Natural Language Processing for Telugu

By

Anirudh Potturi

Siri Chandana Baddula

Sanjana Killi

Srilekha Bandaru

CSS 581 Machine Learning Term Project Report

ABSTRACT

NLP for Telugu

There is a great deal of demand for Sentiment Analysis in today's age of social media. Data, as we know, is generated in unimaginable sizes worldwide. Most of the data users consume daily is through the internet, and social media is a significant contributor. With so many languages spoken worldwide, English is considered a universal language, and the work done in the field of Natural Language Processing has been researched and frequently developed with a focus on this language. Our efforts are geared towards NLP for Regional and National languages apart from English, where limited work has been done. We focused on NLP for Telugu, a traditional language spoken in the southern states of India – Andhra Pradesh and Telangana. With this work, we hope to contribute work for this language and introduce a systematic process for researchers and students wanting to explore NLP for other languages.

Contents

LIST OF FIGURES

LIST OF TABLES

‘‘‘

Chapter 1

Introduction

Telugu is the 4th most spoken language in India and in the top 20 languages spoken worldwide. With over 75 million speakers, there is a growing demand for research with NLP. With the availability of script characters available on devices these days, people can generate content in Telugu script and share it across platforms. Much of this data often goes unchecked or unfiltered. For instance, a simple search on modern search engines in English produces accurate results but also can detect and display safe results. Similarly, social media sites monitor and detect content that does not meet the community guidelines and take them down. None of this is possible without NLP. With a major focus on developing such models to understand data written in the English script, data in other scripts are not necessarily analyzed in the same fashion. As a result, we often come across content that technically does not meet the guidelines of a media website, but the content was not taken down because of the lack of such systems in place. Our goals for this project were to perform Sentiment Analysis of text in Telugu script and classify the genre. Genre classification was specific to the dataset we used, and Sentiment Analysis is something we believe has many applications. This section explains our work done in both these tracks.

1.1 Data Exploration, Cleaning and Translation

The first step we performed was combining the datasets into a single dataset to simplify the data-cleaning process. The dataset we acquired was available as train and test sets. We identified inconsistencies in the text. This was mainly because the data was collected from Indian news media sites, and understandably, the text in Telugu was formatted differently. Thus, we removed the script formatting characters and later translated all our data into English.

The dataset consists of news articles of 5 different categories - Sports, Business, Editorial, Nation, and Entertainment. From Fig. 1.1 and 1.2, the distribution of the dataset. It is observed that the class “Editorial” is under-represented, and the class “Nation” is over-represented.

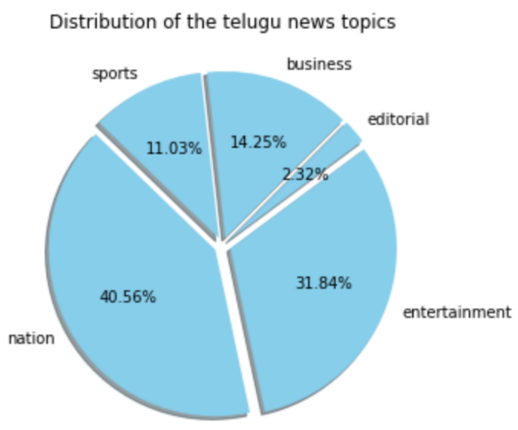


Figure 1.1: Data Distribution 1

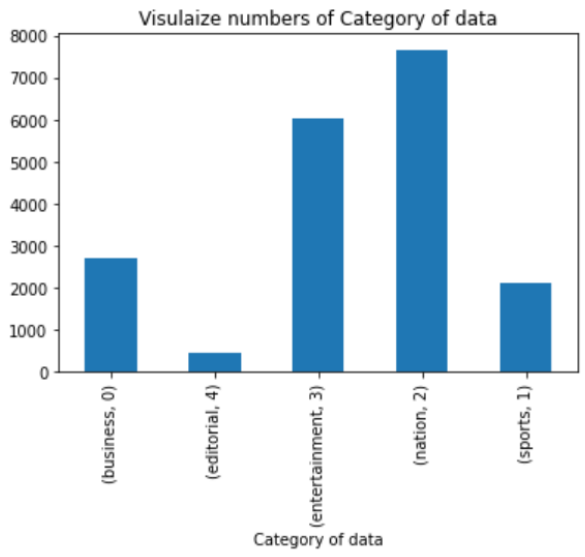


Figure 1.2: Data Distribution 2

From Fig. 1.3, word clouds of text articles from different classes. Word clouds are used to visualize the most frequent or popular phrases or words in a text. The

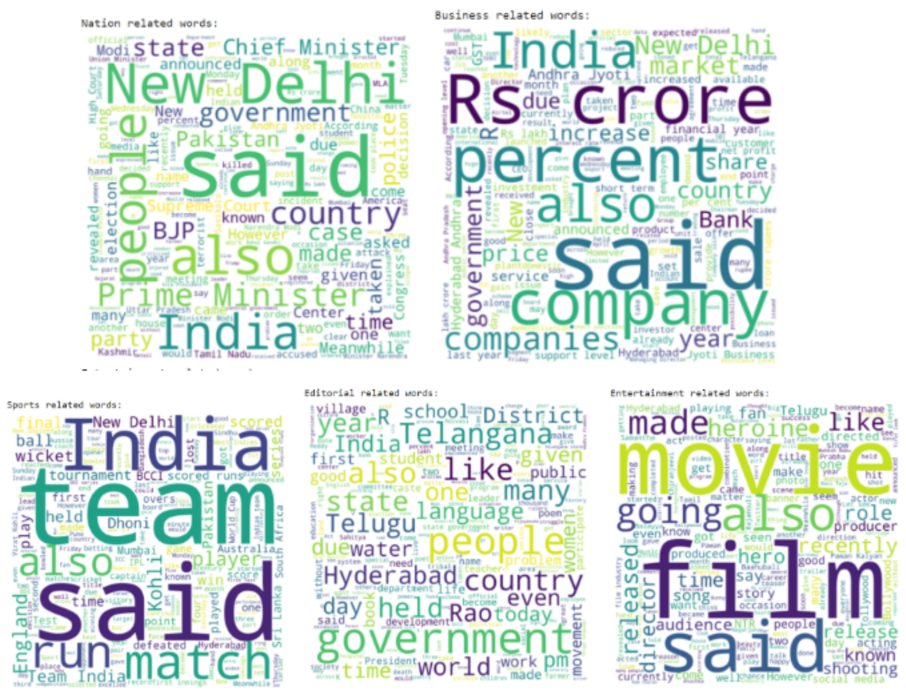
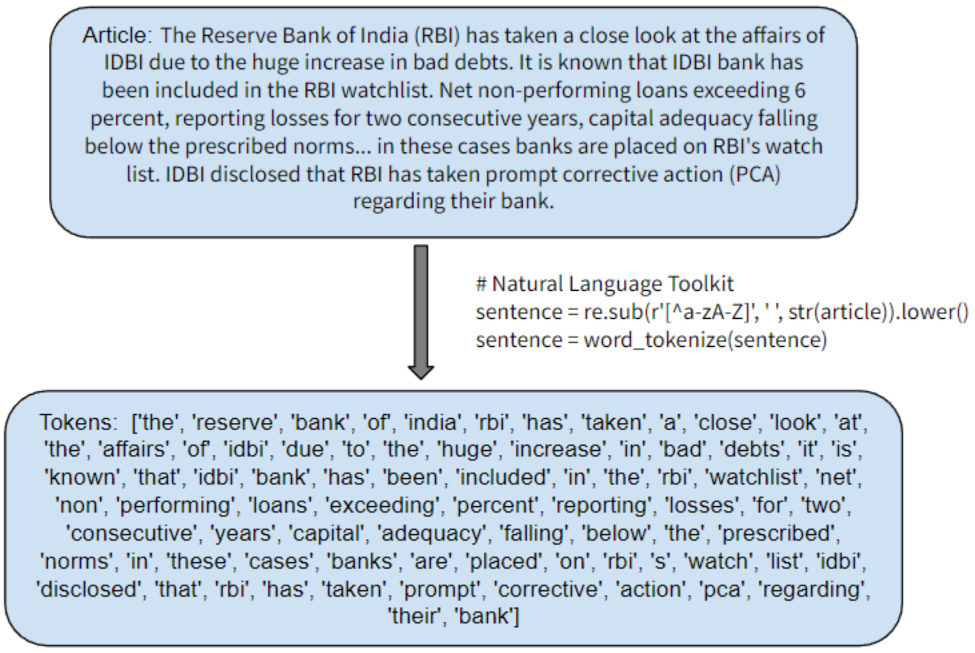


Figure 1.3: Word Clouds

most frequent or popular phrases or words appear bold and large when compared with the words around them.

1.2 Sentiment Scoring and Classification

Sentiment Analysis is a key component of the systems described previously that filter digital content. The first step of this phase was to compute a sentiment score for our data. From various options we explored to do so, we decided upon normalizing the score by the length of the sentence. Essentially, we computed the score by summing the frequency of positive and negative words for each set of news headlines and articles. Lastly, we divide this summation by the total number of words in the entry. This process was performed on each headline and article individually. Note that this process was done on the translated English text with a dictionary of positive and negative words we acquired from openly available resources. From Fig. 1.3 and 1.3, each piece of data is tokenized. After tokenization, stop words, punctuations and numbers are eliminated. Stop words



Article: The Reserve Bank of India (RBI) has taken a close look at the affairs of IDBI due to the huge increase in bad debts. It is known that IDBI bank has been included in the RBI watchlist. Net non-performing loans exceeding 6 percent, reporting losses for two consecutive years, capital adequacy falling below the prescribed norms... in these cases banks are placed on RBI's watch list. IDBI disclosed that RBI has taken prompt corrective action (PCA) regarding their bank.

```
# Natural Language Toolkit
sentence = re.sub(r'[^a-zA-Z]', ' ', str(article)).lower()
sentence = word_tokenize(sentence)
```

Tokens: ['the', 'reserve', 'bank', 'of', 'india', 'rbi', 'has', 'taken', 'a', 'close', 'look', 'at', 'the', 'affairs', 'of', 'idbi', 'due', 'to', 'the', 'huge', 'increase', 'in', 'bad', 'debts', 'it', 'is', 'known', 'that', 'idbi', 'bank', 'has', 'been', 'included', 'in', 'the', 'rbi', 'watchlist', 'net', 'non', 'performing', 'loans', 'exceeding', 'percent', 'reporting', 'losses', 'for', 'two', 'consecutive', 'years', 'capital', 'adequacy', 'falling', 'below', 'the', 'prescribed', 'norms', 'in', 'these', 'cases', 'banks', 'are', 'placed', 'on', 'rbi', 's', 'watch', 'list', 'idbi', 'disclosed', 'that', 'rbi', 'has', 'taken', 'prompt', 'corrective', 'action', 'pca', 'regarding', 'their', 'bank']

Figure 1.4: Sentiment Analysis 1

are frequently appearing words which is not necessarily a universal list. In general, the consensus is that stop words do not add meaning to a sentence. Thus, they can be ignored without losing valuable information. Stop word lists/dictionaries change from one library to another.

With the sentiment scores computed for the entire data, we proceeded to prepare the features for our models. Generating TF-IDF or Term Frequency Inverse Document Frequency is a statistical measure we used to compute the occurrence of words across the data. We can think of it as a 2-Dimensional matrix representation of data. With this matrix, we decided to split it into train and test along with the target variable as a positive or negative classification. From the scores we computed, we decided that a negative score is classified as containing negative sentiment (Binary class 0), and a positive score would be positive sentiment (Binary class 1)

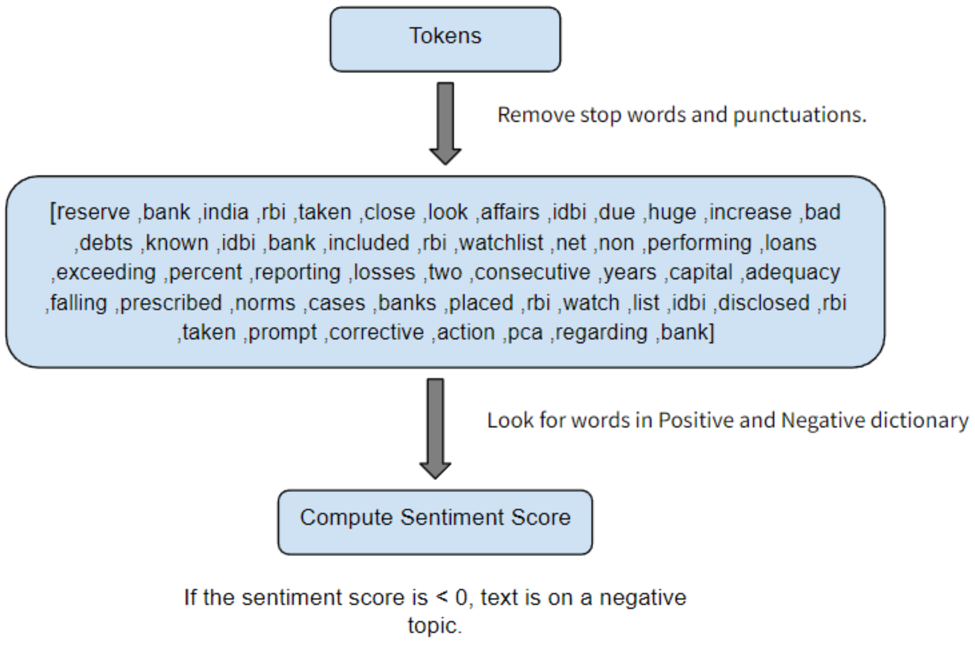

Tokens
Remove stop words and punctuations.
[reserve ,bank ,india ,rbi ,taken ,close ,look ,affairs ,idbi ,due ,huge ,increase ,bad ,debts ,known ,idbi ,bank ,included ,rbi ,watchlist ,net ,non ,performing ,loans ,exceeding ,percent ,reporting ,losses ,two ,consecutive ,years ,capital ,adequacy ,falling ,prescribed ,norms ,cases ,banks ,placed ,rbi ,watch ,list ,idbi ,disclosed ,rbi ,taken ,prompt ,corrective ,action ,pca ,regarding ,bank]
Look for words in Positive and Negative dictionary
Compute Sentiment Score
If the sentiment score is < 0, text is on a negative topic.

Figure 1.5: Sentiment Analysis 2

1.3 Lemmatization and word2vec Classification

For this model, we implemented a lemmatization technique and then classified the text using various classifier models. The text was cleaned by implementing various functions like removing tags, special characters and converting the text to lowercase. Then further, the text was tokenized, and stopwords (such as ‘a’, ‘the’, ‘is’, ‘are’, etc. . .) were removed. Lemmatization was done to group the text back together using WordNetLemmatizer(). We then implemented word2vec model using two different vectorizers, namely TF-IDF and CountVectorizer. A variant of TF-IDF is a technique in which we can find the meaning of sentences consisting of words and cancel out the incapabilities of the Bag of Words technique.

1.4 Classification of Telugu Text using n-gram analysis

There were five different phases in building this model.

1. Importing the necessary dataset and libraries
2. Data pre-processing
3. N-Gram analysis of the entire corpus

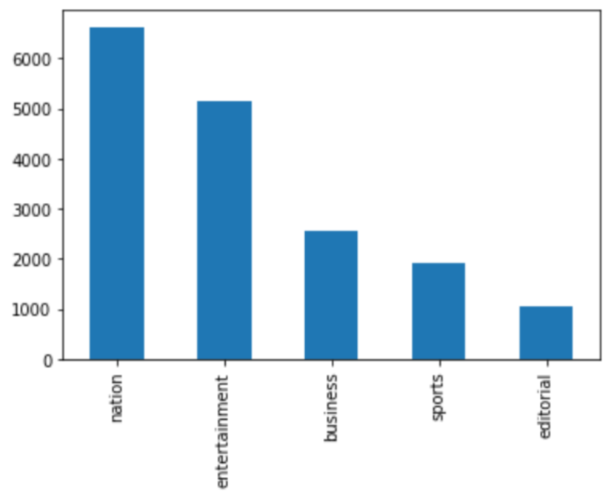


Figure 1.6: Data Visualization

4. Building the Language models
5. Classification of Telugu text based on topic

Phase 1: Importing the necessary dataset and libraries

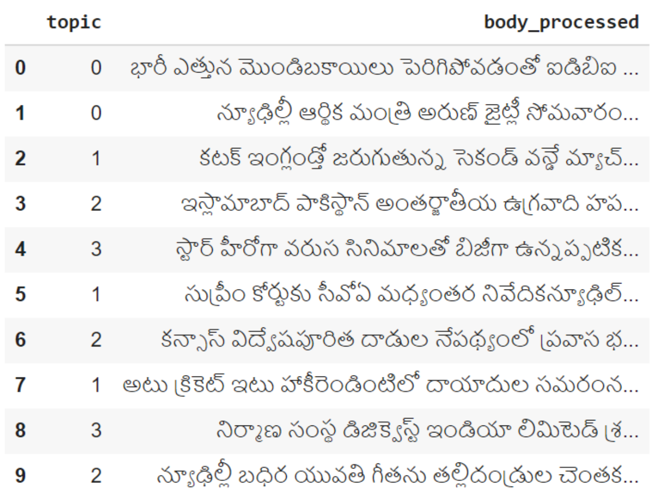
In this we import all the necessary libraries such as NLTK, Indic-NLP, Sklearn, etc, which are mentioned above. We imported the datasets and read the file into a Dataframe.

Phase 2: Data pre-processing

Removed the columns that are not necessary, such as SNo, date etcetera for better modeling purposes. Visualized the data with respect to the topic column. Computed and retrieved the unique attribute values. Processed the Body column by removing data such as punctuations, exclamatory marks, etc. A new column, body-processed, is added to the data frame for the further model-building process. Used Indic NLP dependencies to process the Telugu text directly. With the help of tokenization, we then separate all the words in the body context and build the vocabulary from the given dataset.

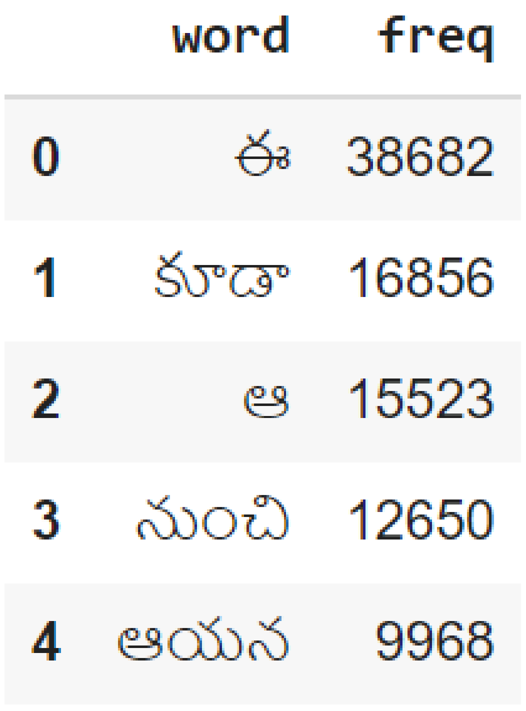
Phase 3: N-Gram analysis of the entire corpus

A Cor·pus is a collection of texts N-Gram analysis is a model that predicts the probability that a word might follow in a sequence. We train this model using the corpus of texts. An N-gram model is built by counting how often word



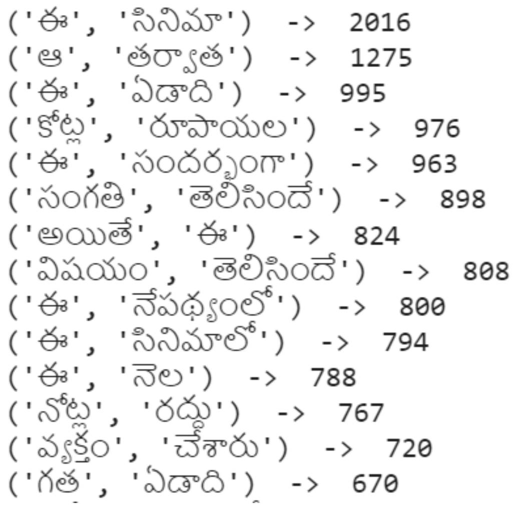
	topic	body_processed
0	0	భారీ ఎత్తున మొండిబకాయిలు పెరిగిపోవడంతో ఐడిబిఐ ...
1	0	న్యూఢిల్లీ ఆర్థిక మంత్రి అరుణ్ జైట్లీ సోమవారం...
2	1	కటక్ ఇంగ్లండ్తో జరుగుతున్న సెకండ్ వన్డే మ్యాచ్...
3	2	ఇస్లామాబాద్ పాకిస్థాన్ అంతర్జాతీయ ఉగ్రవాది హప...
4	3	స్టార్ హీరోగా వరుస సినిమాలతో బిజీగా ఉన్నప్పటికి...
5	1	సుప్రీం కోర్టుకు సీవోఏ మధ్యంతర నివేదికన్యూఢిల్...
6	2	కన్సాస్ విద్వేషపూరిత దాడుల నేపథ్యంలో ప్రవాస భ...
7	1	అటు క్రికెట్ ఇటు హాకీరెండింటిలో దాయాదుల సమరంన...
8	3	నిర్మాణ సంస్థ డిజిక్వెస్ట్ ఇండియా లిమిటెడ్ శ్ర...
9	2	న్యూఢిల్లీ బధిర యువతి గీతను తల్లిదండ్రుల చెంతక...

Figure 1.7: Processed data after removing punctuations



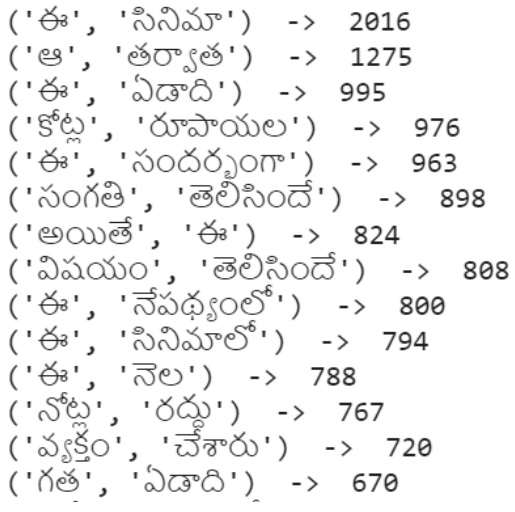
	word	freq
0	ఈ	38682
1	కూడా	16856
2	ఆ	15523
3	నుంచి	12650
4	ఆయన	9968

Figure 1.8: Uni-grams: The vocabulary



```
('ఈ', 'సినిమా')  ->  2016
('ఆ', 'తర్వాత')  ->  1275
('ఈ', 'ఏడాది')  ->  995
('కోట్ల', 'రూపాయల')  ->  976
('ఈ', 'సందర్భంగా')  ->  963
('సంగతి', 'తెలిసిందే')  ->  898
('అయితే', 'ఈ')  ->  824
('విషయం', 'తెలిసిందే')  ->  808
('ఈ', 'నేపథ్యంలో')  ->  800
('ఈ', 'సినిమాలో')  ->  794
('ఈ', 'నెల')  ->  788
('నోట్ల', 'రద్దు')  ->  767
('వ్యక్తం', 'చేశారు')  ->  720
('గత', 'ఏడాది')  ->  670
```

Figure 1.9: Bigrams



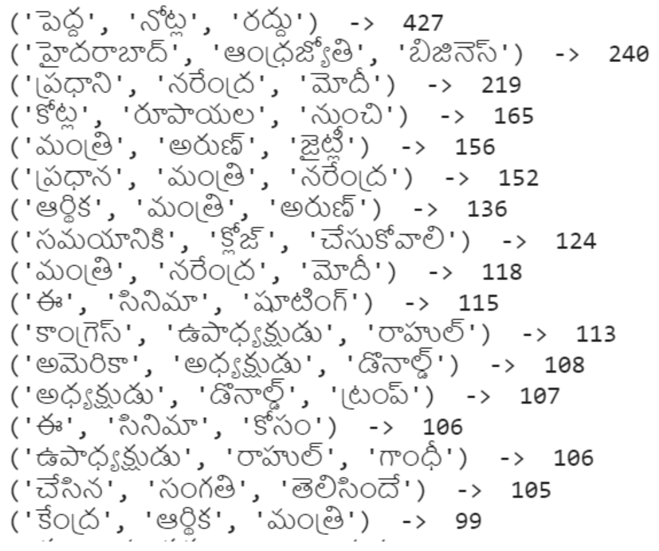
```
('ఈ', 'సినిమా')  ->  2016
('ఆ', 'తర్వాత')  ->  1275
('ఈ', 'ఏడాది')  ->  995
('కోట్ల', 'రూపాయల')  ->  976
('ఈ', 'సందర్భంగా')  ->  963
('సంగతి', 'తెలిసిందే')  ->  898
('అయితే', 'ఈ')  ->  824
('విషయం', 'తెలిసిందే')  ->  808
('ఈ', 'నేపథ్యంలో')  ->  800
('ఈ', 'సినిమాలో')  ->  794
('ఈ', 'నెల')  ->  788
('నోట్ల', 'రద్దు')  ->  767
('వ్యక్తం', 'చేశారు')  ->  720
('గత', 'ఏడాది')  ->  670
```

Figure 1.10: Bigrams

sequences occur in corpus text and then estimating the probabilities. We derive n-grams with different values of n to check the uniqueness of their occurrences. We first find bigrams, two words coming together in the corpus. Similarly, we use the preprocessed body data to derive the tri grams, a set of 3 words, a four-gram word list,s, and a five-gram word list. With the help of these, we retrieve the unique word lists. The observation we achieved from this phase is that the uniqueness is low in unigram lists which says that in any language, most of the words are repetitive in news content.

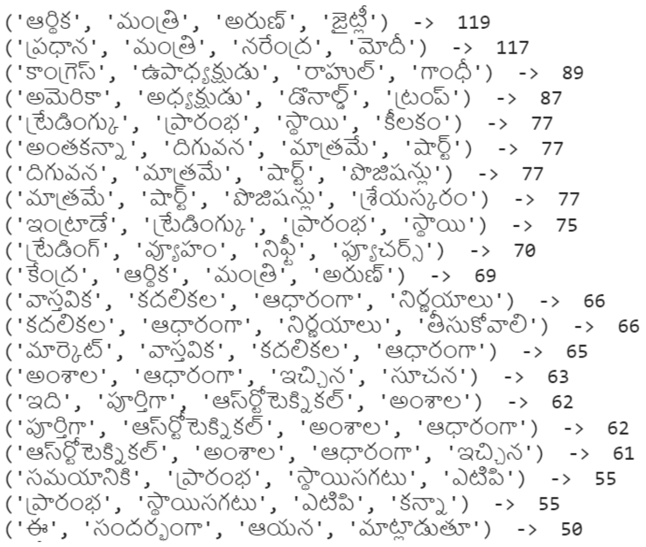
Phase 4: Building the Language models

In this phase, to analyze the data context, we have checked which n-gram model



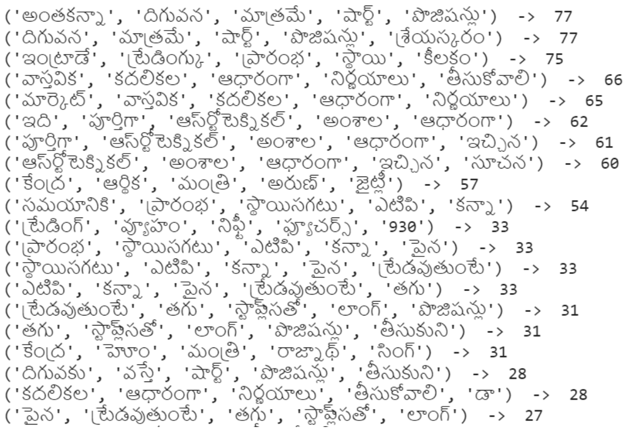
```
('పెద్ద', 'నోట్ల', 'రద్దు')  ->  427
('హైదరాబాద్', 'ఆంధ్రజ్యోతి', 'బిజినెస్')  ->  240
('ప్రధాని', 'నరేంద్ర', 'మోదీ')  ->  219
('కోట్ల', 'రూపాయల', 'నుంచి')  ->  165
('మంత్రి', 'అరుణ్', 'జైట్లీ')  ->  156
('ప్రధాన', 'మంత్రి', 'నరేంద్ర')  ->  152
('ఆర్థిక', 'మంత్రి', 'అరుణ్')  ->  136
('సమయానికి', 'క్లోజ్', 'చేసుకోవాలి')  ->  124
('మంత్రి', 'నరేంద్ర', 'మోదీ')  ->  118
('ఈ', 'సినిమా', 'షూటింగ్')  ->  115
('కాంగ్రెస్', 'ఉపాధ్యక్షుడు', 'రాహుల్')  ->  113
('అమెరికా', 'అధ్యక్షుడు', 'డొనాల్డ్')  ->  108
('అధ్యక్షుడు', 'డొనాల్డ్', 'ట్రంప్')  ->  107
('ఈ', 'సినిమా', 'కోసం')  ->  106
('ఉపాధ్యక్షుడు', 'రాహుల్', 'గాంధీ')  ->  106
('చేసిన', 'సంగతి', 'తెలిసిందే')  ->  105
('కేంద్ర', 'ఆర్థిక', 'మంత్రి')  ->  99
```

Figure 1.11: Trigrams



```
('ఆర్థిక', 'మంత్రి', 'అరుణ్', 'జైట్లీ')  ->  119
('ప్రధాన', 'మంత్రి', 'నరేంద్ర', 'మోదీ')  ->  117
('కాంగ్రెస్', 'ఉపాధ్యక్షుడు', 'రాహుల్', 'గాంధీ')  ->  89
('అమెరికా', 'అధ్యక్షుడు', 'డొనాల్డ్', 'ట్రంప్')  ->  87
('ట్రేడింగ్కు', 'ప్రారంభ', 'స్థాయి', 'కీలకం')  ->  77
('అంతకన్నా', 'దిగువన', 'మాత్రమే', 'షార్ట్')  ->  77
('దిగువన', 'మాత్రమే', 'షార్ట్', 'పొజిషన్లు')  ->  77
('మాత్రమే', 'షార్ట్', 'పొజిషన్లు', 'శ్రేయస్కరం')  ->  77
('ఇంట్రాడే', 'ట్రేడింగ్కు', 'ప్రారంభ', 'స్థాయి')  ->  75
('ట్రేడింగ్', 'వ్యూహం', 'నిఫ్టీ', 'ఫ్యూచర్స్')  ->  70
('కేంద్ర', 'ఆర్థిక', 'మంత్రి', 'అరుణ్')  ->  69
('వాస్తవిక', 'కదలికల', 'ఆధారంగా', 'నిర్ణయాలు')  ->  66
('కదలికల', 'ఆధారంగా', 'నిర్ణయాలు', 'తీసుకోవాలి')  ->  66
('మార్కెట్', 'వాస్తవిక', 'కదలికల', 'ఆధారంగా')  ->  65
('అంశాల', 'ఆధారంగా', 'ఇచ్చిన', 'సూచన')  ->  63
('ఇది', 'పూర్తిగా', 'ఆస్ట్రోటెక్నికల్', 'అంశాల')  ->  62
('పూర్తిగా', 'ఆస్ట్రోటెక్నికల్', 'అంశాల', 'ఆధారంగా')  ->  62
('ఆస్ట్రోటెక్నికల్', 'అంశాల', 'ఆధారంగా', 'ఇచ్చిన')  ->  61
('సమయానికి', 'ప్రారంభ', 'స్థాయిసగటు', 'ఎటిపి')  ->  55
('ప్రారంభ', 'స్థాయిసగటు', 'ఎటిపి', 'కన్నా')  ->  55
('ఈ', 'సందర్భంగా', 'ఆయన', 'మాట్లాడుతూ')  ->  50
```

Figure 1.12: 5-grams



```
('అంతకన్నా', 'దిగువన', 'మాత్రమే', 'షార్ట్', 'పొజిషన్లు')  ->  77
('దిగువన', 'మాత్రమే', 'షార్ట్', 'పొజిషన్లు', 'శ్రేయస్కరం')  ->  77
('ఇంట్రాడే', 'ట్రేడింగ్కు', 'ప్రారంభ', 'స్థాయి', 'కీలకం')  ->  75
('వాస్తవిక', 'కదలికల', 'ఆధారంగా', 'నిర్ణయాలు', 'తీసుకోవాలి')  ->  66
('మార్కెట్', 'వాస్తవిక', 'కదలికల', 'ఆధారంగా', 'నిర్ణయాలు')  ->  65
('ఇది', 'పూర్తిగా', 'ఆస్ట్రోటెక్నికల్', 'అంశాల', 'ఆధారంగా')  ->  62
('పూర్తిగా', 'ఆస్ట్రోటెక్నికల్', 'అంశాల', 'ఆధారంగా', 'ఇచ్చిన')  ->  61
('ఆస్ట్రోటెక్నికల్', 'అంశాల', 'ఆధారంగా', 'ఇచ్చిన', 'సూచన')  ->  60
('కేంద్ర', 'ఆర్థిక', 'మంత్రి', 'అరుణ్', 'జైట్లీ')  ->  57
('సమయానికి', 'ప్రారంభ', 'స్థాయిసగటు', 'ఎటిపి', 'కన్నా')  ->  54
('ట్రేడింగ్', 'వ్యూహం', 'నిఫ్టీ', 'ఫ్యూచర్స్', '930')  ->  33
('ప్రారంభ', 'స్థాయిసగటు', 'ఎటిపి', 'కన్నా', 'పైన')  ->  33
('స్థాయిసగటు', 'ఎటిపి', 'కన్నా', 'పైన', 'ట్రేడవుతుంటే')  ->  33
('ఎటిపి', 'కన్నా', 'పైన', 'ట్రేడవుతుంటే', 'తగు')  ->  33
('ట్రేడవుతుంటే', 'తగు', 'స్టాప్లాసతో', 'లాంగ్', 'పొజిషన్లు')  ->  31
('తగు', 'స్టాప్లాసతో', 'లాంగ్', 'పొజిషన్లు', 'తీసుకుని')  ->  31
('కేంద్ర', 'హోం', 'మంత్రి', 'రాజ్నాథ్', 'సింగ్')  ->  31
('దిగువకు', 'వస్తే', 'షార్ట్', 'పొజిషన్లు', 'తీసుకుని')  ->  28
('కదలికల', 'ఆధారంగా', 'నిర్ణయాలు', 'తీసుకోవాలి', 'డా')  ->  28
('పైన', 'ట్రేడవుతుంటే', 'తగ్గు', 'స్టాప్లాసతో', 'లాంగ్')  ->  27
```

Figure 1.13: 4-grams

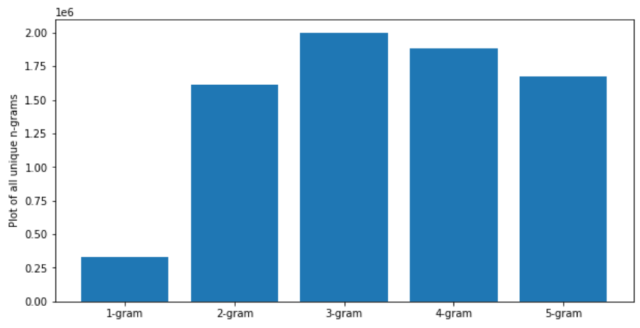


Figure 1.14: N-gram Visualization

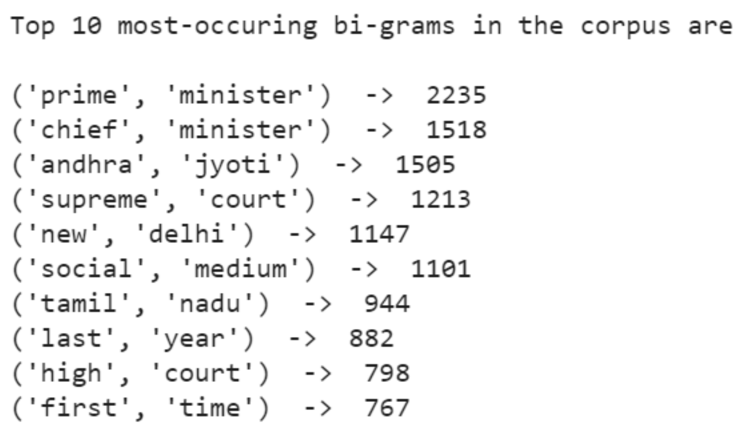
is better at classifying the data using the Laplace transform. The observation we achieved from this phase, as the length of the input sentence increases, the final prediction chances decrease. This is due to the multiplication of probabilities, so we generally use the log probabilities instead of the standard multiplication of probabilities.

Phase 5: Classification of Telugu text based on topic

In this Phase, we first Preprocess the test data by removing punctuations, tokenizing, etc., which is a similar process we followed for the training data. We then use Count Vectorizer to get the data into sk-learn's format. We have used uni-grams and bi-grams here to get the feature vectors. Here, Count Vectorizer helps get the feature vectors and eliminate the stopwords (based on term and inverse document frequency and selecting the top k words in the vocabulary for model development purposes). In Multinomial Naive Baye's classifier, we consider a feature vector where a given term represents the number of times it appears or how often that is the frequency using which the classification is done for the Telugu texts in the test dataset. To predict the sample text, we have created a function with the help of which we observed the performance and prediction accuracy with the Multinomial Naive Bayes classifier.

1.5 Classification of Translated Telugu Text using n-gram analysis

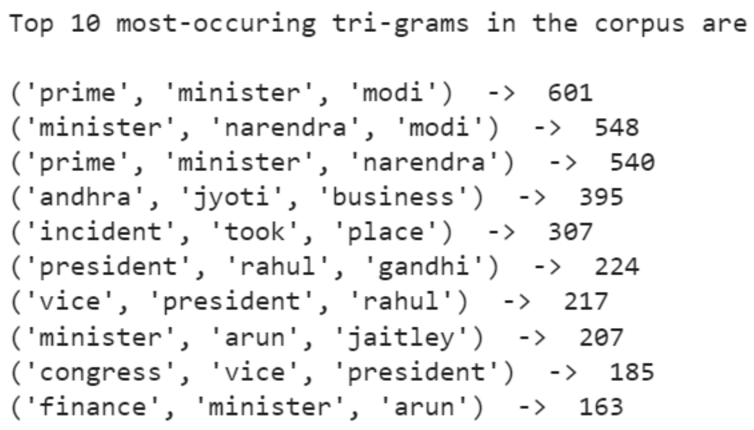
We trained a model combining the Two previously developed models: the lemmatization of translated text and then performing n-gram analysis. Further, unigrams and bi-grams are used as input feature vectors. The model is evaluated using multiple classification models and observed as Multinomial Naive Bayes having a better overall performance.



```
Top 10 most-occuring bi-grams in the corpus are

('prime', 'minister')  ->  2235
('chief', 'minister')  ->  1518
('andhra', 'jyoti')  ->  1505
('supreme', 'court')  ->  1213
('new', 'delhi')  ->  1147
('social', 'medium')  ->  1101
('tamil', 'nadu')  ->  944
('last', 'year')  ->  882
('high', 'court')  ->  798
('first', 'time')  ->  767
```

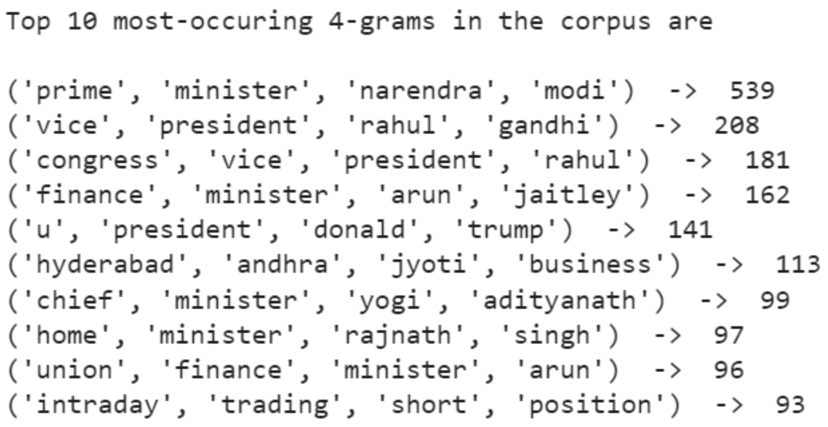
Figure 1.15: Frequently occurring Bi-grams



```
Top 10 most-occuring tri-grams in the corpus are

('prime', 'minister', 'modi')  ->  601
('minister', 'narendra', 'modi')  ->  548
('prime', 'minister', 'narendra')  ->  540
('andhra', 'jyoti', 'business')  ->  395
('incident', 'took', 'place')  ->  307
('president', 'rahul', 'gandhi')  ->  224
('vice', 'president', 'rahul')  ->  217
('minister', 'arun', 'jaitley')  ->  207
('congress', 'vice', 'president')  ->  185
('finance', 'minister', 'arun')  ->  163
```

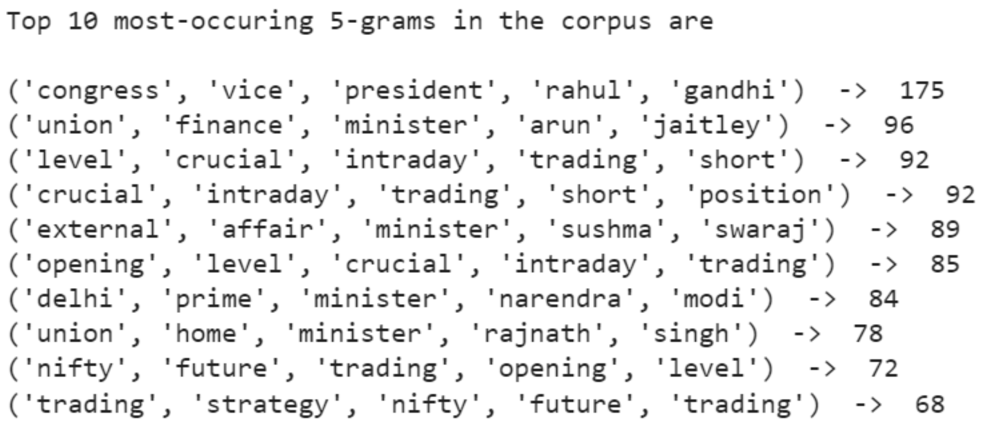
Figure 1.16: Frequently occurring Tri-grams



```
Top 10 most-occuring 4-grams in the corpus are

('prime', 'minister', 'narendra', 'modi')  ->  539
('vice', 'president', 'rahul', 'gandhi')  ->  208
('congress', 'vice', 'president', 'rahul')  ->  181
('finance', 'minister', 'arun', 'jaitley')  ->  162
('u', 'president', 'donald', 'trump')  ->  141
('hyderabad', 'andhra', 'jyoti', 'business')  ->  113
('chief', 'minister', 'yogi', 'adityanath')  ->  99
('home', 'minister', 'rajnath', 'singh')  ->  97
('union', 'finance', 'minister', 'arun')  ->  96
('intraday', 'trading', 'short', 'position')  ->  93
```

Figure 1.17: Frequently occurring 4-grams



```
Top 10 most-occuring 5-grams in the corpus are

('congress', 'vice', 'president', 'rahul', 'gandhi')  ->  175
('union', 'finance', 'minister', 'arun', 'jaitley')  ->  96
('level', 'crucial', 'intraday', 'trading', 'short')  ->  92
('crucial', 'intraday', 'trading', 'short', 'position')  ->  92
('external', 'affair', 'minister', 'sushma', 'swaraj')  ->  89
('opening', 'level', 'crucial', 'intraday', 'trading')  ->  85
('delhi', 'prime', 'minister', 'narendra', 'modi')  ->  84
('union', 'home', 'minister', 'rajnath', 'singh')  ->  78
('nifty', 'future', 'trading', 'opening', 'level')  ->  72
('trading', 'strategy', 'nifty', 'future', 'trading')  ->  68
```

Figure 1.18: Frequently occurring 5-grams

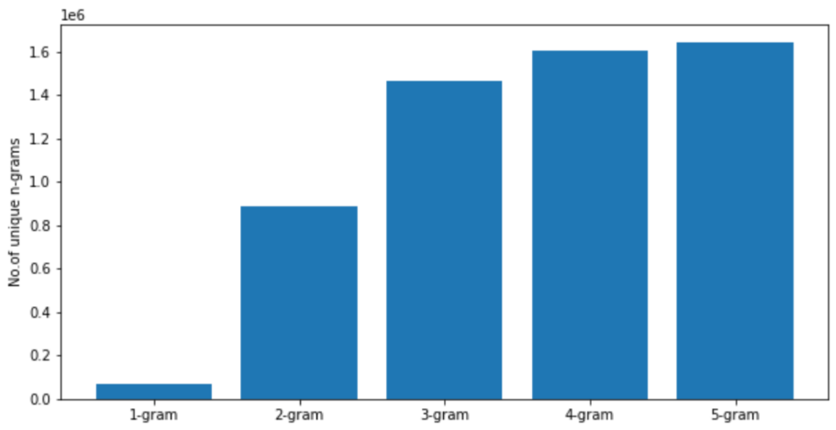


Figure 1.19: N-grams Visualized

Chapter 2

Related Work

Work has been done on the classification of Telugu text previously. The same dataset used in this project was applied in [1] to classify the topic of news. Details of data cleaning and pre-processing seemed to be lacking in the work. The work did not explore the application of sentiment analysis and classification. There is a common use of LSTMs and BERT in NLP and one such piece of work was on the Detection of Click baits [2]. Word2Vec and TF-IDF were used on the data which made us want to explore them. It is clear that there is a common pattern of NLP tasks followed when performing Sentiment Analysis through the use of models like BERT and Data conversion and representation techniques like TF-IDF. This work was further extended to perform the classification of multi-task texts [3]. IndicNLP Library is one Library that offers text-processing methods for Indian Languages. This library demonstrates the growing interest in research in languages other than English [4].

Chapter 3

Data Description

The data we used for this project was acquired from an openly available resource. The raw data we started working with contained five columns. Each entry of data contained a serial number and date of publishing which were redundant. The three remaining features were the most important features for us. They were the heading or title, the article itself, and the topic or genre of news. To start with, we had over 21,000 rows of data combined. The dataset was partitioned from the origin resource. It was combined into one unit. As a result of data pre-processing and translations, we lost some of our data resulting in 19,000 rows to work with.

Chapter 4

Experiments Results

4.1 Sentiment Analysis and Experiment with Pretrained Model

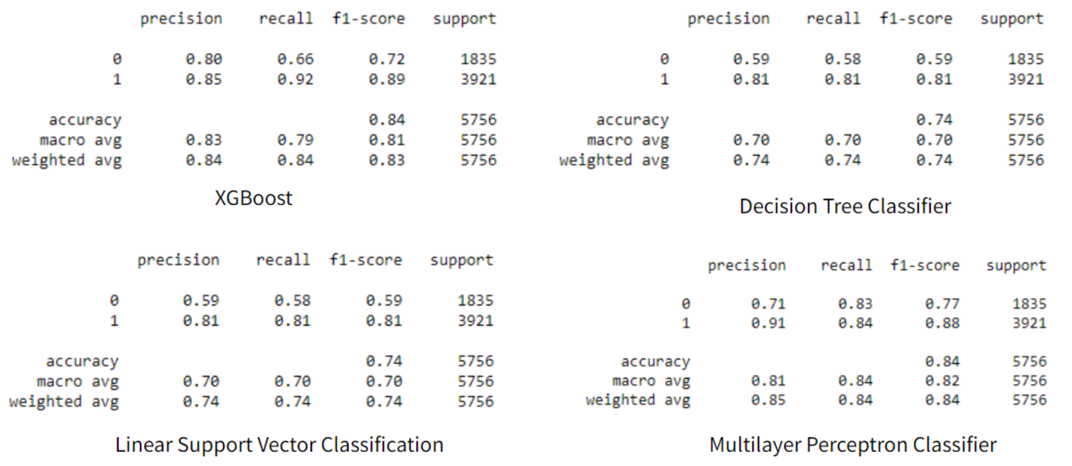
We played around with a pre-trained BERT model that was available for use. We tested the model with the same feature data, which was the Telugu Text. We played with the target variables. In one run, we tested the model with the target as the Topic of news. In the other, we tested with the target variable as the Sentiment Class. The model performed decently when tested with the Sentiment Target Class. However, it was interesting to see some of the other classification models to compare the performance. From Fig. 4.1, Fig 4.2, and Table 4.1, The BERT in some cases delivered a lesser accuracy than models like XGBoost and MPC. Note that the training features data for the classification models were different, ie; The TF-IDF vector.

Table 4.1: Pre-Trained BERT Model Results

Features	Target	Accuracy
Article (Telugu script)	Genre Class	0.2192
Article (Telugu script)	Sentiment Class	0.7138

4.2 Lemmatization and word2vec Classification

From Fig. 4.5 and 4.12, Table 4.2 and 4.3 ,The models were evaluated using various Machine learning algorithms and accuracy is measured. Two different vectorizers namely TF-IDF and CoutVectorizer are used and their performance is measured. From the classification reports of the two methods, we can observe



```
              precision    recall  f1-score   support

           0       0.80      0.66      0.72      1835
           1       0.85      0.92      0.89      3921

    accuracy                           0.84      5756
   macro avg       0.83      0.79      0.81      5756
weighted avg       0.84      0.84      0.83      5756
```

XGBoost

```
              precision    recall  f1-score   support

           0       0.59      0.58      0.59      1835
           1       0.81      0.81      0.81      3921

    accuracy                           0.74      5756
   macro avg       0.70      0.70      0.70      5756
weighted avg       0.74      0.74      0.74      5756
```

Decision Tree Classifier

```
              precision    recall  f1-score   support

           0       0.59      0.58      0.59      1835
           1       0.81      0.81      0.81      3921

    accuracy                           0.74      5756
   macro avg       0.70      0.70      0.70      5756
weighted avg       0.74      0.74      0.74      5756
```

Linear Support Vector Classification

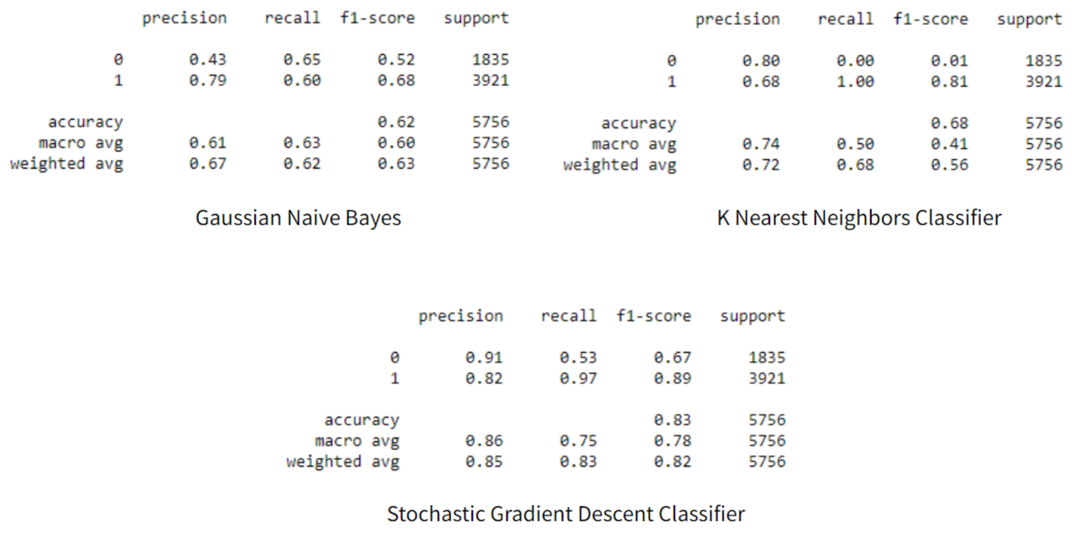
```
              precision    recall  f1-score   support

           0       0.71      0.83      0.77      1835
           1       0.91      0.84      0.88      3921

    accuracy                           0.84      5756
   macro avg       0.81      0.84      0.82      5756
weighted avg       0.85      0.84      0.84      5756
```

Multilayer Perceptron Classifier

Figure 4.1: Sentiment Classification with Classification Models 1



```
              precision    recall  f1-score   support

           0       0.43      0.65      0.52      1835
           1       0.79      0.60      0.68      3921

    accuracy                           0.62      5756
   macro avg       0.61      0.63      0.60      5756
weighted avg       0.67      0.62      0.63      5756
```

Gaussian Naive Bayes

```
              precision    recall  f1-score   support

           0       0.80      0.00      0.01      1835
           1       0.68      1.00      0.81      3921

    accuracy                           0.68      5756
   macro avg       0.74      0.50      0.41      5756
weighted avg       0.72      0.68      0.56      5756
```

K Nearest Neighbors Classifier

```
              precision    recall  f1-score   support

           0       0.91      0.53      0.67      1835
           1       0.82      0.97      0.89      3921

    accuracy                           0.83      5756
   macro avg       0.86      0.75      0.78      5756
weighted avg       0.85      0.83      0.82      5756
```

Stochastic Gradient Descent Classifier

Figure 4.2: Sentiment Classification with Classification Models 2

that the performance metrics are almost similar for both models. We can say that both methods can be used considering their accuracy.

4.3 Classification of Telugu Text using n-gram analysis

From Fig 4.5 to 4.11, Our model accurately predicts any given Telugu news data by placing it under its relevant topic.

Table 4.2: Lemmatization and word2vec Classification with TF-IDF

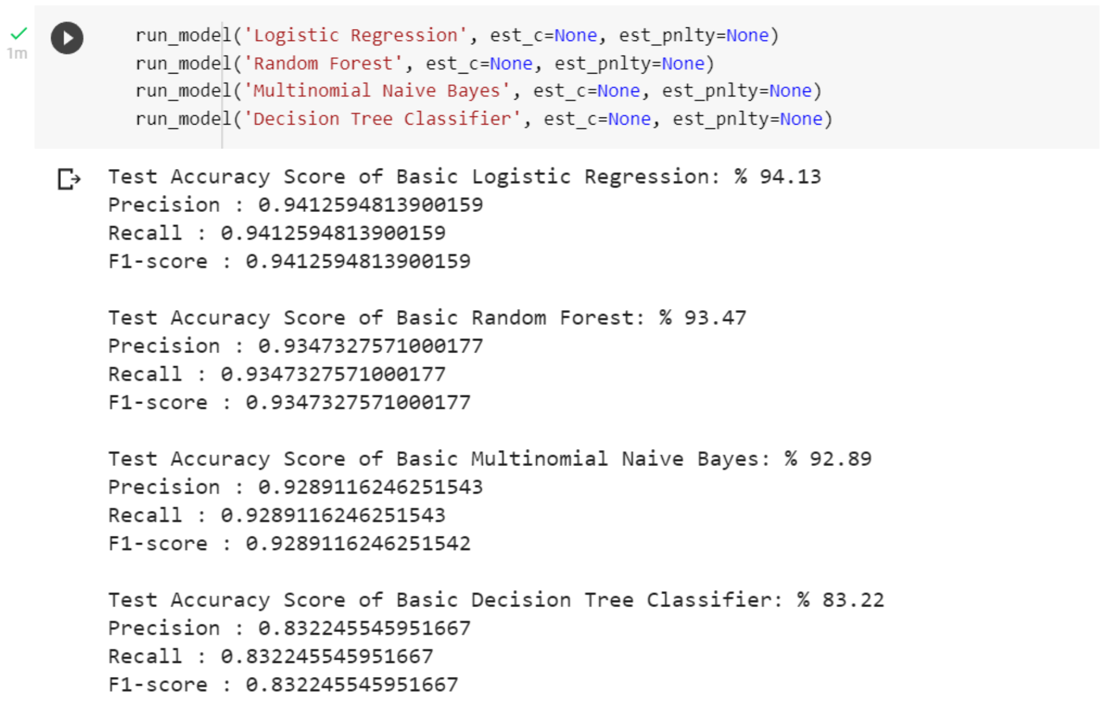
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	94.13	0.94	0.94	0.94
Random Forest	93.47	0.93	0.93	0.93
Multinomial Naive Bayes	93.63	0.94	0.94	0.94
Decision Tree Classifier	84.48	0.84	0.84	0.84

Table 4.3: Lemmatization and word2vec Classification with CountVectorizer

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	94.13	0.94	0.94	0.94
Random Forest	93.47	0.93	0.93	0.93
Multinomial Naive Bayes	92.89	0.93	0.93	0.93
Decision Tree Classifier	83.22	0.83	0.83	0.83

4.4 Classification of Translated Telugu Text using n-gram analysis - A different approach

From Fig 4.12 to 4.14, Multiple classification models are performed and Multinomial Naive Bayes shows the better performance among all. The model is tested with a Telugu text which was taken from a new article website called 'Eenadu'. This test text is not present in the model. This test text is given as input to the trained model to classify Some of the limitations might be because Data sparsity is more of an issue in NLP than in other machine learning fields. We typically deal with large vocabularies where it is impossible to have enough data to actually observe examples of all the things that people can say. There will be many real phrases that we will just never see in the training data. A limited dataset and uneven distribution can be an issue because the model can't learn with very little data which can lead to misclassification.



```
run_model('Logistic Regression', est_c=None, est_pnlty=None)
run_model('Random Forest', est_c=None, est_pnlty=None)
run_model('Multinomial Naive Bayes', est_c=None, est_pnlty=None)
run_model('Decision Tree Classifier', est_c=None, est_pnlty=None)
```

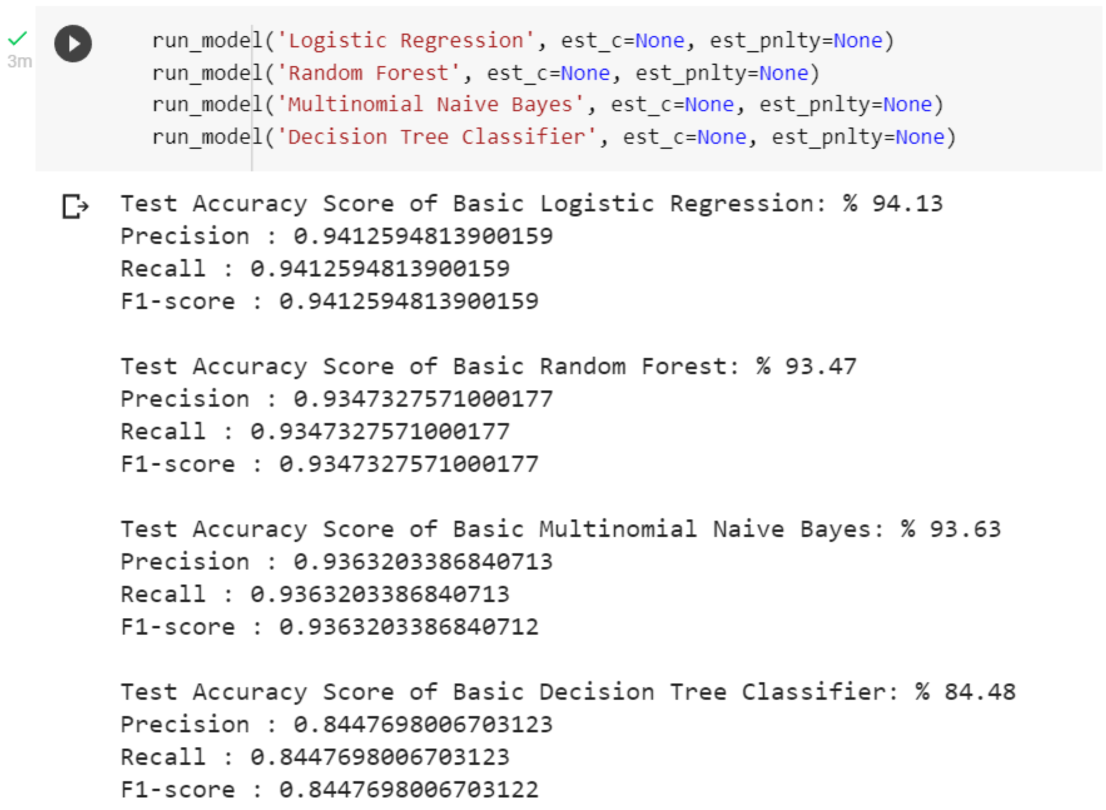
```
Test Accuracy Score of Basic Logistic Regression: % 94.13
Precision : 0.9412594813900159
Recall : 0.9412594813900159
F1-score : 0.9412594813900159

Test Accuracy Score of Basic Random Forest: % 93.47
Precision : 0.9347327571000177
Recall : 0.9347327571000177
F1-score : 0.9347327571000177

Test Accuracy Score of Basic Multinomial Naive Bayes: % 92.89
Precision : 0.9289116246251543
Recall : 0.9289116246251543
F1-score : 0.9289116246251542

Test Accuracy Score of Basic Decision Tree Classifier: % 83.22
Precision : 0.832245545951667
Recall : 0.832245545951667
F1-score : 0.832245545951667
```

Figure 4.3: Word2vec Classification



```
run_model('Logistic Regression', est_c=None, est_pnlty=None)
run_model('Random Forest', est_c=None, est_pnlty=None)
run_model('Multinomial Naive Bayes', est_c=None, est_pnlty=None)
run_model('Decision Tree Classifier', est_c=None, est_pnlty=None)
```

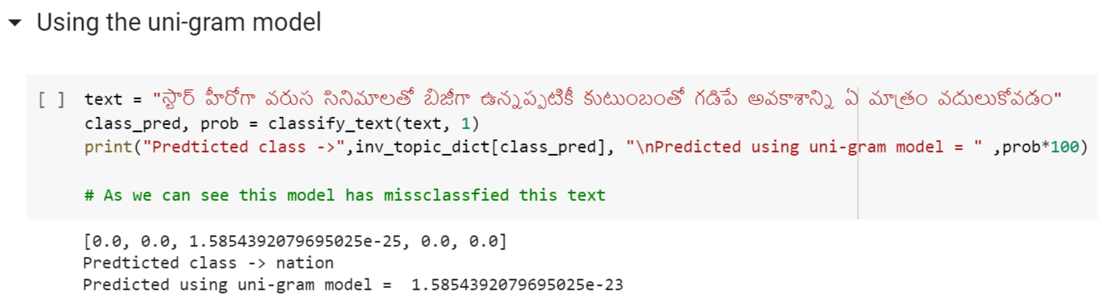
```
Test Accuracy Score of Basic Logistic Regression: % 94.13
Precision : 0.9412594813900159
Recall : 0.9412594813900159
F1-score : 0.9412594813900159

Test Accuracy Score of Basic Random Forest: % 93.47
Precision : 0.9347327571000177
Recall : 0.9347327571000177
F1-score : 0.9347327571000177

Test Accuracy Score of Basic Multinomial Naive Bayes: % 93.63
Precision : 0.9363203386840713
Recall : 0.9363203386840713
F1-score : 0.9363203386840712

Test Accuracy Score of Basic Decision Tree Classifier: % 84.48
Precision : 0.8447698006703123
Recall : 0.8447698006703123
F1-score : 0.8447698006703122
```

Figure 4.4: Lemmatization Classification



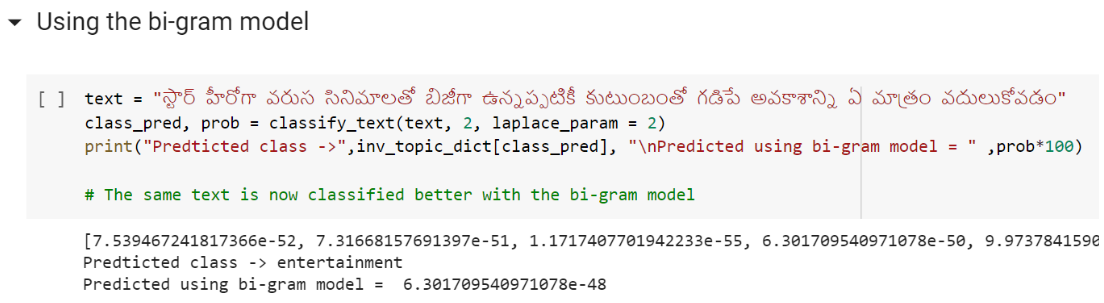
Using the uni-gram model

```
[ ]  text = "స్టార్ హీరోగా వరుస సినిమాలతో బిజీగా ఉన్నప్పటికీ కుటుంబంతో గడిపే అవకాశాన్ని ఏ మాత్రం వదులుకోవడం"
     class_pred, prob = classify_text(text, 1)
     print("Predticted class ->",inv_topic_dict[class_pred], "\nPredicted using uni-gram model = " ,prob*100)

     # As we can see this model has missclassfied this text

     [0.0, 0.0, 1.5854392079695025e-25, 0.0, 0.0]
     Predticted class -> nation
     Predicted using uni-gram model =  1.5854392079695025e-23
```

Figure 4.5: Unigram Model



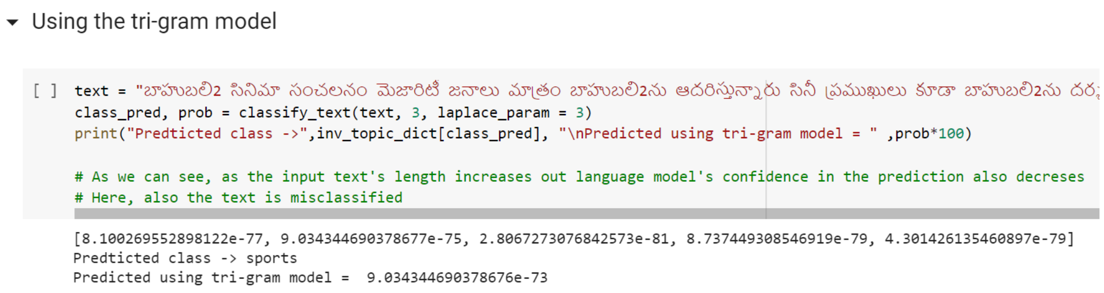
Using the bi-gram model

```
[ ]  text = "స్టార్ హీరోగా వరుస సినిమాలతో బిజీగా ఉన్నప్పటికీ కుటుంబంతో గడిపే అవకాశాన్ని ఏ మాత్రం వదులుకోవడం"
     class_pred, prob = classify_text(text, 2, laplace_param = 2)
     print("Predticted class ->",inv_topic_dict[class_pred], "\nPredicted using bi-gram model = " ,prob*100)

     # The same text is now classified better with the bi-gram model

     [7.539467241817366e-52, 7.31668157691397e-51, 1.1717407701942233e-55, 6.301709540971078e-50, 9.9737841590
     Predticted class -> entertainment
     Predicted using bi-gram model =  6.301709540971078e-48
```

Figure 4.6: Bigram Model



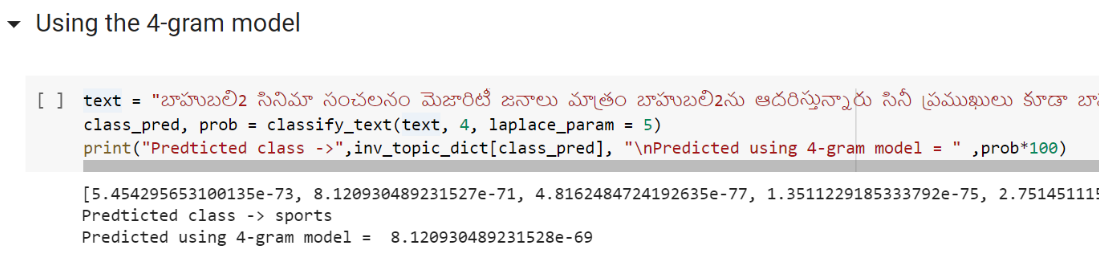
Using the tri-gram model

```
[ ]  text = "బాహుబలి2 సినిమా సంచలనం మెజారిటీ జనాలు మాత్రం బాహుబలి2ను ఆదరిస్తున్నారు సినీ ప్రముఖులు కూడా బాహుబలి2ను దర
     class_pred, prob = classify_text(text, 3, laplace_param = 3)
     print("Predticted class ->",inv_topic_dict[class_pred], "\nPredicted using tri-gram model = " ,prob*100)

     # As we can see, as the input text's length increases out language model's confidence in the prediction also decreses
     # Here, also the text is misclassified

     [8.100269552898122e-77, 9.034344690378677e-75, 2.8067273076842573e-81, 8.737449308546919e-79, 4.301426135460897e-79]
     Predticted class -> sports
     Predicted using tri-gram model =  9.034344690378676e-73
```

Figure 4.7: Trigram Model

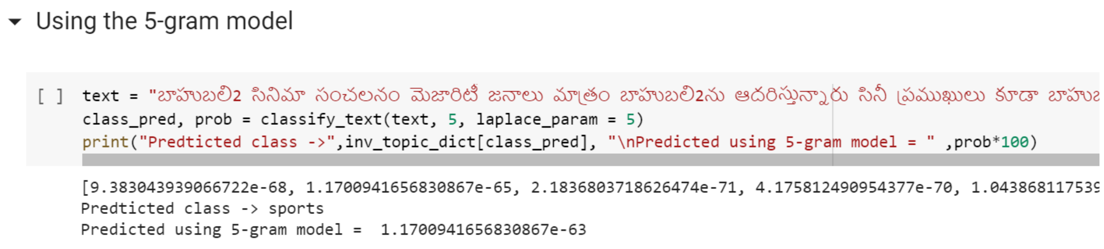


Using the 4-gram model

```
[ ]  text = "బాహుబలి2 సినిమా సంచలనం మెజారిటీ జనాలు మాత్రం బాహుబలి2ను ఆదరిస్తున్నారు సినీ ప్రముఖులు కూడా బా
     class_pred, prob = classify_text(text, 4, laplace_param = 5)
     print("Predticted class ->",inv_topic_dict[class_pred], "\nPredicted using 4-gram model = " ,prob*100)

     [5.454295653100135e-73, 8.120930489231527e-71, 4.8162484724192635e-77, 1.3511229185333792e-75, 2.75145111
     Predticted class -> sports
     Predicted using 4-gram model =  8.120930489231528e-69
```

Figure 4.8: 4-Gram Model

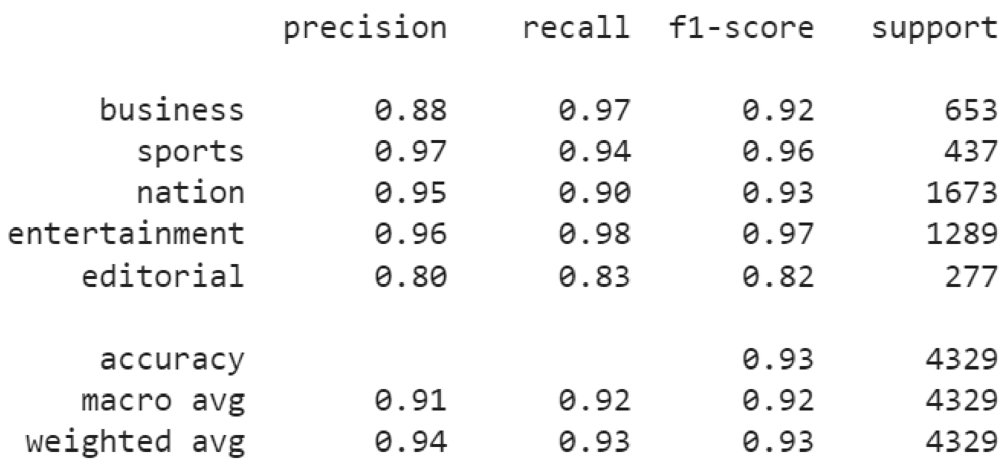


Using the 5-gram model

```
[ ]  text = "బాహుబలి2 సినిమా సంచలనం మెజారిటీ జనాలు మాత్రం బాహుబలి2ను ఆదరిస్తున్నారు సినీ ప్రముఖులు కూడా బాహు
     class_pred, prob = classify_text(text, 5, laplace_param = 5)
     print("Predticted class ->",inv_topic_dict[class_pred], "\nPredicted using 5-gram model = " ,prob*100)

     [9.383043939066722e-68, 1.1700941656830867e-65, 2.1836803718626474e-71, 4.175812490954377e-70, 1.04386811753
     Predticted class -> sports
     Predicted using 5-gram model =  1.1700941656830867e-63
```

Figure 4.9: 5-Gram Model



```
               precision    recall  f1-score   support

     business       0.88      0.97      0.92       653
       sports       0.97      0.94      0.96       437
       nation       0.95      0.90      0.93      1673
entertainment       0.96      0.98      0.97      1289
    editorial       0.80      0.83      0.82       277

     accuracy                           0.93      4329
    macro avg       0.91      0.92      0.92      4329
 weighted avg       0.94      0.93      0.93      4329
```

Figure 4.10: Multinomial Naive Bayes Classification Report

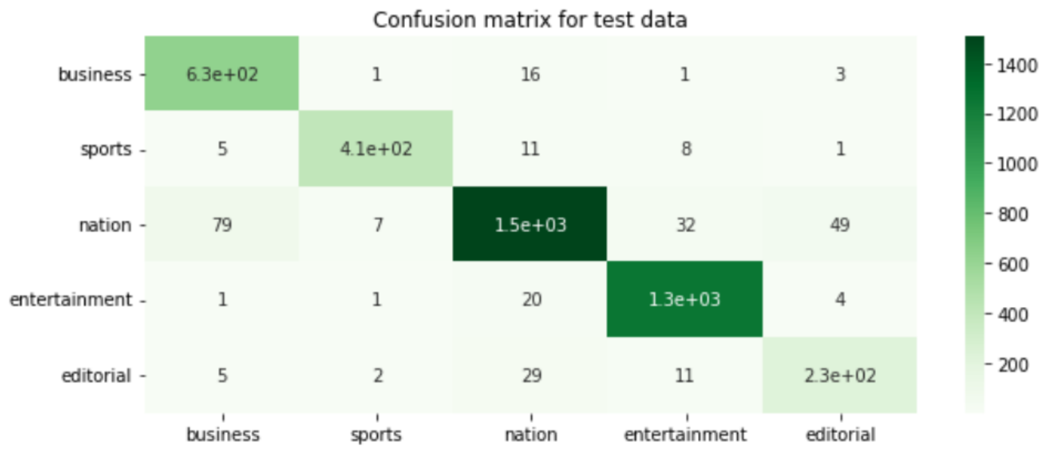
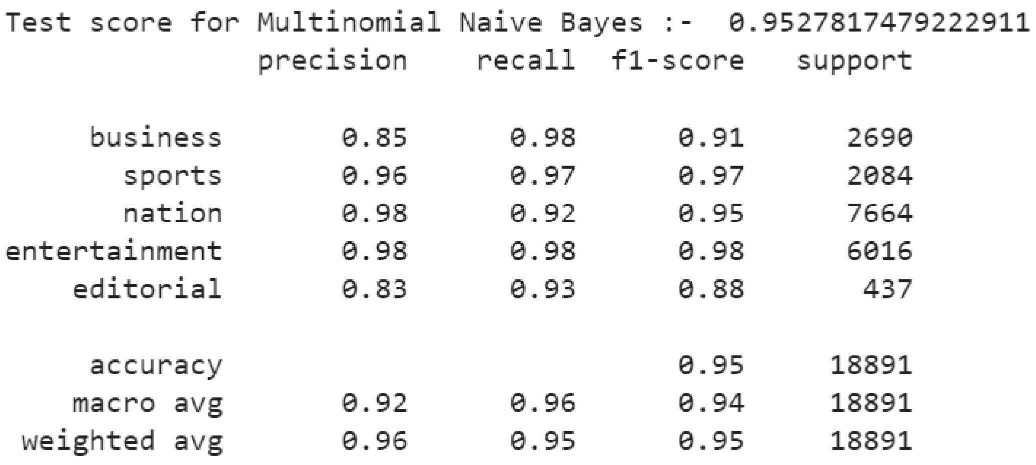


Figure 4.11: Test data confusion matrix



```
Test score for Multinomial Naive Bayes :-  0.9527817479222911
               precision    recall  f1-score   support

     business       0.85      0.98      0.91      2690
       sports       0.96      0.97      0.97      2084
       nation       0.98      0.92      0.95      7664
entertainment       0.98      0.98      0.98      6016
    editorial       0.83      0.93      0.88       437

     accuracy                           0.95     18891
    macro avg       0.92      0.96      0.94     18891
 weighted avg       0.96      0.95      0.95     18891
```

Figure 4.12: Multinomial Naive Bayes Test with Telugu Text

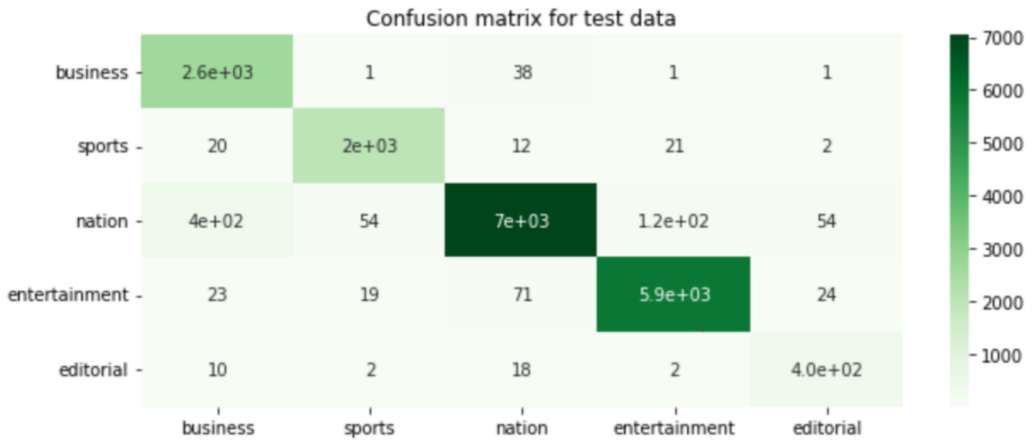
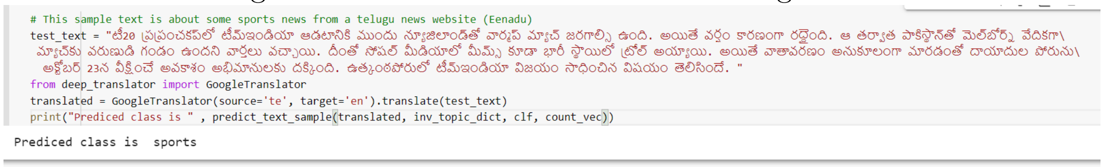


Figure 4.13: Confusion matrix with Telugu Text



```
# This sample text is about some sports news from a telugu news website (Eenadu)
test_text = "టీ20 ప్రపంచకప్‌లో టీమ్‌ఇండియా ఆడటానికి ముందు న్యూజిలాండ్‌తో వార్మప్ మ్యాచ్ జరగాల్సి ఉంది. అయితే వర్షం కారణంగా రద్దైంది. ఆ తర్వాత పాకిస్థాన్‌తో మెల్‌బోర్న్ వేదికగా\
 మ్యాచ్‌కు వరుణుడి గండం ఉందని వార్తలు వచ్చాయి. దీంతో సోషల్ మీడియాలో మీమ్స్ కూడా భారీ స్థాయిలో ట్రోల్ అయ్యాయి. అయితే వాతావరణం అనుకూలంగా మారడంతో దాయాదుల పోరును\
  అక్టోబర్ 23న వీక్షించే అవకాశం అభిమానులకు దక్కింది. ఉత్కంఠపోరులో టీమ్‌ఇండియా విజయం సాధించిన విషయం తెలిసిందే. "
from deep_translator import GoogleTranslator
translated = GoogleTranslator(source='te', target='en').translate(test_text)
print("Prediced class is " , predict_text_sample(translated, inv_topic_dict, clf, count_vec))

Prediced class is  sports
```

Figure 4.14: Test Prediction

Chapter 5

Discussion

There were a few challenges along the way as we pursued our goals. While there was no specific source of information to guide us through the process from start to finish, we were able to identify the next steps at every stage with information gathered from several areas. Identifying a set of steps to follow was not very straightforward because of the limited number of options available in terms of libraries and APIs. For instance, gathering resources to work with the English text could be considered low effort when compared to doing the same for Telugu text. In fact, we were not able to find dictionaries of words and stemming methods for the Telugu script. This can be attributed to the depth and complexity of Telugu as a language when compared to English. The script, the grammar and linguistic morphology are very deep. One may consider limited computing power as a challenge but for us, it did not necessarily hinder our primary focus of the project. Rather, we felt limited while experimenting with other work available. Towards the end of this project, we realized that the work in this area is interesting, and the research and student community wanting to explore or add to this work should be able to start without feeling lost. Finding a dataset large enough to train models is something we faced.

Chapter 6

Conclusion and Future Work

We strongly believe that this work has great potential for research and development. To further strengthen our work, building the morphology and the language itself is an ambitious yet impactful work. We also plan on diversifying our work by applying a transliteration method on English text to produce a text in the Telugu script. This can be helpful because pronunciations in Telugu can be spelled in English. This will be helpful in using real-world data since not all speakers prefer to be limited to one script. Additionally, we are also building a dictionary of words and categorizing them into positive and negative lists for Telugu words. For more data collection, we are developing a Twitter Bot to scrape data from a list of accounts provided. Lastly, we look forward to completing training and testing two BERT models that we experimented with. The BERT models were available publicly and were fitted with our data. For this, we also require a larger set of data which we plan on expanding. So far, we do not claim complete ingenuity towards the work we have put out. We compiled work from several areas, some of which were modified to meet our goals.

LIBRARIES

Library	Description
Indic-NLP	The goal of this Library is to build Python-based libraries for standard text processing
NLTK	It is a suite of libraries and programs for symbolic and statistical natural language pro
Pandas	It is usually imported under the pd alias.
Matplotlib	It is a collection of command-style functions that makes matplotlib work like MATLA
Numpy	NumPy is a library for the Python programming language, adding support for large, n
String	It has a set of built-in methods that can be used on Strings.
Re	This library provides a set of powerful regular expression facilities, which allows us to
Sklearn	This library features various classification, regression, and clustering algorithms, inclu
Seaborn	It is a Python data visualization library built on top of Matplotlib. It is used for maki
OS	The OS module in Python provides functions for creating and removing a directory (f
Turtle	It enables users to create pictures and shapes by providing them with a virtual canvas

REFERENCES

[1] G. Priya, J. Sultana, and U. R. Macigi, *Telugu-News-Data-Classification-Using-Machine-Learning-Approach*, 09 2021, p. 181.

[2] M. Marreddy, S. R. Oota, L. S. Vakada, V. C. Chinni, and R. Mamidi, "Clickbait detection in telugu: Overcoming nlp challenges in resource-poor languages using benchmarked techniques," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[3] ——, "Multi-task text classification using graph convolutional networks for large-scale low resource language," 2022. [Online]. Available: https://arxiv.org/abs/2205.01204

[4] A. Kunchukuttan, "The IndicNLP Library," https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf, 2020.