DLBAIPNLP01 – PROJECT: NLP

STUDENT: HUGO ALBUQUERQUE COSME DA SILVA

ID:92126125

TABLE OF CONTENTS

1 – INTRODUCTION

Sentiment analysis, also known as opinion mining, is a technique that uses natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, and quantify affective states and subjective information. In simple terms, it is a method of understanding the emotions behind a piece of text. Sentiment analysis is important because it helps

businesses and organizations understand the emotions of their customers in their marketing efforts. It is useful for identifying products and brands, customer loyalty, customer satisfaction, the effectiveness of marketing and advertising, and product uptake. By analyzing customer feedback and understanding their needs, businesses can improve their products and services to better meet customer expectations.

In the entertainment industry, Sentiment analysis is an important tool for analyzing movie reviews. It can help determine the overall sentiment of a review, whether it is positive, negative, or neutral. This information can be used to gauge the public's opinion of a movie and can be helpful for movie studios, critics, and audiences alike.

In this project, the goal is to develop a system that will classify movie reviews as having positive sentiment or negative sentiments. More specifically, a dense neural network will be trained on reviews coming from the Stanford Large Movie Review Dataset.

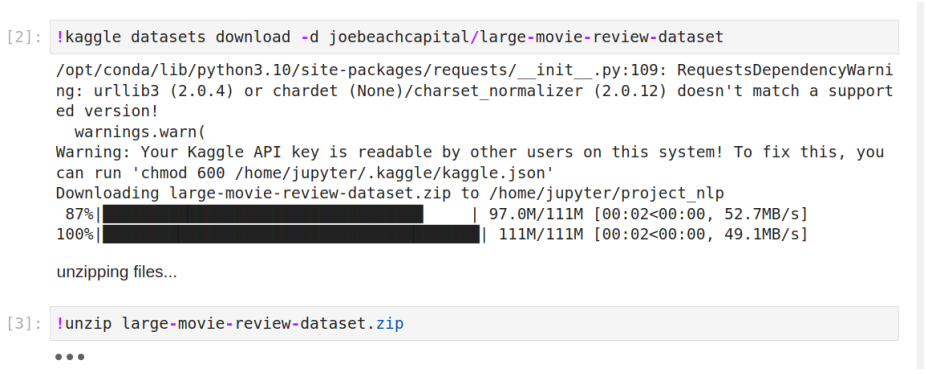
2 - METHODOLOGY

2.1 - DATASET

The Stanford Large Movie review dataset is used throughout this project. It contains 50,000 highly polar movie reviews, 25000 in the training set, and 25000 in the test set. It is necessary to process the dataset files so they fit nicely in a panda's Dataframe tabular format.

The dataset is downloaded from Kaggle:

Fig. 1 - Downloading datasets from Kaggle



There are 25000 reviews in the training set. The dataset is balanced: 12500 positive samples, 12500 negative samples:

Fig. 2 - First Impressions training set

```
[4]: train_pos_reviews = os.listdir('aclImdb/train/pos/')
     print(f'Number of positive reviews: {len(train_pos_reviews)}')
     train_neg_reviews = os.listdir('aclImdb/train/neg/')
     print(f'Number of negative reviews: {len(train_neg_reviews)}')
     train_reviews = train_neg_reviews + train_pos_reviews
     print(f'Training data size: {len(train_reviews)}')

     Number of positive reviews: 12500
     Number of negative reviews: 12500
     Training data size: 25000
```

There are 25000 reviews in the test set. The dataset is balanced: 12500 positive samples, 12500 negative samples:

Fig. 3 - First impressions testing set

```
[5]: test_pos_reviews = os.listdir('aclImdb/test/pos/')
     print(f'Number of positive reviews: {len(test_pos_reviews)}')
     test_neg_reviews = os.listdir('aclImdb/test/neg/')
     print(f'Number of negative reviews: {len(test_neg_reviews)}')
     test_reviews = test_neg_reviews + test_pos_reviews
     print(f'testing data size: {len(test_reviews)}')

     Number of positive reviews: 12500
     Number of negative reviews: 12500
     testing data size: 25000
```

Processing input files:

Fig. 4 - Processing input files

```
train_pos_base = 'aclImdb/train/pos/'
train_neg_base = 'aclImdb/train/neg/'
test_pos_base = 'aclImdb/test/pos/'
test_neg_base = 'aclImdb/test/neg/'

records = []
for name in train_pos_reviews:
    file_path = os.path.join(train_pos_base, name)
    text = open(file_path, 'r').read()
    records.append(['train', text, 1])

for name in train_neg_reviews:
    file_path = os.path.join(train_neg_base, name)
    text = open(file_path, 'r').read()
    records.append(['train', text, 0])

for name in test_pos_reviews:
    file_path = os.path.join(test_pos_base, name)
    text = open(file_path, 'r').read()
    records.append(['test', text, 1])

for name in test_neg_reviews:
    file_path = os.path.join(test_neg_base, name)
    text = open(file_path, 'r').read()
    records.append(['test', text, 0])

data = pd.DataFrame(records, columns=['subset', 'review', 'polarity'])
data.head()
```

	subset	review	polarity
0	train	A very silly movie, this starts with a soft po...	1
1	train	'Five Days' is billed as something special, a ...	1
2	train	This is one of the first independent movies I'...	1
3	train	A good film, and one I'll watch a number of ti...	1
4	train	Okay, that was a pretty damn good episode. Muc...	1

2.2 - Preprocessing Steps

After the data is in a dataframe, it is necessary to preprocess the reviews into a numerical format that is understandable by our model. In this project, the library Spacy will be used to perform the processing steps. In order to use **Spacy**, it is necessary to install the package and to install one of the available language models. Since the movie reviews are in English, and the highest possible accuracy is desirable, I have chosen the ***en_core_web_lg*** model.

Fig. 5 - Spacy language model installation

```
27]: # installing spacy's english language model (large version)
     !python -m spacy download en_core_web_lg
```

Another important preprocessing step will be the conversion of the reviews from text into vector embeddings. HuggingFace has a publicly available library called sentence-transformers. It has a range of pre-trained models that can be used very easily after they are downloaded. I will use a model called ***all-mpnet-base-v2.*** It maps sentences & paragraphs to a 768-dimensional dense vector space and can be used for tasks like clustering or semantic search.

Fig. 6 - Loading model from Hugging Face

```
# Loading pre-trained models from hugging-face
model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')
```

Spacy has a really intuitive interface. So, although small, a lot of things happening in the following function:

Fig. 7 - Preprocessing function

```
.7]: def preprocess_text(text):
         '''
         Function that will tokenize the input text, remove stopwords, remove non alpha-numeric tokens and return the lowered v
         '''
         doc = nlp(text)
         tokens = [token.lemma_.lower() for token in doc if not token.is_stop and token.is_alpha]
         return ' '.join(tokens)
```

Preprocessing steps:

1. Tokenization: ***doc = nlp(text)***
2. stop words removal: ***if not token.is_stop***
3. removing tokens that are not comprised of alphanumerical characters: ***token.is_alpha***
4. Lemmatization: ***token.lemma_***
5. lowercasing: ***token.lower()***

We apply this function to each movie review with the help of the pandas *progress_apply* function:

Fig. 8 - Applying preprocessing function

```
: data['parsed_review'] = data['review'].progress_apply(preprocess_text)
```

The next step is to generate the paragraph's embeddings from the parsed_revie column. This step take a while to complete, but the code to perform it is really simple:

Fig. 9 - Generating embeddings

```
[5]: # Loading pre-trained models from hugging-face
     model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')

[ ]: data['review-embeddings'] = data['parsed_review'].progress_apply(model.encode)
```

After everything that has been done so far, the data looks like this:

Fig. 10 - Processed data

```
[14]: data.head()
```

[14]:

	subset	review	polarity	parsed_review	review-embeddings
0	train	A very silly movie, this starts with a soft po...	1	silly movie start soft porn sequence venture f...	[0.023642657, 0.011921609, 0.003747118, -0.025...
1	train	'Five Days' is billed as something special, a ...	1	day bill special crime drama consist series ep...	[0.009517353, 0.051944733, 0.019322842, -0.020...
2	train	This is one of the first independent movies I'...	1	independent movie see low budget insomniac sym...	[0.023109218, 0.037633322, 0.019782882, -0.015...
3	train	A good film, and one I'll watch a number of ti...	1	good film watch number time rich previous righ...	[0.024958918, 0.03477856, 0.015119431, -0.0285...
4	train	Okay, that was a pretty damn good episode. Muc...	1	okay pretty damn good episode well credit came...	[-0.0037331479, 0.017467482, 0.039064273, -0.0...

It is now necessary to split the data into train and test splits so we can properly train the model:

Fig 11 - Train test split

```
[54]: train_data = data[data['subset'] == 'train']
      X_train, y_train = train_data['review-embeddings'].values, train_data['polarity'].values
      test_data = data[data['subset'] == 'test']
      X_test, y_test = test_data['review-embeddings'].values, test_data['polarity'].values
```

The final step is to reshape both train and test data so the datasets are in the format *(num_training_examples, num_embedding_dimensions)*

Fig. 12 - Reshaping embeddings

```
[56]: def reshape_embeddings(data: np.array):
          '''
          convert embeddings from nested numpy representations (original shape: (25000, ))
          to numpy array with embedding dimensions in axis 1 (final shape (25000, 768))
          '''
          reshaped_data = []
          for idx in range(len(data)):
              reshaped_data.append(data[idx].reshape(768).tolist())
          return np.array(reshaped_data)

[58]: X_train = reshape_embeddings(X_train)
      X_test = reshape_embeddings(X_test)

      X_train.shape, X_test.shape

[58]: ((25000, 768), (25000, 768))

[59]: y_train.shape, y_test.shape

[59]: ((25000,), (25000,))
```

2.3 - Model Training

After some research and given my past experience with similar types of problems, I have decided to experiment with a Keras Dense Neural Network. After analyzing how the number of layers and neurons affects the overall performance, I have settled for the following architecture:

Fig. 13 - Neural network architecture

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 128)               98432

 dense_10 (Dense)            (None, 64)                8256

 dense_11 (Dense)            (None, 1)                 65

=================================================================
Total params: 106753 (417.00 KB)
Trainable params: 106753 (417.00 KB)
Non-trainable params: 0 (0.00 Byte)
```

Some important remarks:

The first layer has input dimensions similar to the embedding dimensions. Since this is a binary classification problem, using a ***sigmoid activation*** in the last layer and ***binary_crossentropy*** for the loss function are appropriate choices.

I have set two callbacks:

- EarlyStopping to avoid overfitting
- ReduceLROnPlateau for trying to escape local minima

Initially, I trained the model for 10 epochs and used 16 for batch size to get a first notion of how loss optimization progresses.

Fig. 14 - Model Training

```
]: early_stopping = EarlyStopping(monitor='val_loss', patience=10)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5)

history = model.fit(X_train, y_train,
                    epochs=10,
                    batch_size=16,
                    verbose=2,
                    validation_data=[X_test, y_test],
                    callbacks=[early_stopping, reduce_lr])
```

Fig. 15 - Training and validation Loss

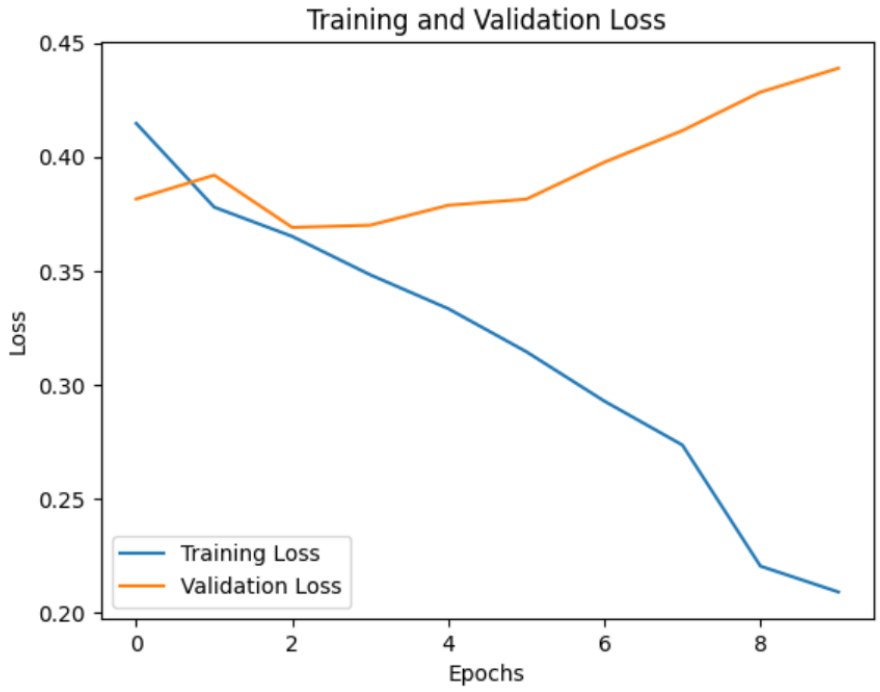
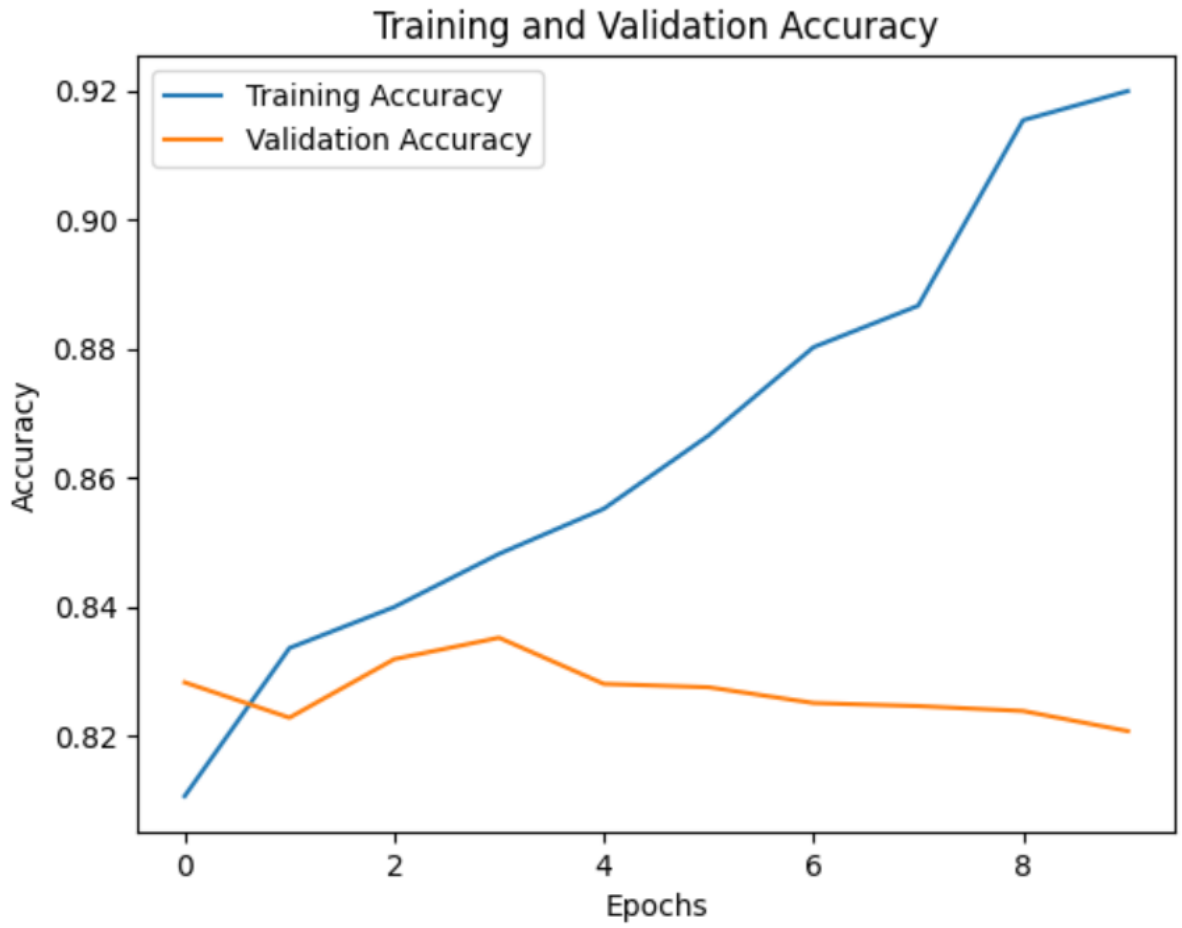


Fig. 16 - Training and validation Accuracy



Looking at the above graphs, it is possible to realize that, although the training loss and accuracy keep improving, the validation accuracy starts to decrease, and the validation loss starts to increase. This is a problem called overfitting. To address this issue, the model was once again compiled and trained only for 3 epochs:

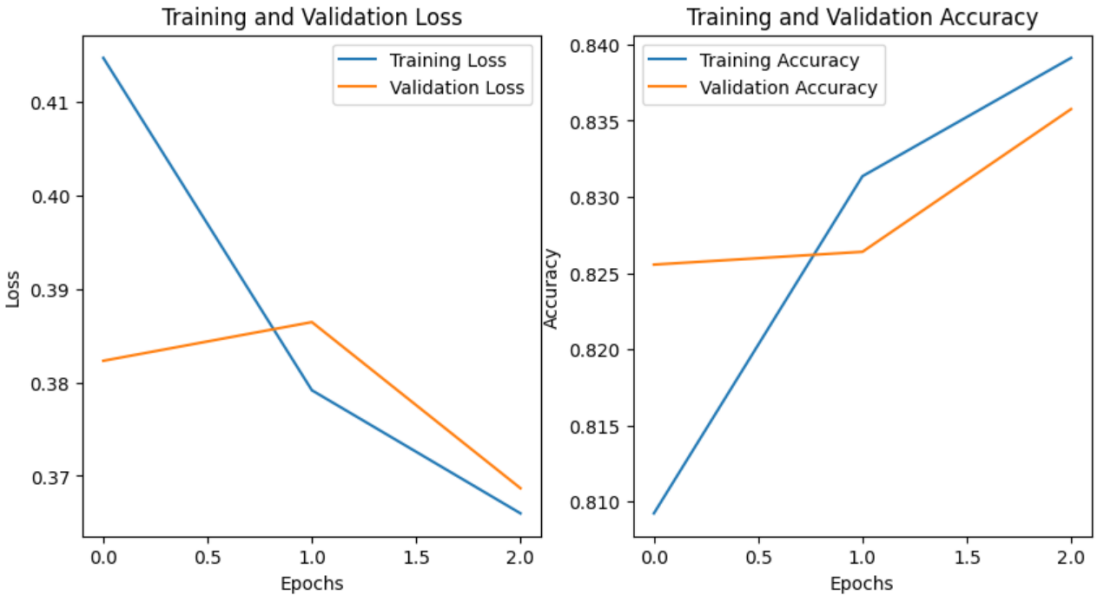
Fig. 17 - Training and validation Accuracy

```
[77]: model = Sequential()
model.add(Dense(128, input_dim=768, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    epochs=3,
                    batch_size=16,
                    verbose=2,
                    validation_data=[X_test, y_test])
```

```
Epoch 1/3
1563/1563 - 4s - loss: 0.4147 - accuracy: 0.8092 - val_loss: 0.3823 - val_accuracy: 0.8256 - 4s/epoch - 2ms/step
Epoch 2/3
1563/1563 - 3s - loss: 0.3791 - accuracy: 0.8314 - val_loss: 0.3864 - val_accuracy: 0.8264 - 3s/epoch - 2ms/step
Epoch 3/3
1563/1563 - 3s - loss: 0.3660 - accuracy: 0.8391 - val_loss: 0.3687 - val_accuracy: 0.8358 - 3s/epoch - 2ms/step
```

Fig. 18 - Final training performance



Note that this model was able to achieve a higher accuracy on the test_set.

2.4 - Model Evaluation

The model evaluation metrics are as follows:

Fig. 19 - Model evaluation metrics

```
print(f'The model accuracy is: {accuracy_score(y_test, y_pred)}')
print(f'Confusion matrix:\n', confusion_matrix(y_test, y_pred))
print(f'Classification report:\n', classification_report(y_test, y_pred))
```

```
The model accuracy is: 0.83576
Confusion matrix:
 [[10199  2301]
 [ 1805 10695]]
Classification report:
              precision    recall  f1-score   support

           0       0.85      0.82      0.83     12500
           1       0.82      0.86      0.84     12500

    accuracy                           0.84     25000
   macro avg       0.84      0.84      0.84     25000
weighted avg       0.84      0.84      0.84     25000
```

The overall accuracy is 83.6 %, the number of incorrect predictions does not seem too biased towards a certain sentiment classification, and the f1_scores for each class are practically the same.

Let's take a look at both correct and incorrect predictions to see whether what we see supports the evaluation metrics:

Fig. 20 - correct predictions

49179	It was (foolishly) with some degree of relish that I sat down to watch what a friend had promised would be the worst/best movie experience of my life, the mighty 'Roller blade 7'. 2 years on and I'm still in therapy. Oh yes my dear friends it REALLY IS THAT BAD. They obviously got about 40 minutes of footage in the can and then decided to use said footage endlessly and repeatedly to brain-numbing effect. My only fear of the kind of post-apocalyptic world featured in this turkey is that somehow, some way, a print of this abomination would survive. Truly the living would envy the dead.	0

37320	Don't listen to what the critics have always said about this cute, charming little movie. Madonna is GREAT in this clever comedy. I worked at a video store for several years and suggested this movie to lots of customers- no one EVER brought it back and screamed at me for telling them to rent it. Everyone always enjoyed it. It's actually a great movie for kids, too.	1

Fig. 21 - Incorrect predictions

30208	A middle aged man, Robert Jordan, set in his ways, takes on a boy scout troop after his predecessor leaves under duress. Jordan takes on the pack mostly to learn what the boys like so he can revive his flagging radio program which is losing it's appeal to the younger set. He has a rough go at first with the boys, especially so with Mike, an 8 year old who forms an attachment for the older man which is anything but reciprocated. Do things work out for Jordan and the scouts? Check out this entertaining and amusing film from the old days.	0
39128	I gotta admit it, I love horror films...especially 80s slasher films. Hell, I even love cheese like Sleepaway Camp and Night of the Demons. But, I didn't think much of this movie. The death scenes weren't very well done, the CGI was terrible, and the acting was ho-hum. Worst of all was the story which didn't make sense at all. I'd say save your money but chances are, if you want to see this movie...you're going to anyway. I didn't hate it...it's just not very good. Overall, it's just another bland, lifeless horror film that lacks life (it's no surprise that this one was on the shelf at Dimension for over a year after it was completed).	1

3 - Conclusion

In this project, a sentiment analysis classifier was built. It outputs 1 for positive sentiment and 0 for negative sentiment. The movie reviews are preprocessed and then converted into sentence embeddings. The embeddings are used as inputs to a Dense Neural Network built using Keras. The classifier achieved 84% accuracy.

Given that Sentiment Analysis is a subjective task, which means that accuracy scores close to 100% are not achievable, and the simplicity of the model, I close this project with a very positive sentiment ;-) (Pun intended.)

4 - BIBLIOGRAPHY

HUALCOSA. (2023). GITHUB - HUALCOSA/DLBAIPNLP01_PROJECT_NLP. GITHUB.

HTTPS://GITHUB.COM/HUALCOSA/DLBAIPNLP01_PROJECT_NLP

PYTHON, R. (2023). NATURAL LANGUAGE PROCESSING WITH SPACY IN PYTHON. REALPYTHON.COM. HTTPS://REALPYTHON.COM/NATURAL-LANGUAGE-PROCESSING-SPACY-PYTHON/

SENTENCE-TRANSFORMERS/ALL-MPNET-BASE-V2 · HUGGING FACE. (2001, JUNE 6). HTTPS://HUGGINGFACE.CO/SENTENCE-TRANSFORMERS/ALL-MPNET-BASE-V2

JURAFSKY, D. & MARTIN, J. (2013). SPEECH AND LANGUAGE PROCESSING [ELECTRONIC RESOURCE]: AN INTRODUCTION TO NATURAL

LANGUAGE PROCESSING, COMPUTATIONAL LINGUISTICS, AND SPEECH RECOGNITION (2ND ED.). PEARSON PRENTICE HALL.

LANE, H., HAPKE, H. M., & HOWARD, C. (2019). NATURAL LANGUAGE PROCESSING IN ACTION [ELECTRONIC RESOURCE]:

UNDERSTANDING, ANALYZING, AND GENERATING TEXT WITH PYTHON. MANNING.

PARAMESHA, K., GURURAJ, H. L., NAYYAR, A., & RAVISHANKAR, K. C. (2023). SENTIMENT ANALYSIS ON CROSS-DOMAIN

TEXTUAL DATA USING CLASSICAL AND DEEP LEARNING APPROACHES. MULTIMEDIA TOOLS AND APPLICATIONS: AN INTERNATIONAL JOURNAL, 1–24.