. Скуп података

Одабрани скуп података над којим ће се применити сви до сада обрађени концепти анализе сентимената јесте *Amazon Reviews: Unlocked Mobile Phones* који садржи више од 400 000 узорака података. Овај скуп података описује рецензије купљених откључаних телефона преко сајта *Amazon.com*. Додатне информације о скупу података се могу пронаћи на следећем линку:

https://www.kaggle.com/datasets/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones



Сл. 4 – Учитавање и преглед скупа података

Скуп података се састоји атрибута *Product Name*, *Brand Name*, *Price*, *Rating*, *Reviews*, *Review Votes*. За процес анализе сентимената ће бити од користи атрибути *Reviews* и *Rating*. Из основног прегледа скупа података, могуће је утврдити да велики број података нема попуњене податке за одређене атрибуте и да треба обратити пажњу приликом

препроцесирања и филтрирања скупа података како би се сачувао највећи број узорака скупа података који имају валидне вредности за два атрибута од значаја.

3.3. Препроцесирање скупа података

У практичном делу рада приликом препроцесирања података примењене су следеће технике:

- Издвајање значајних атрибута
- Енкодирање *Rating* атрибута
- Избацивање редова без попуњених вредности (око 70 редова)
- Избацивање *URL* тагова из улазних текстова
- Избацивање карактера који нису алфа-нумерички
- Пребацивање текстова у мала слова (*eng. lower case*)
- Токенизација текстова
- Избацивање стоп речи
- Лематизација (или стемовање)
- Избацивање редова са више од 2500 токена у текстуалној рецензији
- Скраћивање скупа података на мањи број погодан за тренирање модела

На следећим сликама је приказана практична имплементација појединих техника за препроцесирање текстова у формату појединачних функција које се касније учитавају и :

```
16  def Encode_rating(rating: int) -> str:
17      if rating >= 4:
18          return "Positive"
19      elif rating <= 2:
20          return "Negative"
21      else:
22          return "Neutral"
23
24  def LabelEncode_rating(rating: int) -> int:
25      if rating >= 4:
26          return 2
27      elif rating <= 2:
28          return 0
29      else:
30          return 1
31
32
33  def Removing_url(review: str) -> str:
34      return re.sub(r'http\S+', '', review)
35
36
37  def Clean_non_alphanumeric(review: str) -> str:
38      return re.sub('[^a-zA-Z]', ' ', review)
39
```

Сл. 5 – Први део функција за препроцесирање

```python
def Convert_to_lowercase(review: str) -> str:
    return str(review).lower()


def Tokenize_text(review_text: str) -> str:
    return word_tokenize(review_text)


def Remove_stopwords(token: str) -> list[str]:
    return [item for item in token if item not in stop_words]


def Stemming(review_text: str) -> list[Any]:
    return [stemmer.stem(token) for token in review_text]


def Lemmatization(review_text: str) -> list[str]:
    return [lemma.lemmatize(word=token, pos='v') for token in review_text]


def Remove_short_words(review_text: str) -> list[str]:
    return [token for token in review_text if len(token) > 2]


def Convert_list_of_tokens_to_string(reviews_list: list) -> str:
    return ' '.join(reviews_list)
```

Сл. 6 – Други део функција за препроцесирање

3.4. Експлоративна анализа података

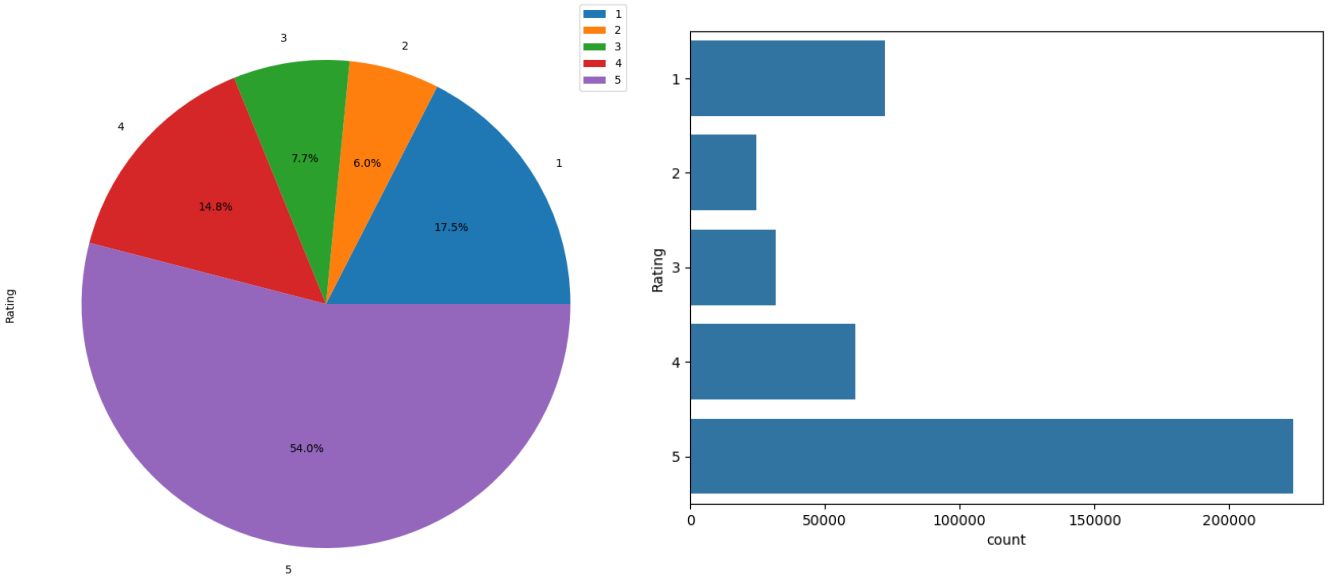
Експлоративна анализа података служи како би се путем различитим графичких алата омогућила боља визуализација података, расподела вредности по посматраним атрибутима, учесталост одређених токена у скупу података, дистрибуција дужина реченица у скупу података и сл.

Визуализација расподеле сентимената

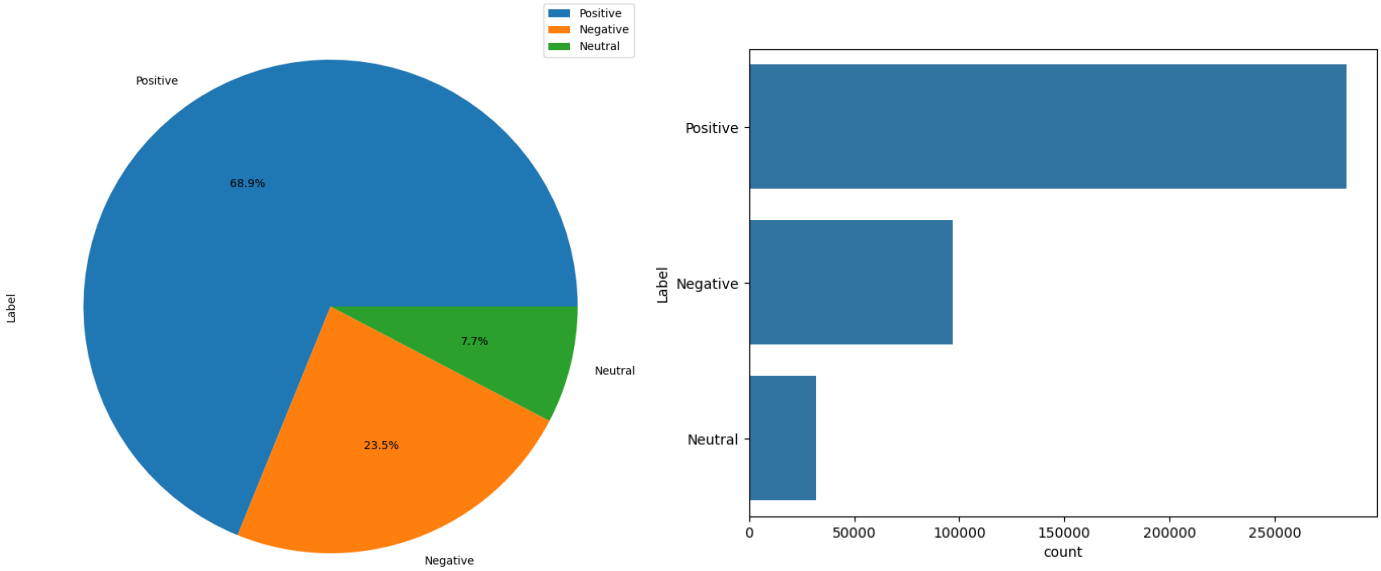
Визуализација расподеле сентимената, тј. категоричког атрибута који описује којој категорији сентимента посматрани текст припада, приказује учешће одређених категоричких атрибута у целокупном скупу података и број понављања.

Визуализација оваквих података је обављена уз помоћ *pd.Dataframe.plot.pie* и *sns.countplot* при чему је у зависности од приступа потребно прилагодити улазне податке на одређени начин.

На визуализацији је приказана расподела не енкодираних вредности *Rating* атрибута, као и енкодираног *Label* атрибута.



Сл. 7 – Графички приказ удела појединачних вредности *Rating* атрибута пре фазе енкодирања



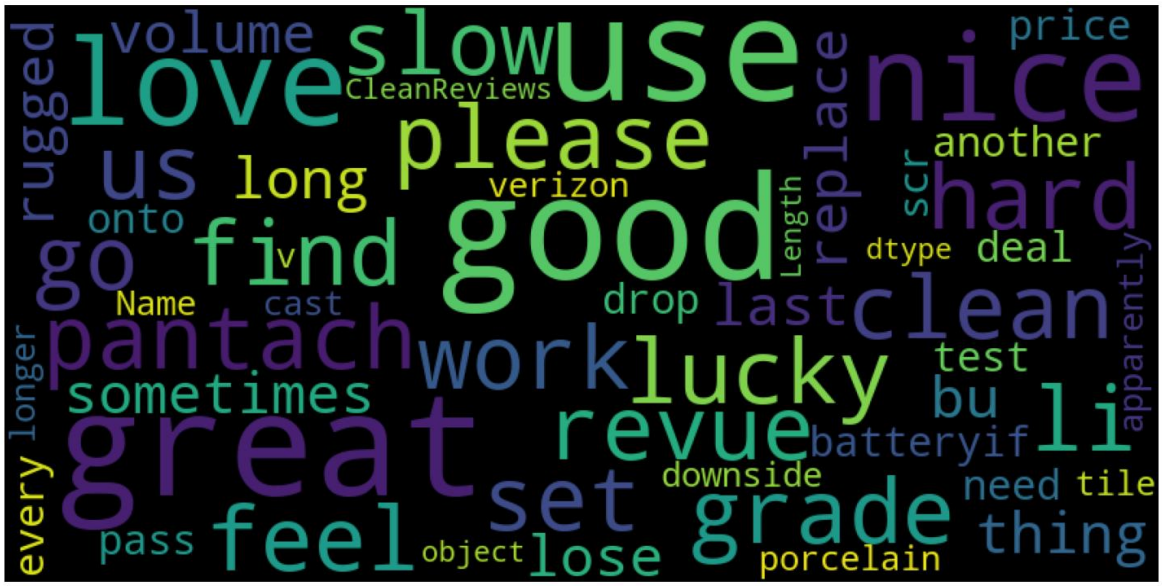
Сл. 8 – Графички приказ удела појединачних вредности *Label* атрибута након фазе енкодирања

Из овог случаја се може приметити да удео позитивних коментара има доминантну улогу у скупу података и да на овакву расподелу узорака треба обратити пажњу приликом тренирања класификатора.

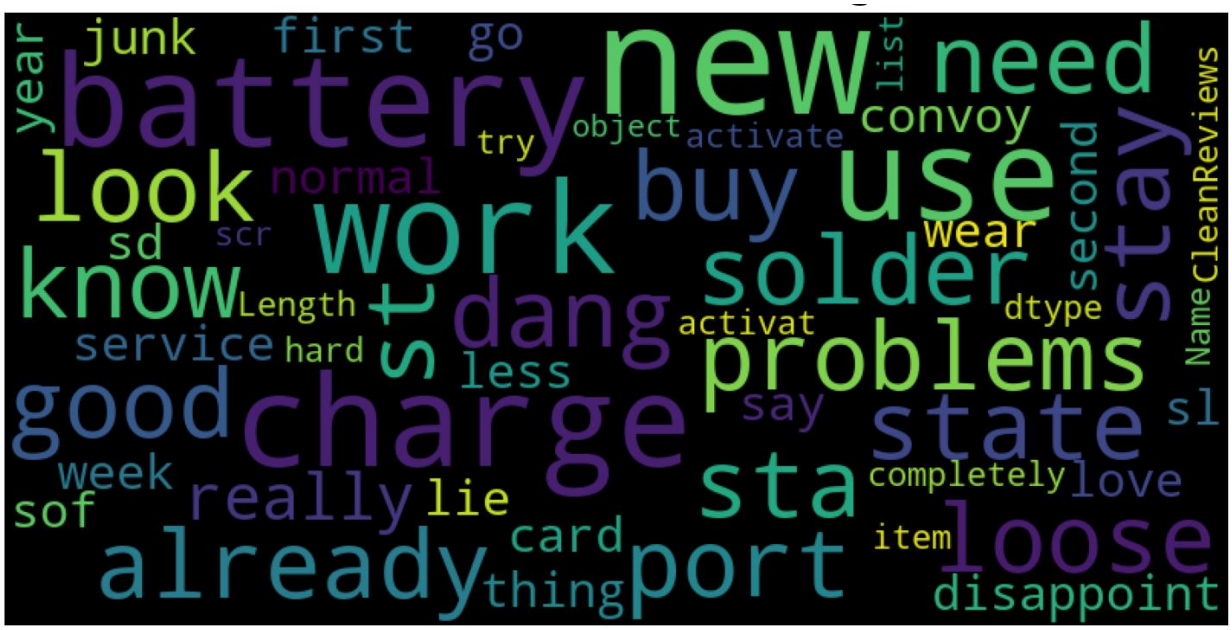
Визуализација најучесталијих речи помоћу *WordCloud*

WordCloud представља концепт који олакшава израчунавање и приказ најучесталијих речи из улазног скупа података. У практичном делу је најпре улазни скуп података подељен на скуп података са позитивних и негативних сентиментима.

На следећим визуализацијама су приказане најучесталије речи у оба споменута скупа података.



Сл. 9 – Речи са највећим број појављивања у позитивном подскупу корпуса

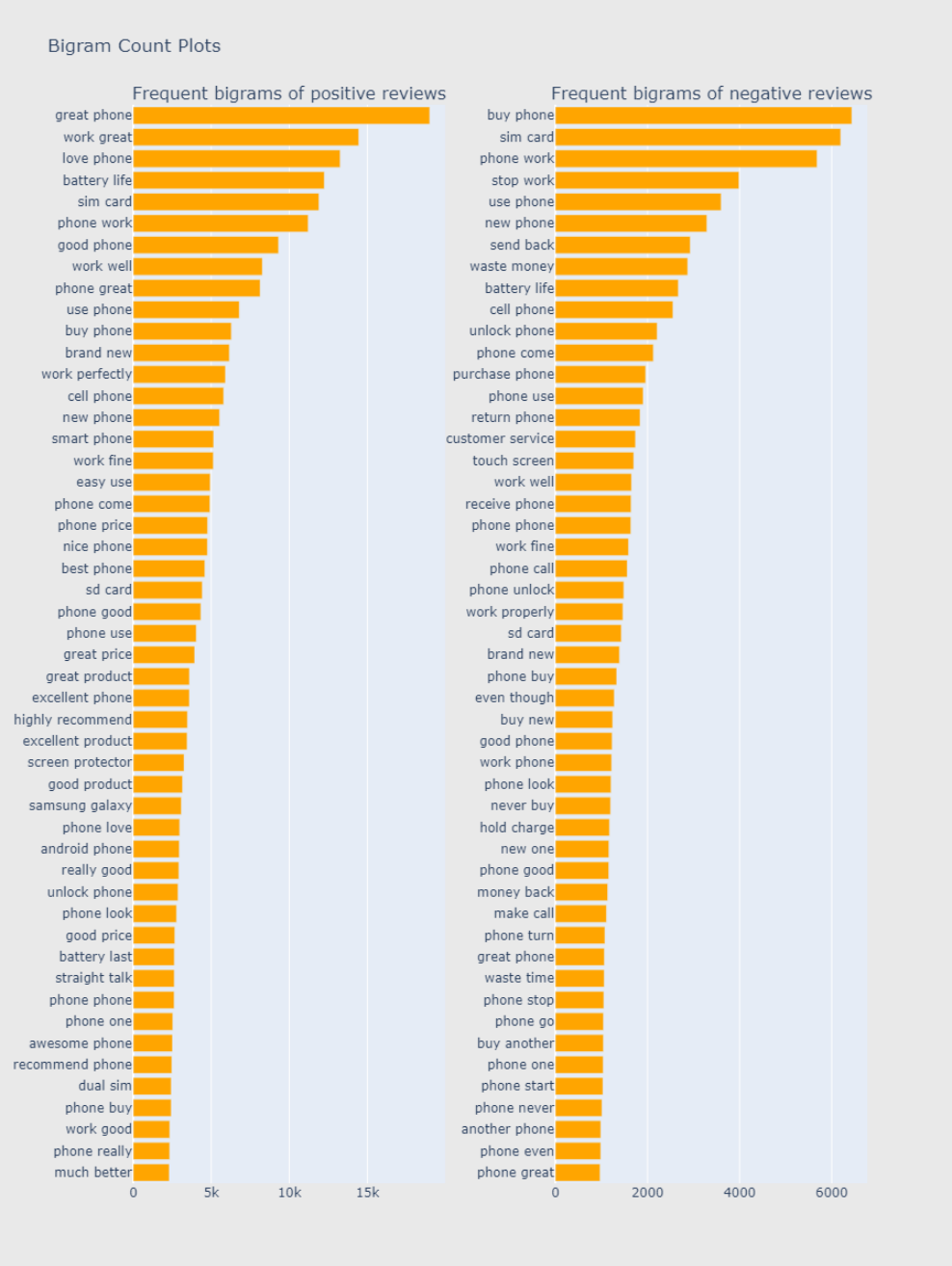


Сл. 10 – Речи са највећим број појављивања у негативном подскупу корпуса

N-gram визуалиција

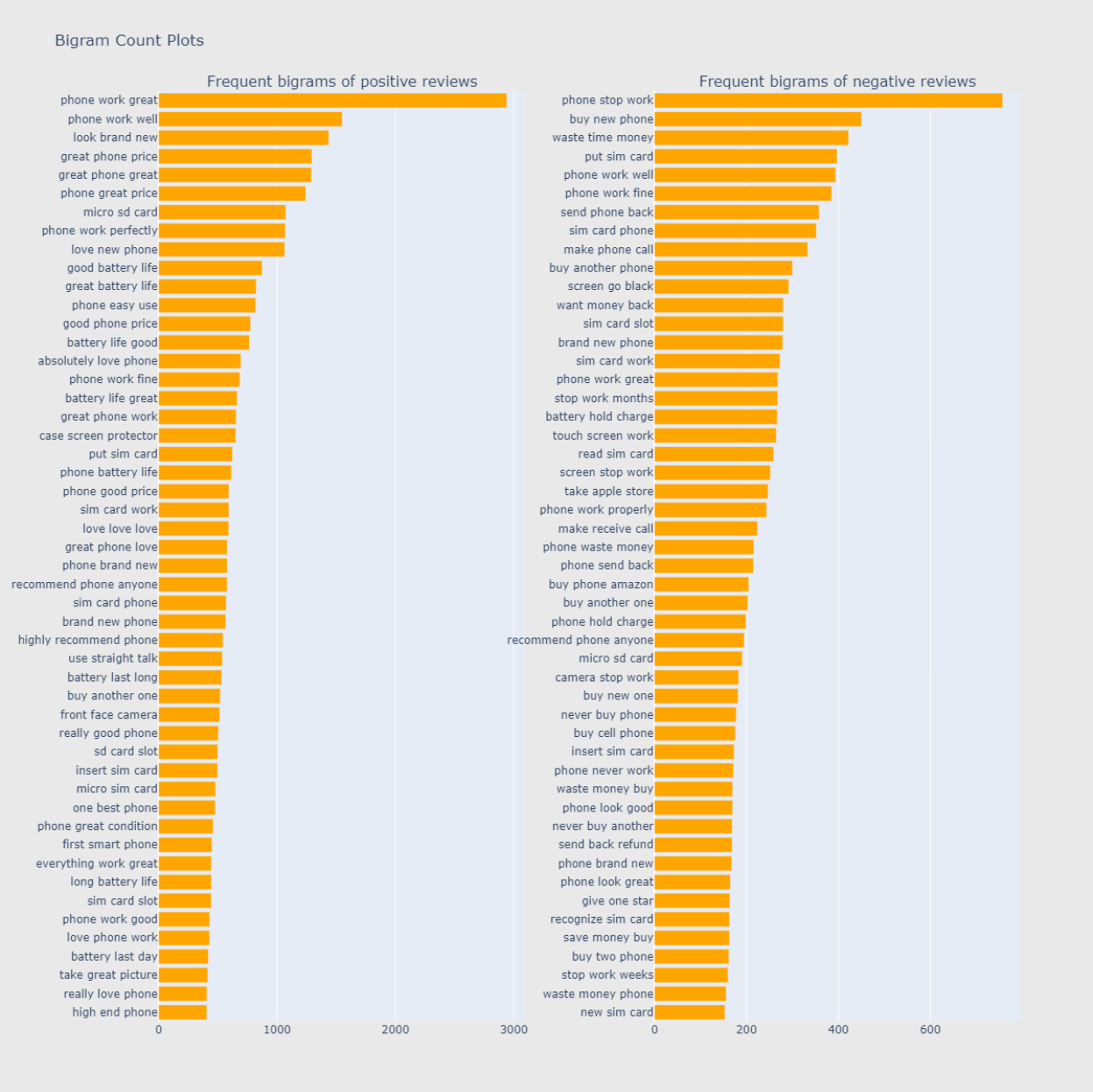
Овај тип визуалиције представља сличан начин представљања фреквенције појединачних *gram*-ова у целокупном корпусу текстова. Уз помоћ ове методе могуће је добити фреквенције вишеструких *gram*-ова (2-*gram*, 3-*gram*…). Предуслов за примену *N-gram* визуализације је претходно креиран *N-gram* скуп података.

На следећим графицима су приказани резултати овакве визуализације.



Сл. 11 – 2-*gram* узорци са највећим бројем појављивања у позитивном подскупу корпуса

На датим сликама је могуће приметити како се позитивни или негативни контекст са различитим фреквенцијама налази у датим подскупима података. Може се уочити позитиван контекст 2-*gram* и 3-*gram* узорака у позитивном подскупу (лево), као и идентична зависност код негативних случајева у негативном подскупу (десно). Из наведених резултата може се даље анализирати утицај на позитивно или негативно задовољство корисника који су остављали своја запажања и критике.



Сл. 12 – 3-*gram* узорци са највећим бројем појављивања у позитивном подскупу корпуса

3.5. Припрема података за тренирање модела

У фази припреме података у практичном делу рада извршена је векторизација података као последњи корак у припреми података за тренирање модела и класификационе алгоритме. У самом процесу векторизације текстова коришћене су обе методе векторизације података претходно описане у теоријском делу овом извештаја.

У практичном делу векторизација података је урађена за документе корпуса при чему је коришћена и 1-gram и 3-gram репрезентација текстова.

Коришћени објекти за векторизацију текстова су:

- CountVectorizer
- TfidfTransformer
- TfidfVectorizer

Векторизација текстова коришћењем 1-*gram* репрезентација

На наредним сликама је приказан начин за припрему података над 1-gram репрезентацијама.

Preparing data using CountVectorizer

```
1  bow_transform = CountVectorizer(lowercase=False)
2  X_tr_bow = bow_transform.fit_transform(training_data['CleanReviews'])
3  y_tr = training_data['Label']
4
5  y_bow_transform = CountVectorizer(lowercase=False)
6
7  y_tr_transformed_using_bow = y_bow_transform.fit_transform(training_data['Label'])
8
9  print("Shape of X_tr_bow: ", X_tr_bow.shape)
10 print("Shape of y_tr: ", y_tr.shape)
11
12 print("len(bow_transform.vocabulary_): ", len(bow_transform.vocabulary_))
13
14 X_te_bow = bow_transform.transform(test_data['CleanReviews'])
15 y_te = test_data['Label']
16
17 y_te_transformed_using_bow = y_bow_transform.transform(test_data['Label'])
18
19 print("Shape of X_te_bow: ", X_te_bow.shape)
20 print("Shape of y_te: ", y_te.shape)
21
22 print("Shape of y_tr_transformed_using_bow: ", y_tr_transformed_using_bow.shape)
23 print("Shape of y_te_transformed_using_bow: ", y_te_transformed_using_bow.shape)
24
```

```
Shape of X_tr_bow:  (288821, 47134)
Shape of y_tr:  (288821,)
len(bow_transform.vocabulary_):  47134
Shape of X_te_bow:  (123781, 47134)
Shape of y_te:  (123781,)
Shape of y_tr_transformed_using_bow:  (288821, 3)
Shape of y_te_transformed_using_bow:  (123781, 3)
```

Сл. 13 – Векторизација 1-*gram* репрезентација текстова коришћењем CountVectorizer

Preparing data using TfidfTransformer

```
1
2  tfidf_transform = TfidfTransformer(norm=None)
3  X_tr_tfidf = tfidf_transform.fit_transform(X_tr_bow)
4  X_te_tfidf = tfidf_transform.transform(X_te_bow)
5
6  y_tfidf_transform = TfidfTransformer(norm=None)
7
8  y_tr_transformed_using_TfIdfTransformer = y_tfidf_transform.fit_transform(y_tr_transformed_using_bow)
9  y_te_transformed_using_TfIdfTransformer = y_tfidf_transform.transform(y_te_transformed_using_bow)
10
11 print("Shape of X_tr_tfidf: ", X_tr_tfidf.shape)
12 print("Shape of X_te_tfidf: ", X_te_tfidf.shape)
13
14 print("Shape of y_tr_transformed_using_TfIdfTransformer: ", y_tr_transformed_using_TfIdfTransformer.shape)
15 print("Shape of y_te_transformed_using_TfIdfTransformer: ", y_te_transformed_using_TfIdfTransformer.shape)
16
17
```

```
Shape of X_tr_tfidf:  (288821, 47134)
Shape of X_te_tfidf:  (123781, 47134)
Shape of y_tr_transformed_using_TfIdfTransformer:  (288821, 3)
Shape of y_te_transformed_using_TfIdfTransformer:  (123781, 3)
```

Сл. 14 – Векторизација 1-*gram* репрезентација текстова коришћењем TfidfTransformer

Preparing data using TfidfVectorizer

```
1  # Initialize TfidfVectorizer
2  tfidf_vectorizer = TfidfVectorizer()
3
4  # Fit and transform the text data
5  X_tfidf_tr = tfidf_vectorizer.fit_transform(training_data['CleanReviews'])
6  X_tfidf_te = tfidf_vectorizer.transform(test_data['CleanReviews'])
7
8  y_tfidf_vectorizer = TfidfVectorizer()
9  y_tr_transformed_using_TfidfVectorizer = y_tfidf_vectorizer.fit_transform(training_data['Label'])
10 y_te_transformed_using_TfidfVectorizer = y_tfidf_vectorizer.transform(test_data['Label'])
11
12 print("Shape of X_tfidf_tr: ", X_tfidf_tr.shape)
13 print("Shape of X_tfidf_te: ", X_tfidf_te.shape)
14
15 print("Shape of y_tr_transformed_using_TfidfVectorizer: ", y_tr_transformed_using_TfidfVectorizer.shape)
16 print("Shape of y_te_transformed_using_TfidfVectorizer: ", y_te_transformed_using_TfidfVectorizer.shape)
17
18
```

```
Shape of X_tfidf_tr:  (288821, 47134)
Shape of X_tfidf_te:  (123781, 47134)
Shape of y_tr_transformed_using_TfidfVectorizer:  (288821, 3)
Shape of y_te_transformed_using_TfidfVectorizer:  (123781, 3)
```

Сл. 15 – Векторизација 1-*gram* репрезентација текстова коришћењем TfidfVectorizer

Као што је и у теоријском делу извештаја описано, применом векторизације података се добија ретко поседнута матрица. Примери таквих матрица над улазним скупом од 5 текстова су приказани на следећој слици.

1-gram representation of TfidfTransformer transforming results from BOW Transformation

```
1  tfidf_transform = TfidfTransformer(norm=None)
2  X = tfidf_transform.fit_transform(X)
3  pd.DataFrame(X.A, columns=bow_vectorizer.get_feature_names_out())
```

	arrive	buy	charge	daughter	december	freeze	go	good	great	hat	...	picture	price	quality	return	screen	store	today	try	unfreeze	year
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.405465	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	2.098612	1.693147	2.098612	0.000000	2.098612	0.000000	4.197225	0.000000	4.197225	0.000000	...	0.000000	1.693147	0.000000	0.000000	0.000000	2.098612	2.098612	3.386294	0.000000	2.098612
2	0.000000	1.693147	0.000000	2.098612	0.000000	2.098612	0.000000	1.405465	0.000000	0.000000	...	2.098612	0.000000	2.098612	2.098612	2.098612	0.000000	0.000000	1.693147	2.098612	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.098612	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.405465	0.000000	0.000000	...	0.000000	1.693147	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

5 rows × 26 columns

1-gram representation of TfIdfVectorizer without IDF

```
1  vectorizer = TfidfVectorizer(use_idf=False)
2  X = vectorizer.fit_transform(training_data['CleanReviews'][:5])
3  df = pd.DataFrame(np.round(X.A,3), columns=vectorizer.get_feature_names_out())
4  df
```

	arrive	buy	charge	daughter	december	freeze	go	good	great	hat	...	picture	price	quality	return	screen	store	today	try	unfreeze	year
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.0	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.174	0.174	0.174	0.000	0.174	0.000	0.348	0.000	0.348	0.0	...	0.000	0.174	0.000	0.000	0.000	0.174	0.174	0.348	0.000	0.174
2	0.000	0.258	0.000	0.258	0.000	0.258	0.000	0.258	0.000	0.0	...	0.258	0.000	0.258	0.258	0.258	0.000	0.000	0.258	0.258	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.0	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.577	0.000	0.0	...	0.000	0.577	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

5 rows × 26 columns

1-gram representation of TfIdfVectorizer with IDF

```
1  vectorizer = TfidfVectorizer()
2  X = vectorizer.fit_transform(training_data['CleanReviews'][:5])
3  df = pd.DataFrame(np.round(X.A,3), columns=vectorizer.get_feature_names_out())
4  df
```

	arrive	buy	charge	daughter	december	freeze	go	good	great	hat	...	picture	price	quality	return	screen	store	today	try	unfreeze	year
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.765	0.000	0.0	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.201	0.162	0.201	0.000	0.201	0.000	0.402	0.000	0.402	0.0	...	0.000	0.162	0.000	0.000	0.000	0.201	0.201	0.324	0.000	0.201
2	0.000	0.243	0.000	0.301	0.000	0.301	0.000	0.202	0.000	0.0	...	0.301	0.000	0.301	0.301	0.301	0.000	0.000	0.243	0.301	0.000
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.0	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.563	0.000	0.0	...	0.000	0.678	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

5 rows × 26 columns

Сл. 16 – Примери ретко поседнутих матрица креираних на три различита начина

Векторизација текстова коришћењем 3-*gram* репрезентација

На наредним сликама је приказан начин за припрему података над 3-gram репрезентацијама.

Preparing data using CountVectorizer

```
1  bow_transform = CountVectorizer(tokenizer=lambda doc: doc.split(), ngram_range=(3,3), lowercase=False) #, stop_words='english')
2  X_tr_bow = bow_transform.fit_transform(training_data['CleanReviews'])
3  y_tr = training_data['Label']
4
5  print("Shape of X_tr_bow: ", X_tr_bow.shape)
6  print("Shape of y_tr: ", y_tr.shape)
7
8  print("len(bow_transform.vocabulary_): ", len(bow_transform.vocabulary_))
9
10 X_te_bow = bow_transform.transform(test_data['CleanReviews'])
11 y_te = test_data['Label']
12
13 print("Shape of X_te_bow: ", X_te_bow.shape)
14 print("Shape of y_te: ", y_te.shape)
15
```

```
Shape of X_tr_bow:  (288821, 2389884)
Shape of y_tr:  (288821,)
len(bow_transform.vocabulary_):  2389884
Shape of X_te_bow:  (123781, 2389884)
Shape of y_te:  (123781,)
```

Сл. 17 – Векторизација 3-*gram* репрезентација текстова коришћењем CountVectorizer

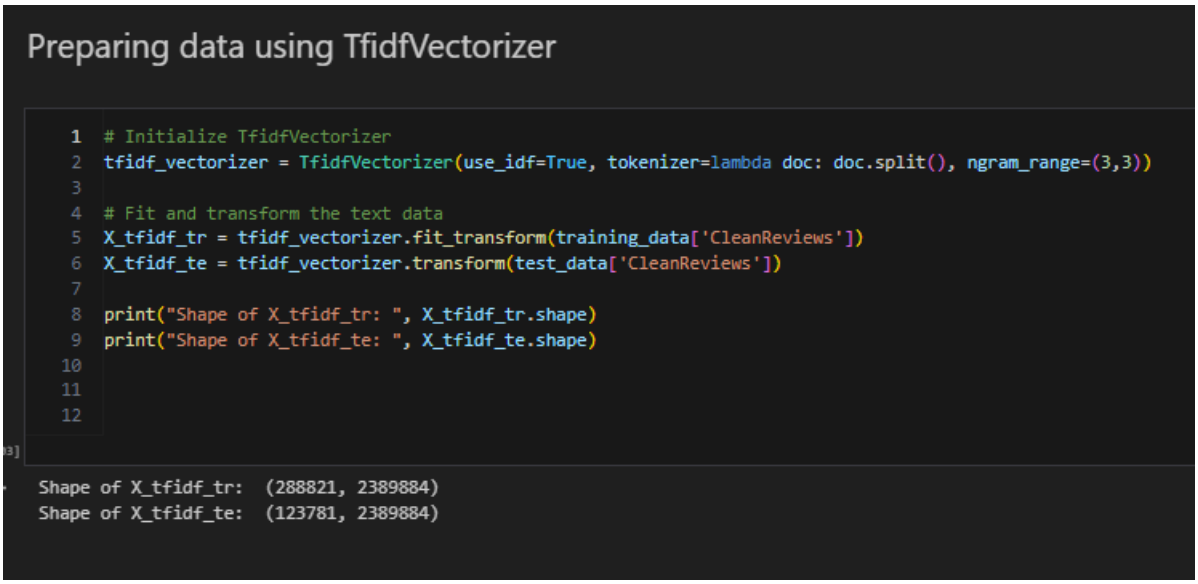
TfifdfTransformer се користи приликом трансформације резултата добијених коришћењем CountVectorizer-а, како би се добили улазни подаци у Tfidf формату.

Preparing data using TfidfTransformer

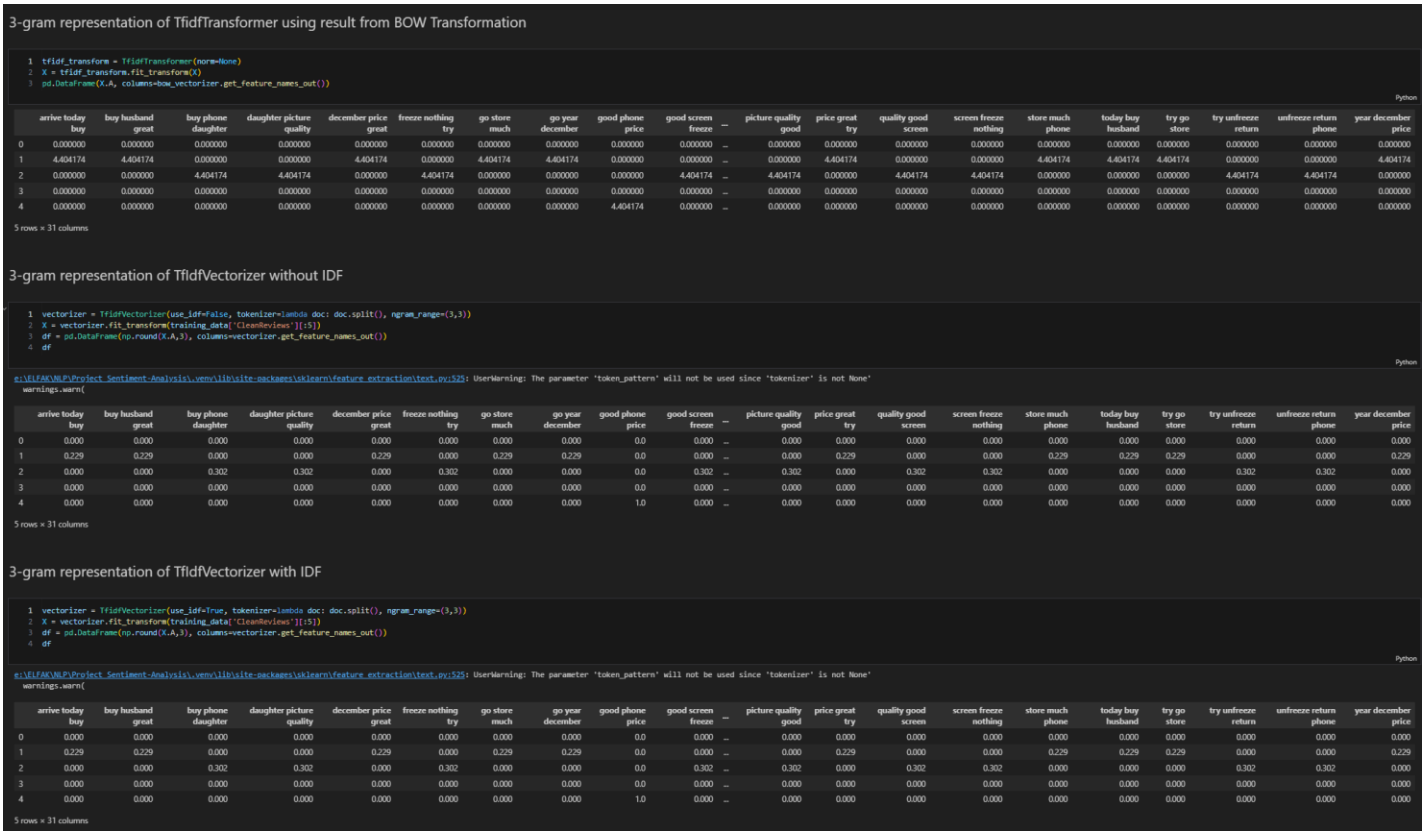
```
1
2  tfidf_transform = TfidfTransformer(norm=None)
3  X_tr_tfidf = tfidf_transform.fit_transform(X_tr_bow)
4  X_te_tfidf = tfidf_transform.transform(X_te_bow)
5
6  print("Shape of X_tr_tfidf: ", X_tr_tfidf.shape)
7  print("Shape of X_te_tfidf: ", X_te_tfidf.shape)
8
```

```
Shape of X_tr_tfidf:  (288821, 2389884)
Shape of X_te_tfidf:  (123781, 2389884)
```

Сл. 18 – Векторизација 3-*gram* репрезентација текстова коришћењем TfidfTransformer-а



Сл. 19 – Векторизација 3-*gram* репрезентација текстова коришћењем TfidfVectorizer-а



Сл. 20 – Примери ретко поседнутих матрица креираних на три различита начина

3.6. Тренирање и евалуација модела

Процес тренирања и евалуције модела представља процес при којем се врши обучавање математичких метода машинског учења или модела базираног на трансформерској архитектури како би се постигла висока прецисизност у процесу евалуације. Пре почетка самог тока процеса тренирања, припремљени подаци се деле на улазне и излазне податке. Сви улазни и излазни подаци се деле на три групе података: тренинг, евалуационе и тест податке.

У нашем случају анализе сентимената улазни подаци представљају припремљене текстуалне рецензије у формату ретко поседнутих матрица уколико тренирамо методу машинском учења, или сами препроцесирани текстови у случају да вршимо евалуацију већ тренираних модела.

Прилагођена метода машинског учења постаје модел за класификацију, као и остали прилагођени и већ постојећи модели на трансформер архитектури.

Евалуација је процес одређивања успешности алгоритма да обави анализу сентимената, тачније да класификује улазни текст у једну од понуђених категорија. Успешности модела се оцењује прецизношћу (*eng. accuracy*) која носи вредност у опсегу од 0 до 100% тачности.

3.6.1. Тренирање метода машинског учења

У току тренирања метода машинског учења примењује се 1-*gram* и 3-*gram* методе припреме података и у два независна тока Jupyter радне свеске се врши прилагођавање (*eng. fitting*) / тренирање и евалуација добијених решења.

У процесу су укључене следеће методе:

- Logistic Regression
- Multinomial Naive Bayes
- XGBCLassifier
- SVC
- Random Forest Classifier

За сваку од метода се примењује прилагођавање на три улазне ретко поседнуте матрице добијене методама:

- Bag of words
- Tf-idf Transformer
- Tf-idf Vectorizer

За сваки од резултата се приказује матрица конфузије и вредност тачности алгоритма. Сви резултати тачности алгоритама се агрегирају и приказују у коначном прегледу резултата.

На следећим сликама је приказан пример изворног кода прилагођавања неких од метода, као и резултати тачности метода.

Logistic Regression

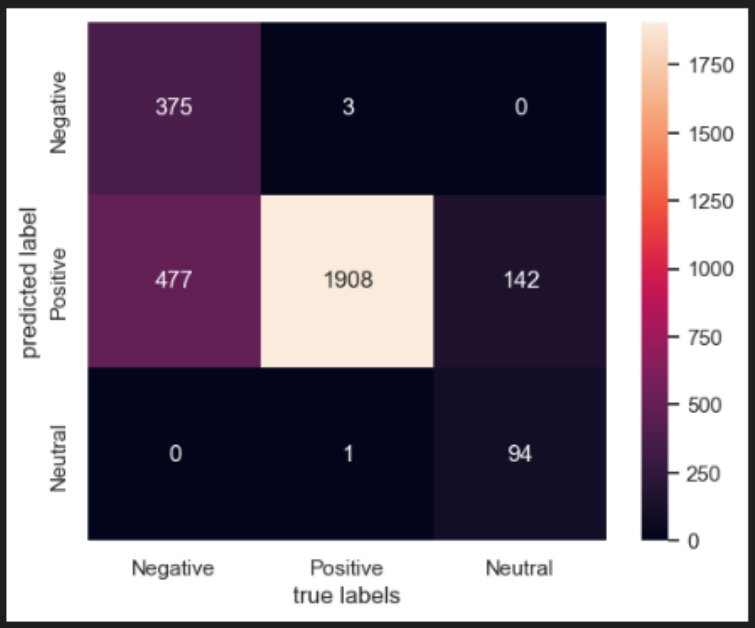
+ Code + Markdown

```
model_name = "LogisticRegression_3_gram"
model = LogisticRegression(max_iter=500) # C=1.0, random_state=0, max_iter=1000)

dataframe_bow = simple_logistic_classify(model, f'{model_name}_{BOW_METHOD_STRING}', X_tr_bow, y_tr, X_te_bow, y_te, BOW_METHOD_STRING)
dataframe_tfidf = simple_logistic_classify(model, f'{model_name}_{TF_IDF_TRANSFORMER_METHOD_STRING}', X_tr_tfidf, y_tr, X_te_tfidf, y_te, TF_IDF_TRANSFORMER_METHOD_STRING)
dataframe_tfidf = simple_logistic_classify(model, f'{model_name}_{TF_IDF_VECTORIZER_METHOD_STRING}', X_tfidf_tr, y_tr, X_tfidf_te, y_te, TF_IDF_VECTORIZER_METHOD_STRING)

data_table_logistic_regression_3_gram = pd.concat([dataframe_bow, dataframe_tfidf, dataframe_tfidf], ignore_index=True)
```

✓ 2.9s

Сл. 21 – Прилагођавање логистичке регресије на податке припремљене 3-*gram* методом

```
Accuracy: 0.7923333333333333
Classification report :
              precision    recall  f1-score   support

    Negative       0.99      0.44      0.61       852
     Neutral       0.99      0.40      0.57       236
    Positive       0.76      1.00      0.86      1912

    accuracy                           0.79      3000
   macro avg       0.91      0.61      0.68      3000
weighted avg       0.84      0.79      0.77      3000
```




```
**************************************************
Accuracy: 0.802
Classification report :
              precision    recall  f1-score   support
```

Сл. 22 – Евалуација резултата логистичке регресије

Укупно је урађено 30 прилагођавања са 5 метода машинског учења, са 2 методе припремљених улазних података (1-*gram* и 3-*gram*) и са 3 методе векторизације података (BOW, Tf-idf Transformer, Tf-idf Vectorizer).

Коначни резултати тачности свих прилагођавања су следећи:

	model	accuracy
0	LogisticRegression_1_gram_bow	0.877667
1	LogisticRegression_1_gram_tf-idf-transformer	0.883667
2	LogisticRegression_1_gram_tf-idf-vectorizer	0.854333
3	MultinomialNB_1_gram_bow	0.844333
4	MultinomialNB_1_gram_tf-idf-transformer	0.827000
5	MultinomialNB_1_gram_tf-idf-vectorizer	0.799333
6	XGBClassifier_1_gram_bow	0.870000
7	XGBClassifier_1_gram_tf-idf-transformer	0.870000
8	XGBClassifier_1_gram_tf-idf-vectorizer	0.879333
9	SVC_1_gram_bow	0.875000
10	SVC_1_gram_tf-idf-transformer	0.866667
11	SVC_1_gram_tf-idf-vectorizer	0.874000
12	RandomForestClassifier_1_gram_bow	0.898667
13	RandomForestClassifier_1_gram_tf-idf-transformer	0.898667
14	RandomForestClassifier_1_gram_tf-idf-vectorizer	0.899333
15	LogisticRegression_3_gram_bow	0.792333
16	LogisticRegression_3_gram_tf-idf-transformer	0.802000
17	LogisticRegression_3_gram_tf-idf-vectorizer	0.758000
18	MultinomialNB_3_gram_bow	0.826667
19	MultinomialNB_3_gram_tf-idf-transformer	0.809000
20	MultinomialNB_3_gram_tf-idf-vectorizer	0.761667
21	XGBClassifier_3_gram_bow	0.685333
22	XGBClassifier_3_gram_tf-idf-transformer	0.685333
23	XGBClassifier_3_gram_tf-idf-vectorizer	0.683667
24	SVC_3_gram_bow	0.791667
25	SVC_3_gram_tf-idf-transformer	0.793667
26	SVC_3_gram_tf-idf-vectorizer	0.786000
27	RandomForestClassifier_3_gram_bow	0.805667
28	RandomForestClassifier_3_gram_tf-idf-transformer	0.805667
29	RandomForestClassifier_3_gram_tf-idf-vectorizer	0.805000

Сл. 23 – Евалуација свих резултата тачности

Из добијених резултата може се уочити да је најбоље резултате класификације анализе сентимената постигао ***Random Forest Classifier*** прилагођен над 1-*gram* припремљеним подацима при чему су за сва три типа векторизације резултати приближно симетрични.

3.6.2. Тренирање и евалуација модела заснованих на BERT архитектури

***Finetuning* модела**

У процесу тренирања и евалуације модела заснованих на BERT архитектури, извршен је *finetuning* претренираног модела ***bert-base-uncased*** при чему је процес трајао ~4 часова. У овом процесу се врши прилагођавање већ постојећег модела за специфичан задатак као што је анализа сентимената.

Скуп података се састајао од укупно 10000 података, при чему је 8000 података коришћено за тренирање модела, 1000 података за евалуацију и 1000 за тестирање модела. Скуп података је потребно преформатирату у *DatasetDict* формат.

```
# Convert Pandas DataFrame to Hugging Face Dataset format
from datasets import Dataset, DatasetDict

dataset_dict = DatasetDict({

    "train": Dataset.from_dict({
        "text": training_data['CleanReviews'].tolist(),
        "label": training_data['Label'].tolist(),  # Include this line if you have labels
    }),
    "test": Dataset.from_dict({
        "text": test_data['CleanReviews'].tolist(),
        "label": test_data['Label'].tolist(),  # Include this line if you have labels
    }),
    "validation": Dataset.from_dict({
        "text": validation_data['CleanReviews'].tolist(),
        "label": validation_data['Label'].tolist(),  # Include this line if you have labels
    }),
    })

dataset_dict
```

✓ 0.0s

```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 8000
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 1000
    })
    validation: Dataset({
        features: ['text', 'label'],
        num_rows: 1000
    })
})
```

Сл. 24 – Промена формата скупа података

За сам процес *finetuning* фазе, користи се *tranformers* библиотека при чему се из ње учитава модел претрениран за класификацију секвенце уз помоћ *AutoModelForSequenceClassification* класе.

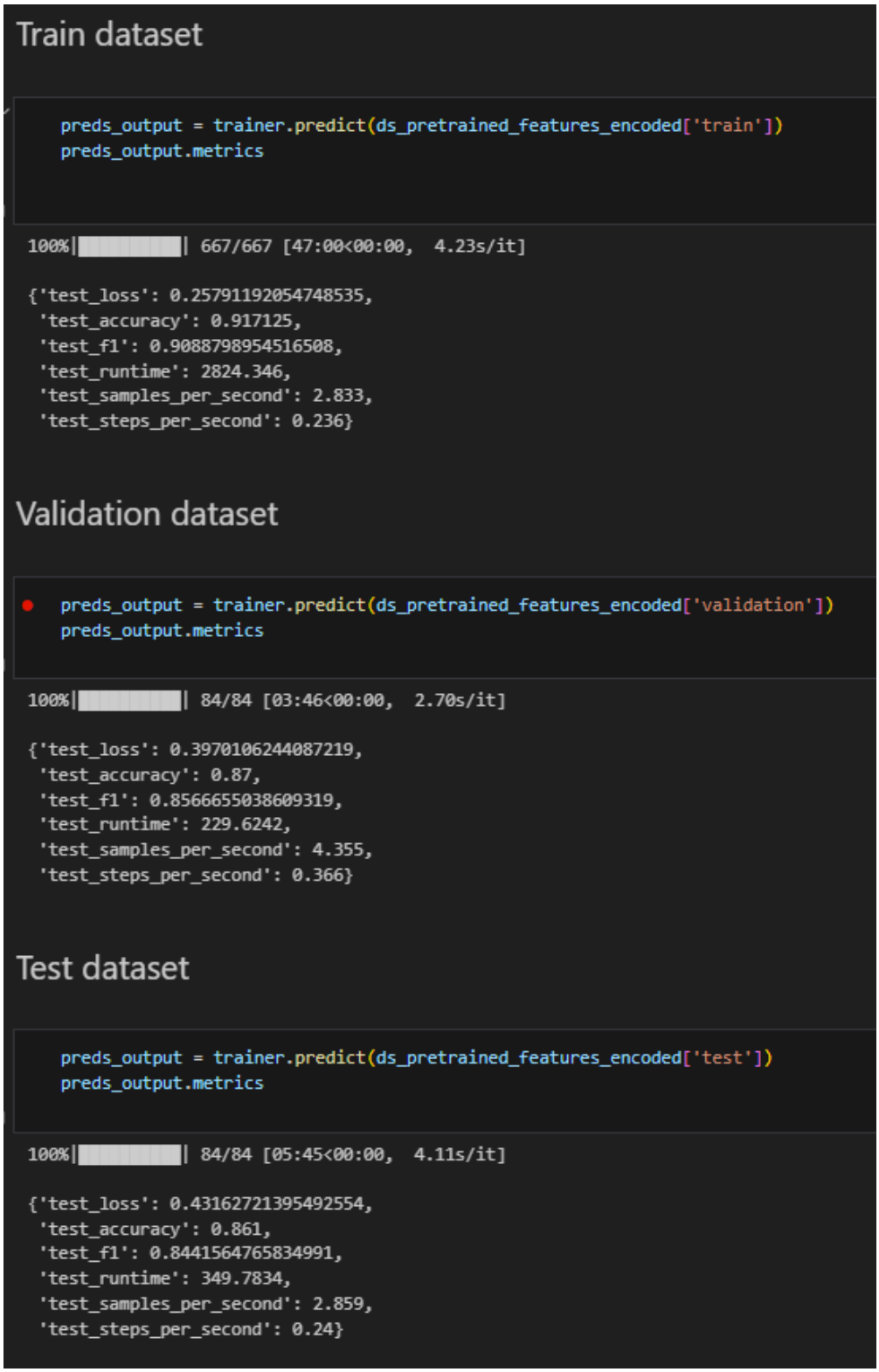
Након учитавања основног *checkpoint*-а, дефинишу се аргументи уз помоћ *TrainingArguments* класе. У тренинг објекат који је типа *Trainer* се прослеђују сви неопходни параметри и започиње фаза дотренирања.

На следећој слици се може видети изворни код:

```
from transformers import AutoModelForSequenceClassification
from transformers import Trainer, TrainingArguments

num_labels = len(df_embedded_from_hidden['label'].unique())

model = (AutoModelForSequenceClassification.from_pretrained(model_ckpt, num_labels=num_labels ).to(device) )

from sklearn.metrics import accuracy_score, f1_score

def compute_metric(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    f1 = f1_score(labels, preds, average = 'weighted')
    acc = accuracy_score(labels, preds)
    return {'accuracy': acc, 'f1': f1}

batch_size = 12

logging_steps = len(ds_pretrained_features_encoded['train']) // batch_size

model_name = f"{model_ckpt}-finetune"

training_args = TrainingArguments(output_dir = model_name,
                                  num_train_epochs=2,
                                  learning_rate=2e-5,
                                  per_device_train_batch_size=batch_size,
                                  per_device_eval_batch_size=batch_size,
                                  weight_decay=0.01,
                                  evaluation_strategy="epoch",
                                  disable_tqdm=False,
                                  logging_steps=logging_steps,
                                  push_to_hub=False,
                                  log_level="error"
                                  )


trainer = Trainer(model=model, args=training_args, compute_metrics=compute_metric,
                  train_dataset = ds_pretrained_features_encoded['train'],
                  eval_dataset = ds_pretrained_features_encoded["validation"],
                  tokenizer = tokenizer
                  )
trainer.train()
```

Сл. 25 – Изворни код за тренирање, тј. *finetuning* у овом случају

На следећој слици се могу видети резултати евалуације овог модела након *finetuning* фазе над тренинг, евалуационим и тест подацима:



Сл. 26 – Резултати модела

Коришћење претренираних модела за евалуацију

У процесу коришћења модела базираних на BERT архитектури, могуће је искористити постојеће тренирање моделе над другим идентичним или прибложно идентичним скуповима података за анализу сентимената. У овом делу практичног рада су коришћени следећи модели преузети са *HuggingFace* сајта:

- nlptown/bert-base-multilingual-uncased-sentiment
- cardiffnlp/twitter-roberta-base-sentiment
- cardiffnlp/twitter-roberta-base-sentiment-latest
- distilbert-base-uncased-finetuned-sst-2-english

Треба напоменути да је код евалуације ових модела неопходно применити мапирање резултата предикције датог модела на резултате улазног скупа података како би било могуће израчунати тачност модела.

Сл. 27 – Пример мапирања и евалуације претренираних модела

Након евалуације свих претренираних модела, добија се следећи резултат:

	model	accuracy
0	nlptown/bert-base-multilingual-uncased-sentiment	0.746000
1	cardiffnlp/twitter-roberta-base-sentiment	0.631900
2	cardiffnlp/twitter-roberta-base-sentiment-latest	0.678700
3	distilbert-base-uncased-finetuned-sst-2-english	0.737324

Сл. 28 – Тачност претренираних модела

Из ових резултата, могуће је претпоставити да се доста разликују скупови података над којима су обучавани наведени модели у односу на скуп података који се обрађује у овом раду те стога и резултати нису задовољавајући у овим примерима. Такође, неки од ових модела имају другачији систем оцењивања сентимента (пр. од 1 до 5) те стога и мапирање може додатно утицати на резултате ове евалуације.

3.6.3. Коначна оцена евалуације

Коначна оцена евалуације се може извести из свих укупних резултата тако да се тражи најбоља прецизност класификације сентимената. Из досадашњих тестирања, најбоље резултате је имала метода машинског учења ***Random Forest Classifier*** прилагођен над 1-*gram* припремљеним подацима са Tf-idf типом векторизације са прецизношћу **0.89%** на тест подацима.

Могуће унапређење и бољи резултати се могу очекивати са повећањем скупа података и код основних математичких метода машинског учења попут *Random Forest Classifier*, као и код процеса *Finetuning*-а модела. Због недостатка хардверских ресурса, није било могуће извршити додатно побољшање резултата класификације.

4. Закључак

У току израде овог пројекта, упознао сам се великим бројем концепата који обухватају обраду природних језика, различитим методама и моделима. Област обраде природних језика се убрзано развија у последње време и управо зато представља додатни изазов за проучавање у овој области.

Упознавање са различитим методама препроцесирања, визуализације, векторизације и евалуације модела је представљало занимљив поступак темељног истраживања концепата везаних за поменуту област. Обзиром да је ово само један начин од могућих примена обраде природних језика, тек након оваквог упознавања могуће је схватити обиме развоја области као што је обрада природних језика.

Гледајући у будућност и убрзани развој вештачке интелигенције, можемо само наслутити домет који се може достићи примењујући и развијајући овакве технологије.

5. Литература

1. https://huggingface.co/blog/sentiment-analysis-python
2. https://brighteshun.medium.com/sentiment-analysis-part-1-finetuning-and-hosting-a-text-classification-model-on-huggingface-9d6da6fd856b
3. https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/04-n-grams
4. https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/05-tokenization-stemming-lemmatization
5. https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/06-classify-articles
6. https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/09-text-as-vectors-bow-tfidf
7. https://www.nlplanet.org/course-practical-nlp/01-intro-to-nlp/13-classify-articles-embeddings
8. https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment
9. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment
10. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest
11. https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english
12. https://www.kaggle.com/datasets/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones?resource=download