

CSCE 5290: Natural Language Processing Project Proposal

Project Title:

Topic Modeling and Sentiment Analysis for movie reviews

Team Members:

Gayatri Annapurna Vadlani
Harsha Preetham Pukkalla
Rakshith Dyavari Shetty
Harshitha Ambilpur

GITHUB Link:

https://github.com/Gayatri345/CSCCE5290_ProjectNLP.git

Demo Video Link:

https://unt.zoom.us/rec/play/batCMaG8nIcQo2_VHU8WBCs4uwlZewtszDlZ3KhA6KoYcnuleU8bngaN25h8G2UBygUoASdPtsxEtzfD.-9kj_t68mxO0oi-C?autoplay=true

Presentation Video Link:

https://us02web.zoom.us/rec/play/9pIW1aKVpyHJqswsTGK4Qcb_3J6ChEv05IB064kz4gaEmxhaTzvNBpSu9tRMM_r8Yo_fOloHcrblzPXg.GU0EKc4dDxD0rFAW?autoplay=true

Motivation:

The main goal of this project is to know the opinion by using reviews from a movie reviewing website and also to identify popular topics discussed from the reviews.

Movie reviews can be used to understand opinions of people or group of people. In websites like Imdb we will have number of reviews provided for each movie. By analyzing these reviews and using techniques like Topic modelling and Sentiment analysis we can know about a movie, topics that are been used, knowing the opinion of the people. We apply topic modeling to infer the different topics of discussion and sentiment analysis is applied to determine overall feelings whether a document is having positive opinion or negative opinion.

Significance:

In Natural Language Processing, sentiment analysis is one of the hot topics, where we can analyze the sentiments of individual opinions, opinions, group of people. It is used in many industries to understand the sentiment of the customers, in retail, entertainment, gaming, stock markets, housing, etc. Industries use this to understand how much impact on their services or products from the users. If a product is having more positive reviews and it can be analyzed through the Sentiment Analysis, then a organization can increase or decrease the sales of a product based on the sentiment analyzed.

Using many machine learning algorithms and techniques from NLP we can classify a document in to positive, negative or neutral to get the probabilities of opinion.

In our project we will take the reviews from IMDB dataset and classify them as 'positive' or 'negative' review. We train a model using these reviews which can predict and classify any reviews from IMDB website as positive or negative.

IMDB is a movie reviewing website, where people discusses and reviews movies, tv shows, actors, fans, etc. It provides a big database of information about movies, actors, reviews, discussions, etc. From these reviews database we can extract the topics that have been mostly discussed in such a platform.

Objectives:

In this project we focus on building models which predicts the sentiment of people and also extracts the topics from the discussions. Initially we will start with

analyzing datasets and use the preprocessing techniques, model building techniques, NLP techniques learned in this course to build our project. We want to compare different models for our application, we will start with transfer learning models to fine tuning the models. Once we develop required models, we will focus on increasing the accuracy and choose best fitting model for our application.

Features:

1.Data

Our data is gathered from Kaggle website. It is a dataset of IMDB movie reviews in the form csv files. This dataset has two columns one is 'review' and the other is 'sentiment'. Sentiments of the reviews are classified into positive or negative. It has total of 50,000 reviews, in which 25,000 reviews into positive, and 25,000. Positive are encoded into '1' and negatives are encoded '0'.

2. Topic Modelling:

Topic modelling is a unsupervised learning model which extracts the topics from the text, corpus, documents provided. When we want to know the important topics going on in a discussion and not sure of what we are looking for we can use Topic Modelling to generate topics being discussed into clusters.

In this project we want perform Topic Modelling on the reviews document, and extract Top_n topics that are been discussed. For this we are planning to use and compare two models, 1. BERT 2. LDA for topic modelling.

We achieve topic modeling using BERT (Bidirectional Encoder Representations from Transformers) which main purpose is to extract embeddings based on the context of the word. BERT is transformer model which uses 12 layers of encoders.

We will try to apply Topic Modeling for different combination of algorithms LDA and Bert. In our analysis we expect BERT to give better results than other models.



Fig.1 Flowchart of BERT

3.Sentiment Analysis:

Sentiment Analysis is widely used to classify whether the data is positive, negative or neutral. This will give the information whether the users are having good or bad opinion on a movie or product.

In this project we are planning to implement Sentiment Analysis by using SVM and LSTM models, and also use different 'loss' and 'optimizers', compare both the outputs and use the best fitting model for the final output.

We want to predict whether a review is positive ('1') or negative ('0'). We want to provide metrics of measurement for accuracy, precision, recall and generate confusion matrix for the classes predicted. For visualization we want to plot the graphs between test and train accuracies of the models.

These comparisons will help to understand which model is performing better and use the best working model.

We assume that SVM performs better as it is a transfer learning model, but we would like to fine tune our LSTM model to make it perform better than SVM and use it for final output.

The Svm model algorithm identifies the right hyperplane and segregates the classifications. In Linear SVC it returns a best fitting hyperplane to categorize the data. For this model we are planning to use transfer learning model. We want to compare SVM with LSTM which is a long-short term memory.

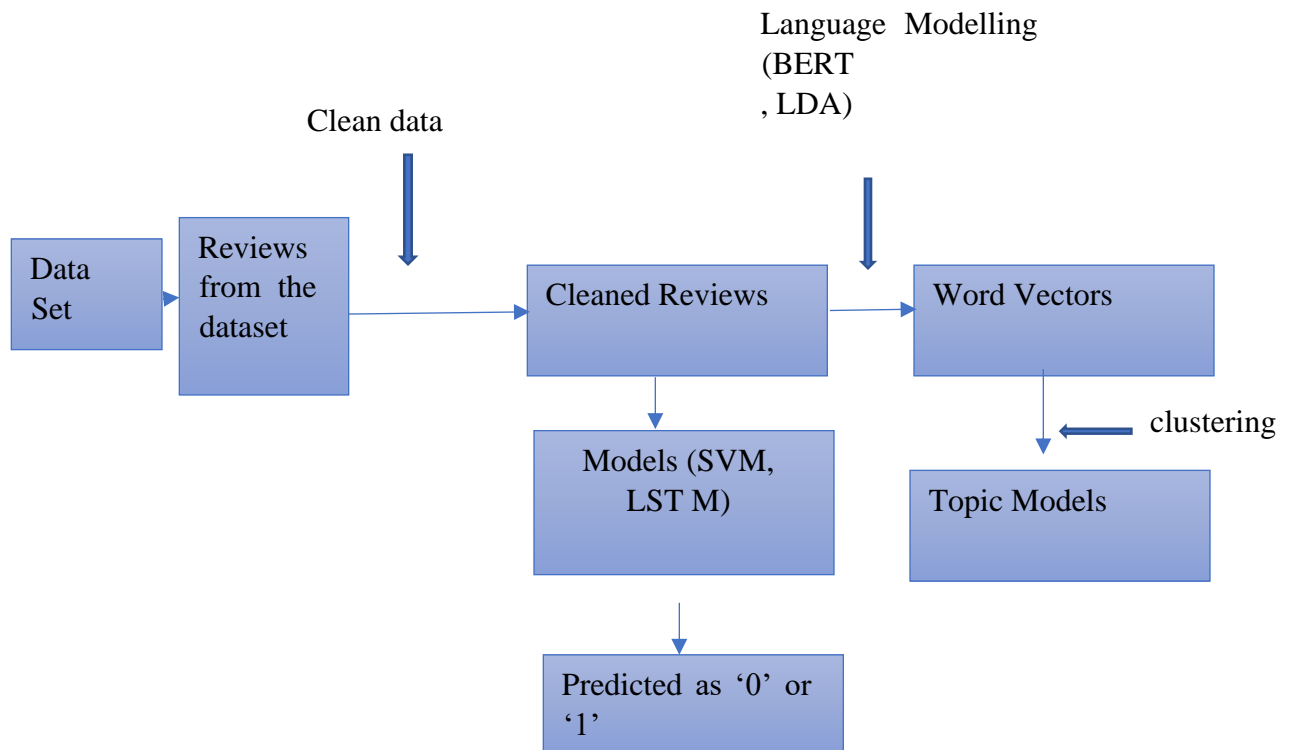


Fig.2. Flowchart

Work Plan:

SPRINTS	Module	Due
SPRINT0:	Work Plan.	Nov 2
SPRINT1:	Data Analysis, Preprocessing	Nov 6-7
SPRINT2:	Build Algorithms for Sentiment Analysis, Train and Tune	Nov 13-14
SPRINT3:	Build Algorithms for Topic Modelling, Train and Tune	Nov 13-14

SPRINT4:	Test Models performance, test on custom reviews.	Nov 20-21
SPRINT5:	Fine Tuning models	Nov 20-21
SPRINT6:	Final Delivery	Nov 27-28

Increment 1:

Dataset:

Analysis and Implementation:

We are using **IMDB dataset** from Kaggle. This dataset has ‘review and ‘sentiment’ column. We have raw form of reviews and ‘positive’ for positive review and ‘negative’ for ‘negative’ review in sentiment column.

We decided to use Kaggle version of this dataset in csv format with total of 50,000 reviews. It has 25,000 positive reviews and 25,000 negative reviews. Dataset Links [10][11]

Implementation:

1. Reading the dataset

```
[10]: df = pd.read_csv('movie_data.csv')
df.head(10)
```

```
[10]:
```

	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0
3	hi for all the people who have seen this wonde...	1
4	I recently bought the DVD, forgetting just how...	0
5	Leave it to Braik to put on a good show. Final...	1
6	Nathan Detroit (Frank Sinatra) is the manager ...	1
7	To understand "Crash Course" in the right cont...	1
8	I've been impressed with Chavez's stance again...	1
9	This movie is directed by Renny Harlin the fin...	1

We observe that the reviews need to be cleaned. So, we implemented python code using 're' to clean the data.

After cleaning the reviews.

```
}]: df.head()
```

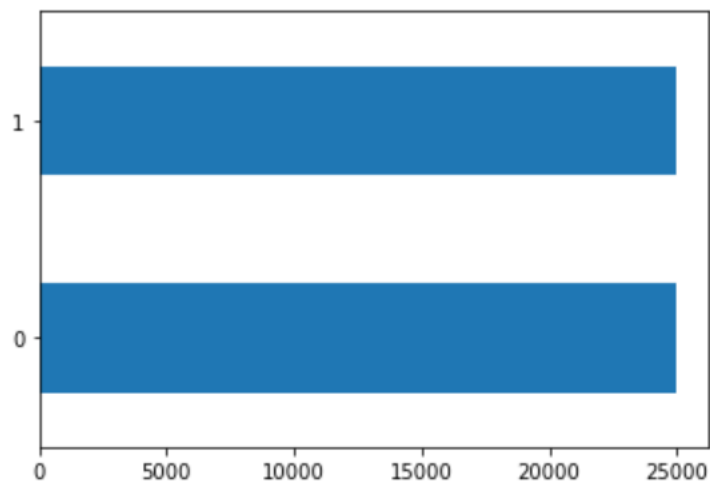
	review	sentiment
0	in 1974 the teenager martha moxley maggie grac...	1
1	ok so i really like kris kristofferson and his...	0
2	spoiler do not read this if you think about w...	0
3	hi for all the people who have seen this wonde...	1
4	i recently bought the dvd forgetting just how ...	0

Observing positive and negative reviews.

```
t', 'Available']  
[106]: df.sentiment.value_counts()  
[106]: 0    25000  
       1    25000  
       Name: sentiment, dtype: int64
```

Bar graph to analyze dataset features visually

```
.]: df.sentiment.value_counts().sort_values().plot(kind = 'barh')  
.]: <AxesSubplot:>
```



2. Sentiment Analysis:

We used LinearSVC transfer learning model for our data and predicted the accuracy, we also generated confusion matrix, precision, recall and F1 scores for this model.

This model gave us an accuracy of nearly 88%.

We want to also implement a naïve bayes approach algorithm to do the comparison. Assuming Naïve Bayes model will have very less accuracy generated.

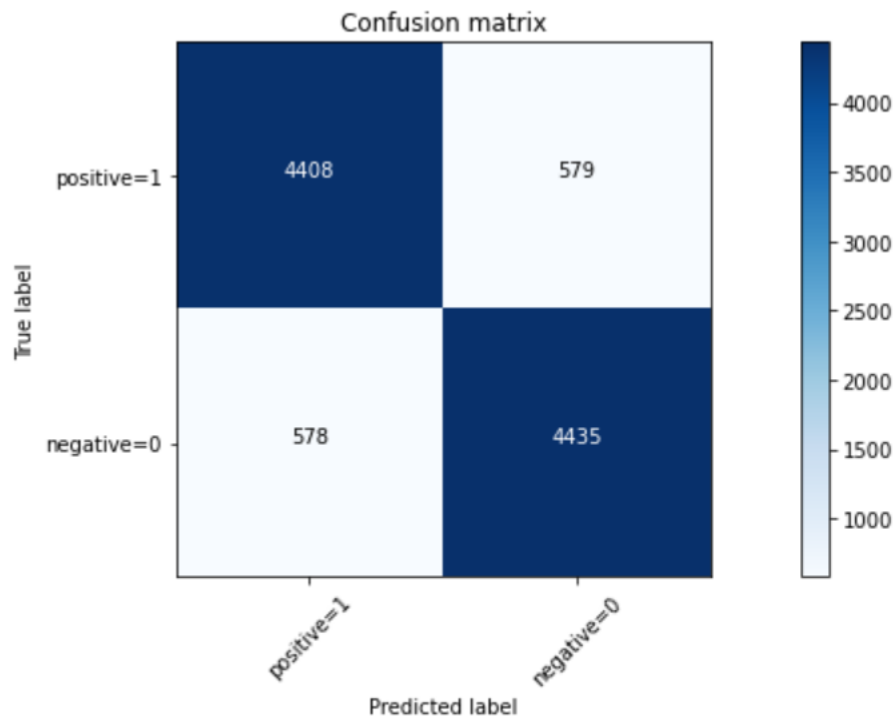
We also want to experiment with LSTM and fine tuning them to achieve accuracy as good as transfer model SVM.

Below image is the classification report for LinearSVC

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.88	0.88	5013
1	0.88	0.88	0.88	4987
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

Confusion matrix generated:



3. Topic Modelling:

After going through several topic modelling techniques, we have decided to use BERT for our project. For our initial model we generated a list from all the reviews to extract the topics. The model we used for our primary analysis is BERTopic model for our initial understanding. We generated topic frequencies, which indeed generated 772 topics for our dataset (from -1 to 770).

Below is the screenshot for the topic frequencies and Intertopic Distance map:

```
| model.get_topic_freq()
```

	Topic	Count
0	-1	24017
1	0	675
2	1	635
3	2	383
4	3	327
...
756	769	10
755	765	10
754	764	10
752	766	10
...



Each of the topic here is having nearby words, for example Topic 1 has words like worst, waste, horrible, terrible, awful. We want to implement LDA model to check how the topics are categorized and use the best model for Topic Modelling. We assume that the results from BERT will be more accurate than LDA model. We plan to experiment with all the above-mentioned models and generate the graphs for visualization of topics and compare all the models.

✓ **Project Management:**

➤ **Implementation status report**

• Work completed:

We have finalized the dataset and cleaned the data. Also built a simple LinearSVC model for Sentiment Analysis and BERT model for Topic Modelling.

Responsibility (Task, Person)

Background and references: Everyone

Data Set: Gayatri

Sentiment Analysis: Gayatri and Rakshith

Topic Modelling: Harsha and Harshitha
Writing and editing: Everyone

- *Contributions* (members/percentage):

Harsha: 30%

Harshita: 20%

Gayatri: 30%

Rakshith:20%

- *Work to be completed*

- *Description:*

Building RNN LSTM model for sentiment Analysis.

Comparing the results, fine tuning it and generating confusion matrix, predictions, classification report and compare with the SVM model. Fine tuning the model to achieve accuracy as SVM for sentiment Analysis part.

For Topic Modelling build an LDA model and compare the results with BERT Model. After that we will use our models on custom reviews of different kinds (like complicated review which are difficult for a model to classify whether positive or negative) and check for the predicted outputs on working model.

INCREMENT-2

Introduction:

Sentiment Analysis and Topic modelling are widely used in the industries like retail, entertainment, etc., to determine the opinions, Topic modelling helps to identify abstract topics that occur in a collection of documents to determine what the people are thinking about a particular movie, Sentiment Analysis helps us to identify the opinion of the people whether a document is positive or negative score.

We want to implement our project for movie reviewing. Sentiment Analysis can be used to understand the opinions of the people from the reviews and Topic modelling can be used to generate the topics from the reviews.

We want to create an application which identifies a review as positive or negative using. We also want to generate the topics from the reviews to visualize.

For this we planned to implement Sentiment Analysis on IMDB dataset using different models and analyze the model performance and finally check the output with real-time data.

For generating topics using Topic Modelling we planned to use two different models and analyze how the topics are from the reviews and how they are categorized in both.

In this increment we continue to work on our goals of the project. Once we started working, we shifted our focus to more on sentiment analysis, which is providing us more and interesting scope to experiment, rather than topic modelling. The basic idea of our Topic modelling is generating common topics when we do web scrapping, after we started using cleaned dataset, more analysis was performed on sentiment analysis module. In the process of tuning and developing models for sentiment analysis, pretraining models like Bert, LDA for Topic Modelling we encountered many hardware issues, runtime issues, memory problems, limited coding knowledge, limited knowledge on tuning machine learning models, reaching target in limited time. We extended our efforts to provide the best possible outcomes.

Out of our interest to experiment, after some more extended research on this topics we increased our scope of Sentiment Analysis module to compare models with Naïve Bayes, SVM, LSTM, LSTM+CNN rather than SVM and LSTM.

Background:

There have been many research and papers presented in the field of sentiment analysis and topic modelling on the data. These topics always provide us with wide range of experiments in the field of NLP. We started researching the papers related to our basic idea for comparing SVM and LSTM models for sentiment analysis. This research is very useful and made us perform interesting experiments by increasing the scope of the project for comparing Naïve Bayes, SVM, LSTM and LSTM+CNN models for sentiment Analysis.

One of the papers that influenced us to work on LSTM+CNN is by Ahmad Fathan Hidayatullah [1], the basic idea of this paper is to perform sentiment analysis on

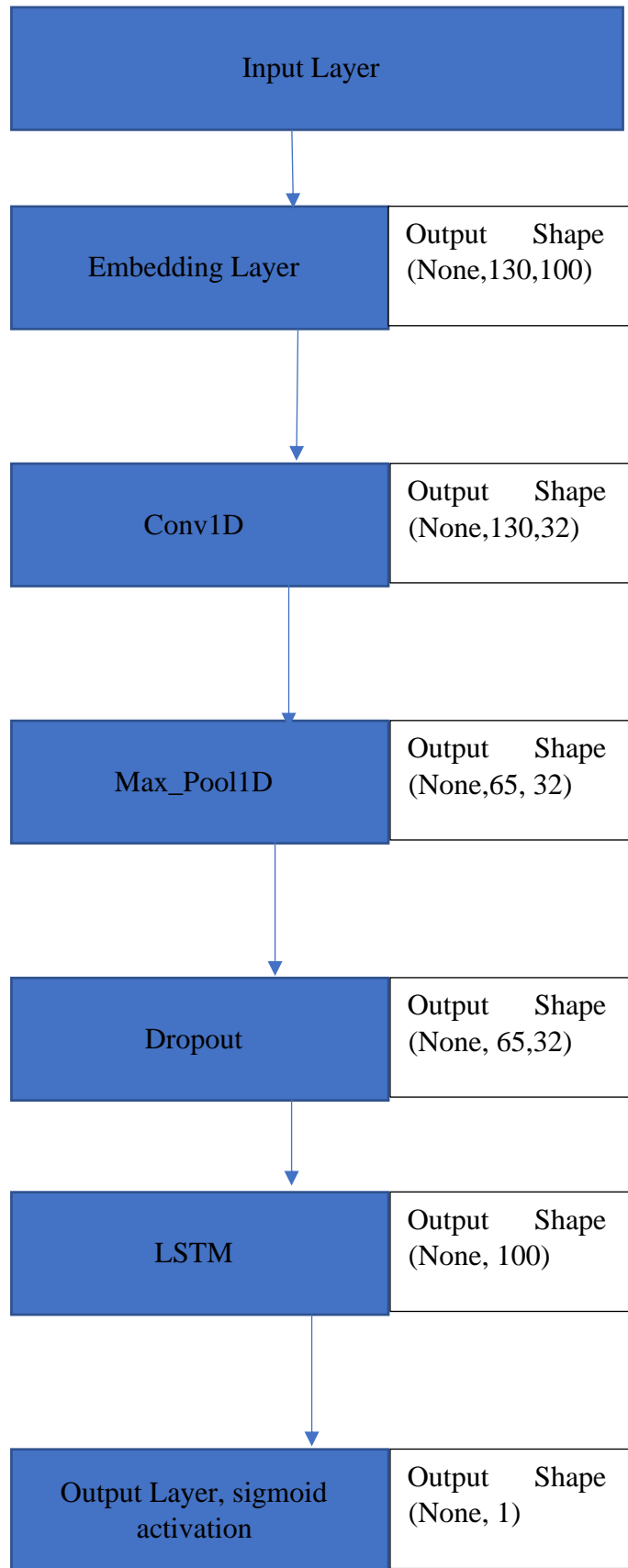
twitter dataset and classify the sentiment as positive or negative. In the approach this paper drives to interesting experiments with LSTM, CNN, CNN+LSTM. This paper also discusses about the Naive Bayes approach and SVM approaches. One more paper that gave us more insights in to developing models with LSTM+CNN is [2] by J. Shobhana and M. Murali where they discussed about efficient methodology based on LSTM networks. In this paper they also discussed about the feature engineering part of the model which is deeper in analysis that uses skip-gram based word embeddings. The final evaluation of this paper provide comparison between trained model using SVM, LSTM, ANN and APSO-LSTM. One interesting paper that contributes in the area of Topic Modelling is that [3] by Natalie Cyagan, which provides the discussions between topic modelling using BERT, LDA and SBERT. This paper discusses how SBERT is helpful to develop better document embeddings.

Model:

Architecture Diagram:

Sentiment Analysis:

For sentiment analysis we experimented with different architectural models like Naïve Bayes, linearSVC (which is svm model), LSTM, LSTM+CNN. We finally choose our LSTM+CNN model as the final model which is fitting the data efficiently and giving competitive results compared with other tuned models. The basic architecture of our tuned model looks like below.



Our model architecture for LSTM+CNN basically consists of an input layer with input of size '130'. That means all the inputs should be converted to 130 size word vectors to pass to the model. We are now using Conv1D filter with filters of '32' in number and filter size '3x3' and activation unit as 'relu'. This convolution layer generates the feature maps of size '3x3'. Maxpooling layer is the pooling layer which extracts the maximum size of the features from the convolution layer. We are using Dropout layer which regularizes the overfitting of the model with 50% of dropout. The next is the LSTM layer with 100 units is used. Now we have a output layer with 'sigmoid' activation unit which gives the probabilities of positive or negative review. We now compiled our model with Adam optimizer and hyper tuning with parameters like 'learning rate' of 0.0001, beta_1, beta_2, epsilon(for smoothening). In our study we got to know that Adam optimizer is used in most of the NLP tasks which improves performance with learning rate.

Below figure shows the model summary of our final model.

Model: "sequential_15"

Layer (type)	Output Shape	Param #
=====		
embedding_15 (Embedding)	(None, 130, 100)	6474100

conv1d_5 (Conv1D)	(None, 130, 32)	9632

max_pooling1d_4 (MaxPooling1	(None, 65, 32)	0

dropout_9 (Dropout)	(None, 65, 32)	0

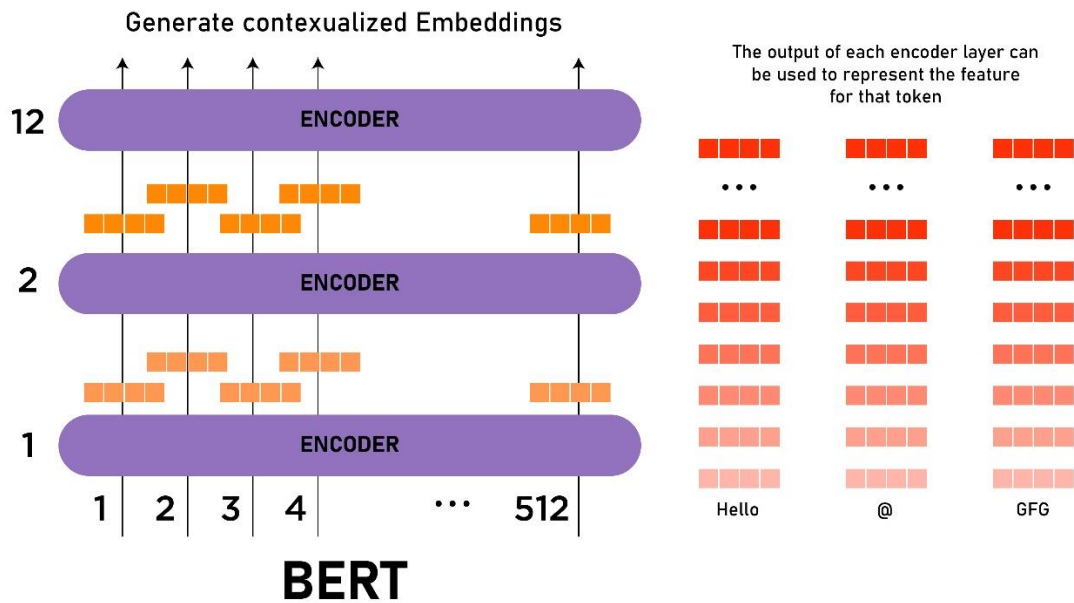
lstm_21 (LSTM)	(None, 100)	53200

dense_21 (Dense)	(None, 1)	101
=====		
Total params: 6,537,033		
Trainable params: 6,537,033		
Non-trainable params: 0		

None		

Topic Modelling BERT:

As BERT is a pre trained model, where we use our data to fit the model, we took the reference of this blogpost to understand BERT architecture [4]. This architecture diagram clearly explains us how the topics are generated. Bert is an encoder stack of transformer architecture. It basically has encoder-decoder network that uses attention on both the sides. The architecture explains that it is an encoder with 12 layers, generates contextual embedding from the input documents and the output of each encoder layer can be used to represent the features of the tokens.



Workflow Diagram:

Below diagram explains the basic workflow of our models. First, we need to collect data, analyze it and do the preprocessing of data as required to fit in to the models. Then we use machine learning algorithms for sentiment analysis and Topic modelling to achieve our results. Finally, we analyze the sentiment of real time data for sentiment analysis which is predicted as negative or positive and topics generated for the dataset, we have given in Topic modelling stage.

Step 1: Collect the data required for models.

Step 2: Clean and preprocess the data.

Following are some of the steps we followed for cleaning and preprocessing.

- Cleaning special characters
- Cleaning numbers
- Lower casing
- Remove stop words
- Lemmatize

Step 3: Generate Features

Features we generated are different types for different models.

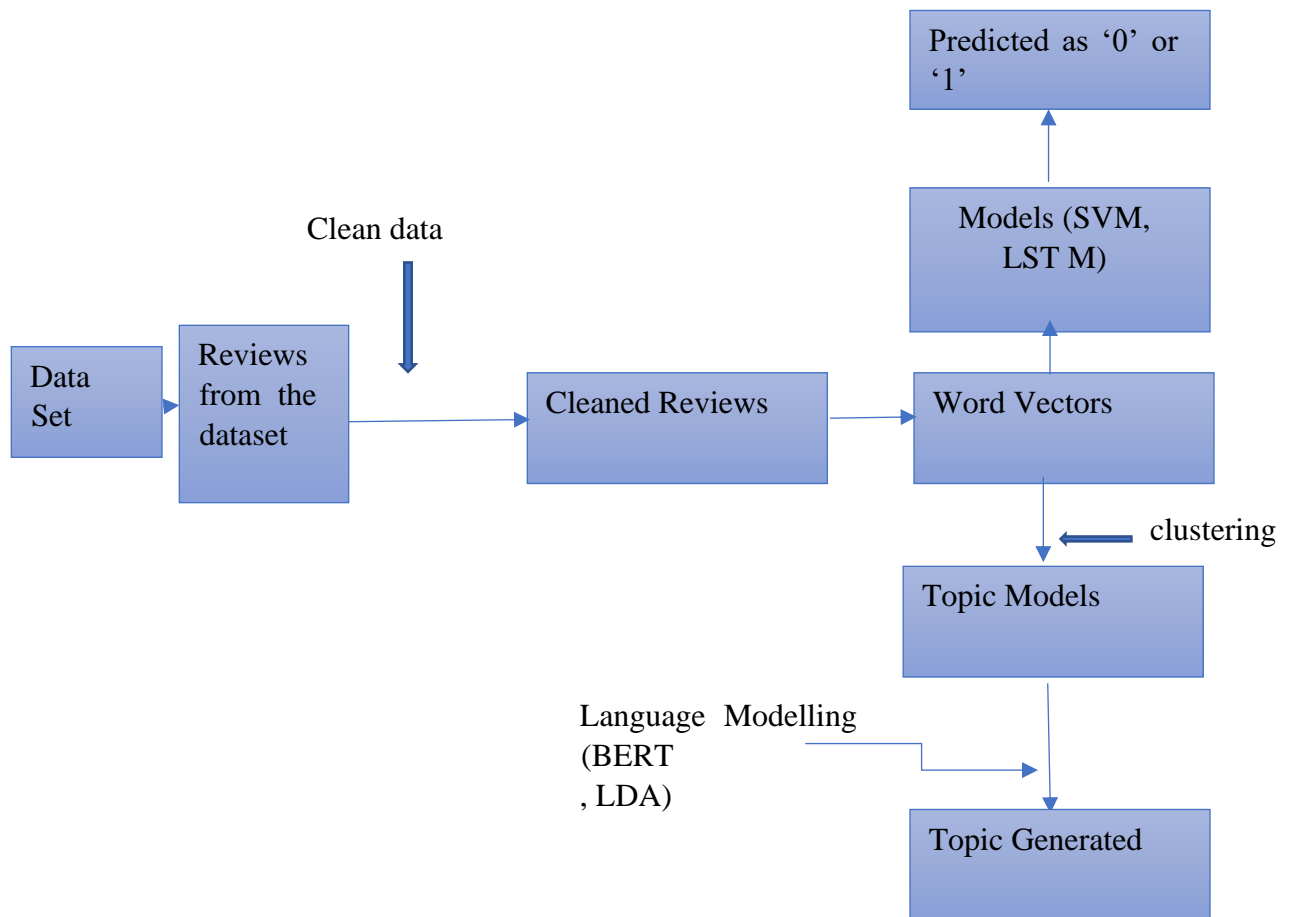
- Vector embeddings
- BoW vector representations
- Tfidf vector representations

Step 4: Now we have features that are necessary to train our models.

We use them to train all our sentiment analysis models.

Step 5: Analyze, Evaluate, Validate the output of sentiment analysis model.

Step5: Train the Topic modelling models with generated features and analyze the output from the models with topics generated.



Dataset:

Detailed description of Dataset:

The dataset we used is from Kaggle source [5] is a form csv file, with 50,000 reviews. 25,000 are for positive reviews and 25,000 are for negative reviews. This

file has two columns one is 'review' and one is respective 'sentiment' score of the review.

We analyzed this dataset in our first increment whose histogram shows us 25,000 reviews for each sentiment score. That means this dataset is good to use without any further preprocessing required.

Below figure shows the data in our dataset which is read in the form of csv file.

```
[234]: #Reading the data in csv file in to dataframe
df = pd.read_csv('Data/movie_data.csv')
```

```
[235]: #Exploring data
df.head()
#df.shape
```

```
[235]:
```

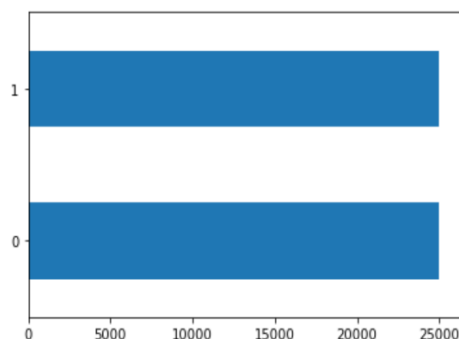
	review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1
1	OK... so... I really like Kris Kristofferson a...	0
2	***SPOILER*** Do not read this, if you think a...	0
3	hi for all the people who have seen this wonde...	1
4	I recently bought the DVD, forgetting just how...	0

Detail Design of Features:

Below is the histogram plot to visualize the sentiment values or features of the dataset, total number of reviews in the dataset. This dataset has two feature classes, one is 'positive' represented by '1' and one is 'negative' represented by '0'.

```
.]: df.sentiment.value_counts().sort_values().plot(kind = 'barh')
```

```
.]: <AxesSubplot:>
```



```
t', 'Available']
```

```
[106]: df.sentiment.value_counts()
```

```
[106]: 0    25000
      1    25000
      Name: sentiment, dtype: int64
```

Analysis of Data:

Data Pre-Processing:

We developed our preprocessing of input in three stages as follows:

1. Cleaning of the data
2. Stemming input
3. Removing stop words.

Cleaning of the data:

In this stage we remove 'special characters', 'digits', 'white spaces', 'double spaces', etc., for this we are 'regex' library. 'Unidecode' is also used to represent repetitive string froms in Unicode form. For example, if text has '?????', to know the context of the text we don't need this. Unidecode helps in dealing with such kind of strings. Below screenshot shows our function with regex and unidecode.

```
[236]: #After many data cleaning techniques we merged all our ideas into this function a
def clean_data(text,remove_polish_letters):

    text = remove_polish_letters(text)
    text = str(text)
    text = text.lower()

    # Clean the text
    text = sub(r"^[A-Za-z0-9^,!?.\/'"]", " ", text)
    #Removes special characters
    text = sub(r"\+", " plus ", text)
    text = sub(r",", " ", text)
    text = sub(r"\.", " ", text)
    text = sub(r"!", " ! ", text)
    text = sub(r"\?", " ? ", text)
    text = sub(r"\"", " ", text)
    #Removes : with and without spaces
    text = sub(r":", " : ", text)
    #Removes matches more than 2 white spaces
    text = sub(r"\s{2,}", " ", text)
    #Removes digits
    text = sub(r"\d+", " ", text)

    #text = text.split()

    return text
#Applying clean_data to every review in our dataset
df.review = df.review.apply(lambda x: clean_data(x, unidecode))
```

Below image shows before and after output of data passing through 'clean_data' function.

	review	sentiment		review	sentiment
0	In 1974, the teenager Martha Moxley (Maggie Gr...	1	0	in the teenager martha moxley maggie grace mo...	1
1	OK... so... I really like Kris Kristofferson a...	0	1	ok so i really like kris kristofferson and his...	0
2	***SPOILER*** Do not read this, if you think a...	0	2	spoiler do not read this if you think about w...	0
3	hi for all the people who have seen this wonde...	1	3	hi for all the people who have seen this wonde...	1
4	I recently bought the DVD, forgetting just how...	0	4	i recently bought the dvd forgetting just how ...	0

Stemming input:

Now we want to stem our data for converting words like 'writing' to write, 'stopping' to 'stop'. Only the stems should be preserved. For this we used porter stemmer. Below is the screenshot of the function we are using. This applies stemmer to all the reviews.

```
#Stemming the text
def simple_stemmer(text):
    ps=nlk.porter.PorterStemmer()
    text= ' '.join([ps.stem(word) for word in text.split()])
    return text
#Apply function on review column
df['review']=df['review'].apply(simple_stemmer)
```

The output of the sample review before and after stemming looks like this, sample stemming is highlighted in the images.

Before stemming:

```
df['review'][0]
```

'in the teenager martha moxley maggie grace moves to the high class area of belle haven greenwich connecticut on t he mischief night eve of halloween she was murdered in the backyard of her house and her murder remained unsolved t wenty two years later the writer mark fuhrman christopher meloni who is a former la detective that has fallen in di sgrace for perjury in o j simpson trial and moved to idaho decides to investigate the case with his partner stephen weeks andrew mitchell with the purpose of writing a book the locals squirm and do not welcome them but with the sup port of the retired detective steve carroll robert forster that was in charge of the investigation in the s they d iscover the criminal and a net of power and money to cover the murder br / br / murder in greenwich is a good tv mo vie with the true story of a murder of a fifteen years old girl that was committed by a wealthy teenager whose moth er was a kennedy the powerful and rich family used their influence to cover the murder for more than twenty years h owever a snoopy detective and convicted perjurer in disgrace was able to disclose how the hideous crime was committ ed the screenplay shows the investigation of mark and the last days of martha in parallel but there is a lack of th e emotion in the dramatization my vote is seven br / br / title brazil not available'

After stemming:

```
df['review'][0]
```

```
'in the teenag martha moxley maggi grace move to the high class area of bell haven greenwich connecticut on the mis  
chief night eve of halloween she wa murder in the backyard of her hous and her murder remain unsolv twenti two year  
later the writer mark fuhrman christoph meloni who is a former la detect that ha fallen in disgrac for perjuri in o  
j simpson trial and move to idaho decid to investig the case with hi partner stephen week andrew mitchel with the p  
urpos of write a book the local squirm and do not welcom them but with the support of the retir detect steve carrol  
robert forster that wa in charg of the investig in the s they discov the crimin and a net of power and money to cov  
er the murder br / br / murder in greenwich is a good tv movi with the true stori of a murder of a fifteen year old  
girl that wa commit by a wealthi teenag whose mother wa a kennedi the power and rich famili use their influenc to c  
over the murder for more than twenti year howev a snoopi detect and convict perjur in disgrac wa abl to disclos how  
the hideou crime wa commit the screenplay show the investig of mark and the last day of martha in parallel but ther  
e is a lack of the emot in the dramt my vote is seven br / br / titl brazil not avail'
```

Removing stop words.

Now we can see that this review is having a lot of stop words. So, removing them makes our task of extracting context more efficient. For this we are using nltk corpus stopwords library, which provides with list of 'english' stop words that can be removed. This is how this output after removing stop words looks like. In previous image we can see words like 'in', 'the', 'to'. Now this data is cleaned from stopwords.

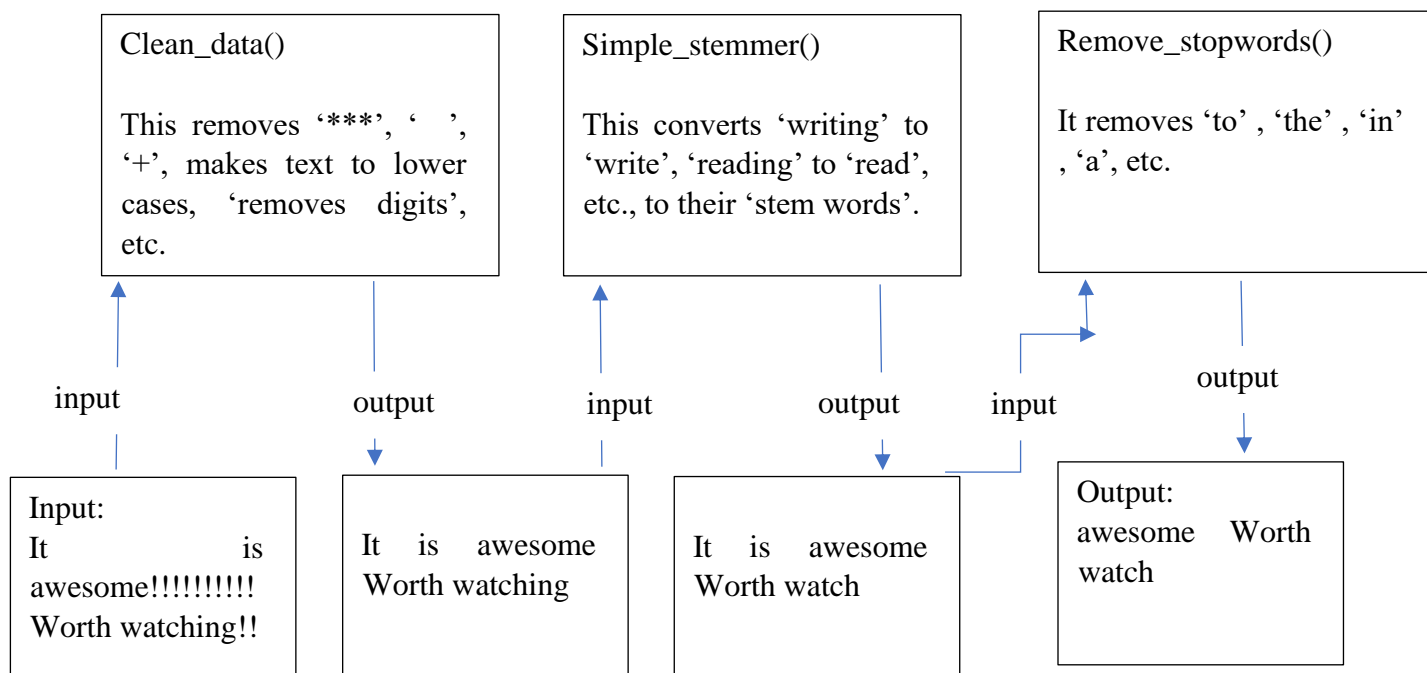
```
@0: df['review'][0]
```

```
@0: 'teenag martha moxley maggi grace move high class area bell greenwich connecticut mischief night eve halloween wa m  
under backyard hous murder remain unsolv twenti two year later writer mark fuhrman christoph meloni former la detec  
t ha fallen disgrac perjuri j simpson trial move idaho decid investig case hi partner stephen week andrew mitchel p  
urpos write book local squirm welcom support retir detect steve carrol robert forster wa chang investig discov crim  
in net power money cover murder br / br / murder greenwich good tv movi true stori murder fifteen year old girl wa  
commit wealthi teenag whose mother wa kennedi power rich famili use influenc cover murder twenti year howev snoopi  
detect convict perjur disgrac wa abl disclos hideou crime wa commit screenplay show investig mark last day martha p  
arallel lack emot dramt vote seven br / br / titl brazil avail'
```

This data looks good for extracting the context or sentiment. All the unnecessary words have been cleaned or removed from all the reviews in our dataframe.

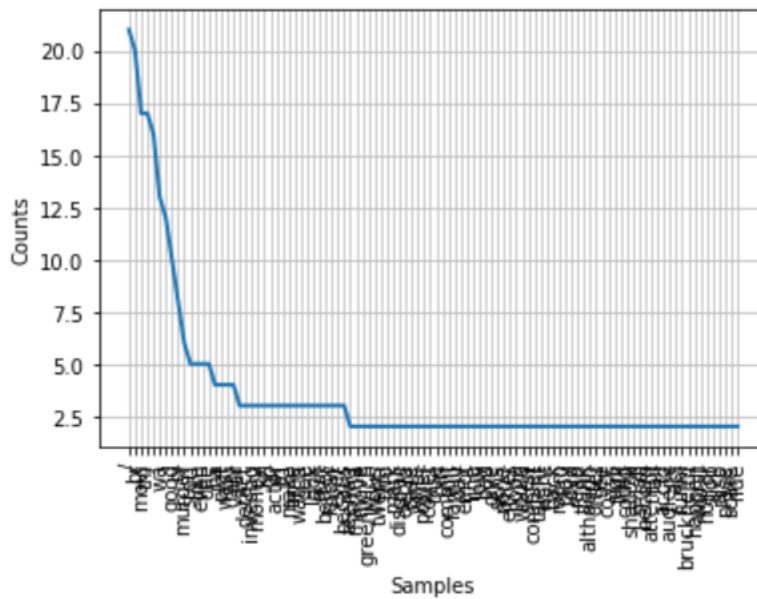
Graph model with explanation:

This removes all the stopwords in 'english' library of nltk corpus stopwords. This how the model works with example input. Overall, it reduces the final length of the review after passing all the stages.



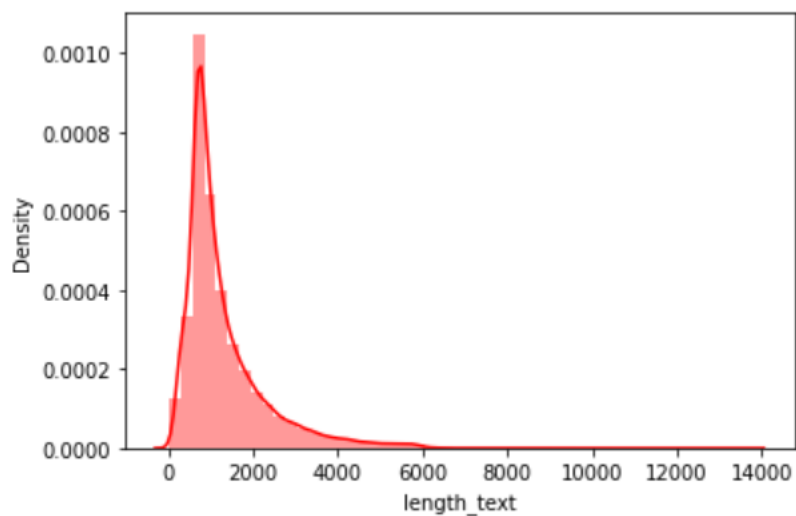
The below image shows the frequency distribution of top 100 words in the corpus. This gives the plot between samples of data vs number of counts of the data.

```
|: freq_plot(100)
```



Below graph shows the length and density of the texts in out dataframe

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
```



Algorithms/Pseudocode:

Pseudocode:

Preprocess data

Convert into vectors

Define the model

Compile the model

Evaluate

Make predictions on test data

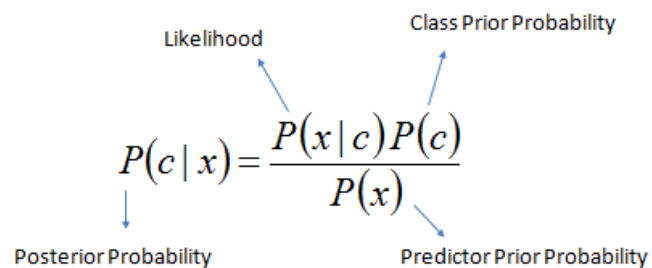
Input real-time data

Preprocess the real-time data

Use the trained model for predictions.

Naïve Bayes:

Naïve Bayes algorithm is a classification technique. In this algorithm all the features are considered as independent. That means any feature occurrence is independent of any other features in the corpus. In Bayes theorem we calculate Posterior Probability. Image source [7]. So, this probability returns the class of the feature belonging to.



The diagram shows the equation $P(c | x) = \frac{P(x | c)P(c)}{P(x)}$ with four blue arrows pointing to its components: 'Likelihood' points to $P(x | c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c | x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Svm(Linear svc):

The Svm model algorithm identifies the right hyperplane and segregates the classifications. In Linear SVC it returns a best fitting hyperplane to categorize the data.

LSTM:

It is a type of recurrent neural network, that is capable of predicting in sequence. LSTM are transfer based learning models. The traditional RNN's have short-term memory problem, they don't remember which word comes nearby, which is known as vanishing gradient problem. LSTM comes up with implementing new cell state called long term memory. This long-term memory stores the keywords, which helps in generating the result of the predictions. During the training process LSTM decides what to discard and what to store in its long-term memory. 'Forget gate' uses sigmoid to forget previous state. 'Input gate' uses sigmoid and tanh and multiply both of them to add memory. In 'output gate' we take weighted sum of hidden state and apply sigmoid function product it with memory state with tan function which gives hidden state. Topic referred from article [8].

LSTM+CNN:

CNN LSTM uses CNN for feature extraction and then combined with LSTM to support sequence prediction. The sequence generally looks like

Input layer, cnn model layers, lstm model layers, dense layer, output layers.

BERT for Topic modelling:

BERT is a transformer-based algorithm, it takes attention mechanism which learns relations between words and subwords, this indeed generates the predicted topics with these relations.

LDA for Topic modelling:

LDA builds Dirichlet Distributions. This builds Topic for document model and words for topic model. It is a unsupervised learning algorithm and a probabilistic model, to assign topics to the clusters it uses $P(\text{word} \mid \text{topics})$ and $P(\text{topics} \mid \text{documents})$.

Explanation of Implementation:

Features for the model:

Once we are done with the preprocessing, we will generate the features requires for our models.

We are generating three types of features for our model,

1. *Bag of words* vector representation

For this we use count vectorizer, which transfers each word into vector representation based on the frequency of word occurrences

Below image shows sample features of text generated by countvectorizer and these will be converted into respective vectors.

```
#print(vec.get_feature_names())
#print(len(vec.get_feature_names()))

[85]: CountVectorizer(max_df=0.8, min_df=0.03, stop_

[88]: #Ref:https://scikit-learn.org/stable/modules/c
#The first 10 generated features by countvecto

[89]: vec.get_feature_names()[0:10]

[89]: ['abl',
      'abov',
      'absolut',
      'act',
      'action',
      'actor',
      'actress',
      'actual',
      'ad',
      'add']
```

When we transform the matrix we get the sparse mat

And this is how they are assigned with frequencies.

```
108]: print(x_train_transformed[0])
```

```
(0, 0)      1
(0, 54)     1
(0, 57)     4
(0, 71)     1
(0, 86)     1
(0, 106)    2
(0, 116)    1
(0, 121)    1
(0, 134)    1
(0, 148)    1
(0, 176)    1
(0, 205)    1
(0, 209)    1
(0, 216)    1
(0, 230)    1
(0, 231)    1
(0, 243)    1
```

Topics and their frequency of words:

```
2]: #This gives the representation of topics and words in the topics.
pd.DataFrame(x_train_transformed.toarray(), columns=vec.get_feature_names_out()).head()
```

```
2]:
```

	abl	abov	absolut	act	action	actor	actress	actual	ad	add	...	wors	worst	worth	write	writer	written	wrong	ye	year	yc
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	1	1	0	0	0	3	
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
2	0	0	0	1	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	

5 rows × 557 columns

2. Tfidf word vectors:

It performs the product of term frequency and inverse document frequency and assign frequencies to the word. For this we use TFIDF vectorizer.

Below is the sample features generated by tfidf vectorizer,

Tfidf features:

```
[120]: tfidf.get_feature_names()[0:10]
```

```
[120]: ['aaron',  
        'abandon',  
        'abc',  
        'abduct',  
        'abil',  
        'abl',  
        'aboard',  
        'abomin',  
        'abort',  
        'abound']
```

```
[121]: v_train_tf_idf
```

Tfidf vectors:

This shows that it is calculating product of term frequency and inverse document frequency.

```
! = print(X_train_tf[0])  
(0, 299) 0.081377791706589823  
(0, 532) 0.10302165627172809  
(0, 4526) 0.061242867423280264  
(0, 3945) 0.08848313092234944  
(0, 4810) 0.08454041261998095  
(0, 1331) 0.07433214925032057  
(0, 1454) 0.06221772907311682  
(0, 2525) 0.060222820804650676  
(0, 3195) 0.09748089233920494  
(0, 1116) 0.04915121899490374  
(0, 2542) 0.05219458408497249  
(0, 4000) 0.04063208256131472  
(0, 3877) 0.0757752543540788  
(0, 1035) 0.07372208578134022  
(0, 2098) 0.10578478291218843  
(0, 5) 0.06479488063273464  
(0, 968) 0.09322177521233396  
(0, 2179) 0.050736204732643356  
(0, 2293) 0.08572952782017006  
(0, 4725) 0.045785591695282804  
(0, 1623) 0.05548759739833852  
(0, 3712) 0.07726976594956397
```

3. Tokenizer

This helps us to transfer words to vectors, it basically converts each text into sequence of integers. Here we used padding to convert all the data in to equal lengths.

Below is the image of encoded tokens with equal length padding using tokenizer.

```

]: #We calculated max_length to make our encodings i
print('Encoded X Train\n', X_train_lstm, '\n')
print('Encoded X Test\n', X_test_lstm, '\n')
print('Maximum review length: ', max_length)

```

Encoded X Train

```

[[ 694 3163 17588 ... 2452 314 324]
 [ 526 22 8 ... 241 341 506]
 [ 574 186 2 ... 2 3 36]
 ...
 [ 139 2 4 ... 0 0 0]
 [ 2 5770 2823 ... 1 1 530]
 [ 1920 1097 847 ... 0 0 0]]

```

Encoded X Test

```

[[ 1498 3883 33 ... 0 0 0]
 [ 1506 3046 3392 ... 0 0 0]
 [ 71 2 4 ... 77 45 14367]
 ...
 [ 18 44 175 ... 0 0 0]
 [ 680 3595 61 ... 0 0 0]
 [ 407 122 15 ... 0 0 0]]

```

Apart from this we used doc2word conversions to generate bag of words in LDA and Bert embedding in bert model.

Now after generating our feature vectors, we are ready to train the models. Models we choose to use for sentiment analysis are,

1. Naïve Bayes
2. Svm
3. LSTM
4. LSTM+CNN

For visualizing topics in Topic Modelling module, we used BERT and LDA

For implementation in Naïve Bayes and SVM we used bag of *words vectors* and *tfidf vectors*. We analyzed the output and generated evaluation metrics for both of the models with these combinations.

For all the models we generated train, test sets by using sklearn train_test_split function. We have train test sets for both tfidf vectors and bag of word vectors.

Below image show the train set that transformed in to vectors using Bag of words model.

For train set:

```
[47]: x_train_transformed=vec.transform(X_train)
      x_train_transformed

[47]: <40000x560 sparse matrix of type '<class 'numpy.int64'>'
      with 1820907 stored elements in Compressed Sparse Row format>

[48]: #converting transformed matrix to an array
      x_train=x_train_transformed.toarray()

[51]: print(x_train_transformed[0])

      (0, 1)          1
      (0, 56)         1
      (0, 59)         4
      (0, 73)         1
      (0, 88)         1
      (0, 108)        2
      (0, 118)        1
      (0, 123)        1
      (0, 136)        1
```

Below image show the train and test sets generated for tfidf vectors.

```
58]: X_test_tf = tfidf.transform(X_test)

93]: tfidf.get_feature_names()[50:60]

93]: ['actual',
      'ad',
      'adam',
      'adapt',
      'add',
      'addict',
      'addit',
      'address',
      'adequ',
      'adjust']

60]: X_train_tf.shape

60]: (40000, 5000)

61]: print(X_train_tf[0])

      (0, 301)      0.08141496769690561
      (0, 534)      0.10306856110180874
      (0, 4527)     0.06125733410048056
```

Naïve Bayes:

In this approach we used multinomial naïve baye model, which uses ‘alpha’ as a default smoothing technique, so avoiding zero probability scenarios. For this model we have performed, training, prediction and evaluation. Below image shows train and test of naïve bayes model on BoW model vectors.

```
Training and Predicting

[64]: #fitting the model on training data of BoW
mnbnv_Bow = mnbn.fit(x_train_transformed,y_train)
ypred_Bow = mnbn.predict(x_test_transformed)

[65]: print("Predicted array for BoW",ypred_Bow)
print("Original array for BoW ",y_test)

Predicted array for BoW [1 0 1 ... 0 0 1]
Original array for BoW [1 0 1 ... 0 0 1]

[67]: #Getting the probability of predictions

probability_prediction=mnbnv_Bow.predict_proba(x_test_transformed)
probability_prediction
#The above output shows probabilities of prediction for '0' and '1'
print("Probability of first prediction is ",np.argmax(probability_prediction[0]))
#1 is positive review
#0 is negative review

Probability of first prediction is 1

[68]: probability_prediction[:10]

[68]: array([[1.22943167e-01, 8.77056833e-01],
 [5.06424790e-01, 4.93575210e-01],
 [1.52305895e-01, 8.47694105e-01],
```

We also did apply it on tfidf vectors.

```
[69]: #fitting the model on training data of tfidf
mnbnv_tfidf = mnbn.fit(X_train_tf,y_train)
ypred_tf = mnbn.predict(X_test_tf)

[70]: print("Predicted array for tfidf",ypred_tf)
print("Original array for tfidf ",y_test)

Predicted array for tfidf [1 1 1 ... 0 0 1]
Original array for tfidf [1 0 1 ... 0 0 1]

[71]: #Getting the probability of predictions

probability_prediction=mnbnv_Bow.predict_proba(X_test_tf)
probability_prediction
#The above output shows probabilities of prediction for '0' and '1'
print("Probability of first prediction is ",np.argmax(probability_prediction[0]))
#1 is positive review
#0 is negative review

Probability of first prediction is 1

[72]: probability_prediction[:10]

[72]: array([[0.18464421, 0.81535579],
 [0.49837921, 0.50162079],
 [0.31664862, 0.68335138],
 [0.72284833, 0.27715167],
 [0.3310793 , 0.6689207 ]]
```

SVM (Linear SVC ()):

To conduct experiments with SVM we used Linear svc model from scikit learn svm models. Linear svc uses penalty which gives us a competitive result for comparison. It has 'l1' and 'l2' penalties, by default penalty is set to 'l2'.

Below is the image for implementation of linear svc().

Training & Predicting

```
3]: #svm model for Bow
svm_bow=svm.fit(x_train_transformed,y_train)
y_pred_svm = svm_bow.predict(x_test_transformed)

4]: #svm model for Tfidf
svm_tf=svm.fit(X_train_tf,y_train)
y_pred_svm_tf=svm_tf.predict(X_test_tf)

5]: print("Predicted values by svm Bow",y_pred_svm)
print("Original values by svm Bow",y_test)

print("Predicted values by svm Tfidf",y_pred_svm_tf)
print("Original values by svm Tfidf",y_test)

Predicted values by svm Bow [1 1 1 ... 0 1 1]
Original values by svm Bow [1 0 1 ... 0 0 1]
Predicted values by svm Tfidf [1 1 1 ... 0 0 1]
Original values by svm Tfidf [1 0 1 ... 0 0 1]
```

LSTM:

For LSTM model we used sequential models and layers from tensorflow keras models. We implemented the sequential model as below.

For LSTM model inputs we used word to vector embedding using Tokenizer and padding them to equal length. All our models that we trained from initial project days we have trained them with input of sequence length '130'. So, this model takes only inputs of length 130.

The final model we used for LSTM is having Embedding layer with predefined input dimensions. Two LSTM layers are used one with '50' units and other with '25' units, followed by 'dense' layer with 'relu' activation unit and a dropout unit with '50%' dropout, final output layer is having 'sigmoid' activation units, which helps in prediction. We used optimizers as 'Adam', loss as 'Binary cross entropy', we hyper tuned it with learning rate, beta and epsilon values. This model has given us an accuracy of '87%'. Below image shows the architecture of the model.

```

|: #Building our model
   #Architecture

   # ARCHITECTURE
   EMBED_DIM = 100
   LSTM_OUT = 50

   model = Sequential()
   model.add(Embedding(total_words, EMBED_DIM, input_length = max_length))
   model.add(LSTM(LSTM_OUT,return_sequences = True))
   model.add(LSTM(25))

   model.add(Dense(50,activation= 'relu'))
   model.add(Dropout(0.5))
   model.add(Dense(1, activation='sigmoid'))
   model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001, beta_1=

   print(model.summary())

```

Model: "sequential_8"

LSTM+CNN

After trying multiple hyper tuning paraments with 'LSTM' we did not get much difference or higher accuracy than this in our experiments. So, we decided to try LSTM with CNN model which should generate better accuracy if the model is tuned well according to the literature. We experimented with multiple tuning and layers. Most of our experiments showed that this model is overfitting. Finally, we were able to achieve a model which slightly showed increase in accuracy, and we used that model as final model. This model gave us accuracy 87%. Which is slightly higher than LSTM, we believe if the model can be more tuned and add more layers then the accuracy might increase. Given the time and our skills permitted we were able to work till this model. There are many interesting ideas which we want to explore but keeping the scope of the project we used this model as final model.

The model we ran for 5 epochs, which started slightly overfitting after the 4th epoch. Below image shows the model. Below image shows the run-time of the model.

```

batch_size = 128, validation_split = 0.2, verbose = 1, callbacks=call_backs)

Epoch 1/5
250/250 [=====] - 85s 320ms/step - loss: 0.6929 - accuracy: 0.5094 - val_loss: 0.6920 - va
l_accuracy: 0.5017

Epoch 00001: val_accuracy improved from -inf to 0.50175, saving model to models\LSTM_CNN3.h5
Epoch 2/5
250/250 [=====] - 90s 362ms/step - loss: 0.4620 - accuracy: 0.7721 - val_loss: 0.3334 - va
l_accuracy: 0.8586

Epoch 00002: val_accuracy improved from 0.50175 to 0.85862, saving model to models\LSTM_CNN3.h5
Epoch 3/5
250/250 [=====] - 87s 349ms/step - loss: 0.2796 - accuracy: 0.8868 - val_loss: 0.3100 - va
l_accuracy: 0.8694

Epoch 00003: val_accuracy improved from 0.85862 to 0.86937, saving model to models\LSTM_CNN3.h5
Epoch 4/5
250/250 [=====] - 83s 332ms/step - loss: 0.2161 - accuracy: 0.9175 - val_loss: 0.2985 - va
l_accuracy: 0.8741

Epoch 00004: val_accuracy improved from 0.86937 to 0.87413, saving model to models\LSTM_CNN3.h5
Epoch 5/5
250/250 [=====] - 83s 333ms/step - loss: 0.1726 - accuracy: 0.9350 - val_loss: 0.3391 - va
l_accuracy: 0.8683

Epoch 00005: val_accuracy did not improve from 0.87413

```

BERT:

Below image shows the training of BERT model, with our cleaned and preprocessed dataset. We also extracte top 20 topics by creating a model with bert which generates 20 groups of topics. Bert uses Bert embeddings to convert words into bag of words.

```
[ ] model = BERTopic(language = 'English')
```

```
[ ] #It took nearly an 2 hours to execute this model
    topics, probs = model.fit_transform(review_list)
```

Downloading: 100%	<div></div>	1.18k/1.18k [00:00<00:00, 28.5kB/s]
Downloading: 100%	<div></div>	10.2k/10.2k [00:00<00:00, 268kB/s]
Downloading: 100%	<div></div>	612/612 [00:00<00:00, 14.0kB/s]
Downloading: 100%	<div></div>	116/116 [00:00<00:00, 2.10kB/s]
Downloading: 100%	<div></div>	39.3k/39.3k [00:00<00:00, 663kB/s]
Downloading: 100%	<div></div>	349/349 [00:00<00:00, 8.02kB/s]
Downloading: 100%	<div></div>	90.9M/90.9M [00:02<00:00, 36.5MB/s]
Downloading: 100%	<div></div>	53.0/53.0 [00:00<00:00, 1.04kB/s]
Downloading: 100%	<div></div>	112/112 [00:00<00:00, 2.47kB/s]
Downloading: 100%	<div></div>	466k/466k [00:00<00:00, 6.38MB/s]

LDA:

In LDA model features are generated using doc2bow vector, which converts words into bag of words.

Below shows the model training with the data after cleaning and preprocessing. We used pyLDAvis to visualize the topics generated. It generates bubbles with topics inside the bubble or cluster.

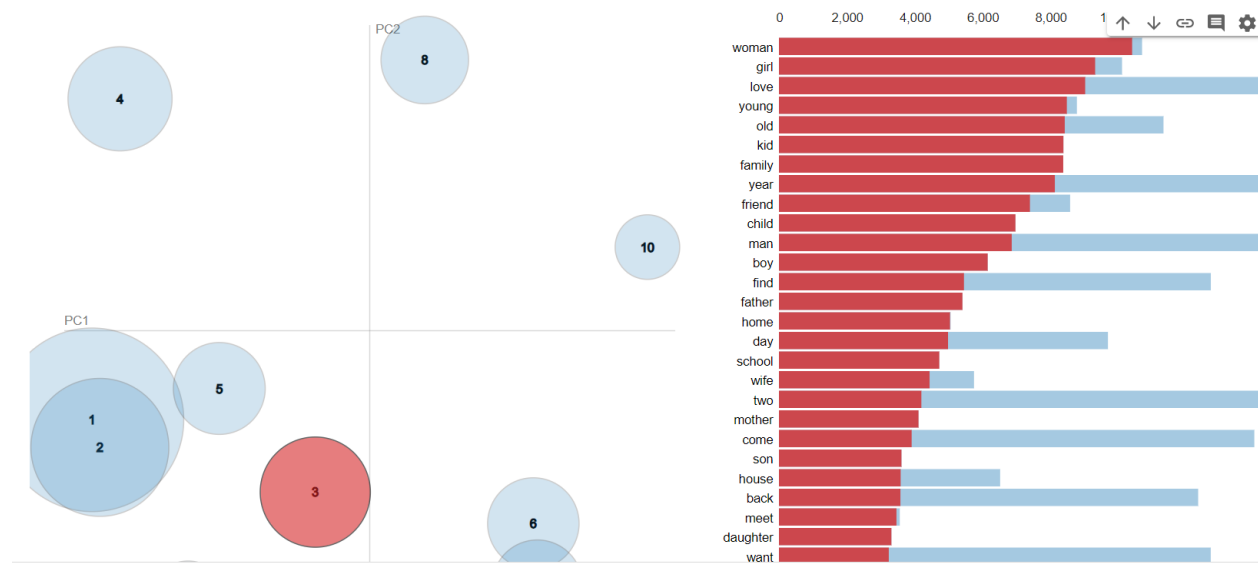
```
[ ] # Build LDA model
# This model took almost an hour to execute
# We tried executing multiple time by tuning the parameters to observe interesting results
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                             id2word=dictionary,
                                             num_topics=10,
                                             random_state=100,
                                             update_every=1,
                                             chunksize=500,
                                             passes=20,
                                             alpha='auto',
                                             per_word_topics=True)
```

Results:

Topic Modelling Results:

LDA

Topics visualized from LDA model, we generated top 10 topics. Below is the image of intertopic distance map generated using pyLDAvis.



This is prediction of probabilities of words generated in their respective clusters.

```
lda_model.save('model10.gensim')
topics = lda_model.print_topics(num_words=6)
for topic in topics:
    print(topic)

(0, '0.017*woman" + 0.015*girl" + 0.014*love" + 0.014*young" + 0.014*old" + 0.013*kid"')
(1, '0.011*work" + 0.010*director" + 0.008*seems" + 0.008*may" + 0.008*however" + 0.007*quite"')
(2, '0.067*show" + 0.025*series" + 0.018*episode" + 0.014*tv" + 0.011*u" + 0.009*world"')
(3, '0.009*world" + 0.009*art" + 0.009*american" + 0.009*cinema" + 0.008*black" + 0.008*white"')
(4, '0.009*look" + 0.008*ever" + 0.008*better" + 0.007*lot" + 0.007*something" + 0.007*watching"')
(5, '0.045*horror" + 0.015*blood" + 0.012*violence" + 0.012*flick" + 0.012*gore" + 0.011*sex"')
(6, '0.015*war" + 0.009*man" + 0.008*hero" + 0.007*human" + 0.006*evil" + 0.005*earth"')
(7, '0.013*role" + 0.013*john" + 0.011*play" + 0.008*played" + 0.008*man" + 0.008*robert"')
(8, '0.011*head" + 0.011*guy" + 0.010*game" + 0.010*run" + 0.009*car" + 0.008*kill"')
(9, '0.026*best" + 0.018*performance" + 0.015*music" + 0.014*love" + 0.013*book" + 0.010*version"')
```

BERT:

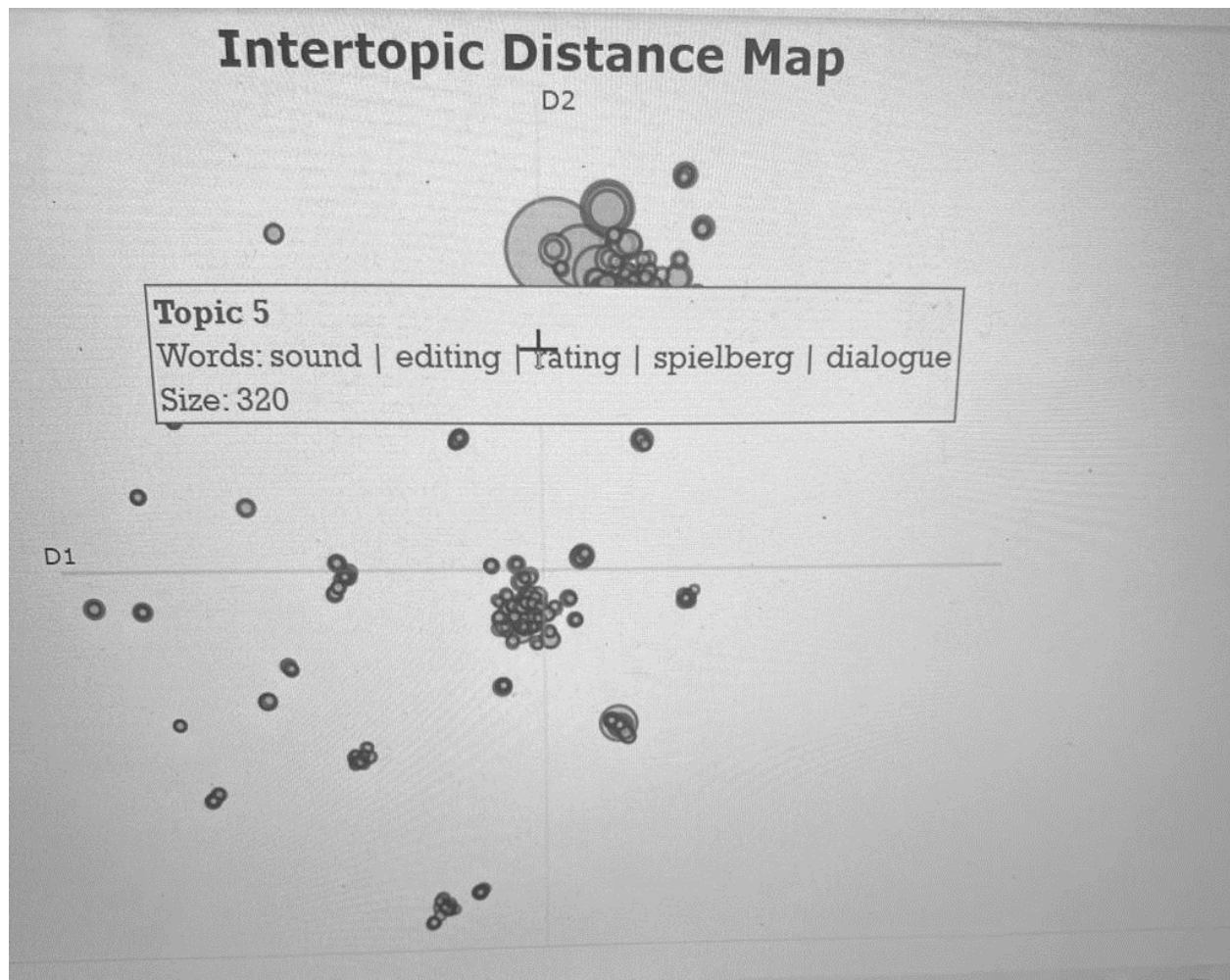
This is the count of topics generated by BERT, '-1' indicated topics that are not assigned to any group.

```
[ ] model.get_topic_freq()
```

	Topic	Count
0	-1	22879
1	0	1569
2	1	1070
3	2	668
4	3	457
...
727	731	10
726	732	10
725	733	10
723	729	10
736	735	10

737 rows × 2 columns

Below image shows the result of Topics generated by bert and we can observe that topic 5 is having words related to sound.



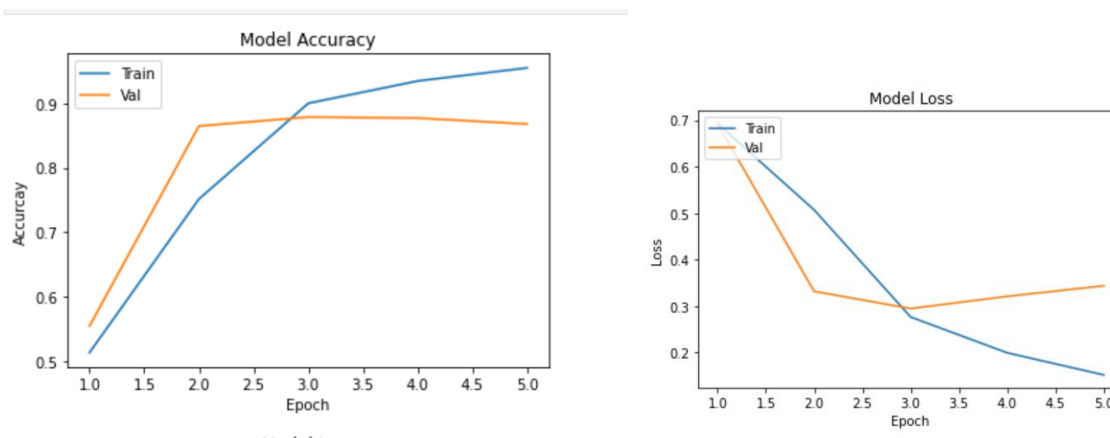
Sentiment Analysis:

We used evaluation metrics like accuracy, plotting learning plots between validation and test accuracy and losses, plotting auc, generating confusion matrix and representing predictions over a bar graph for true vs predicted values.

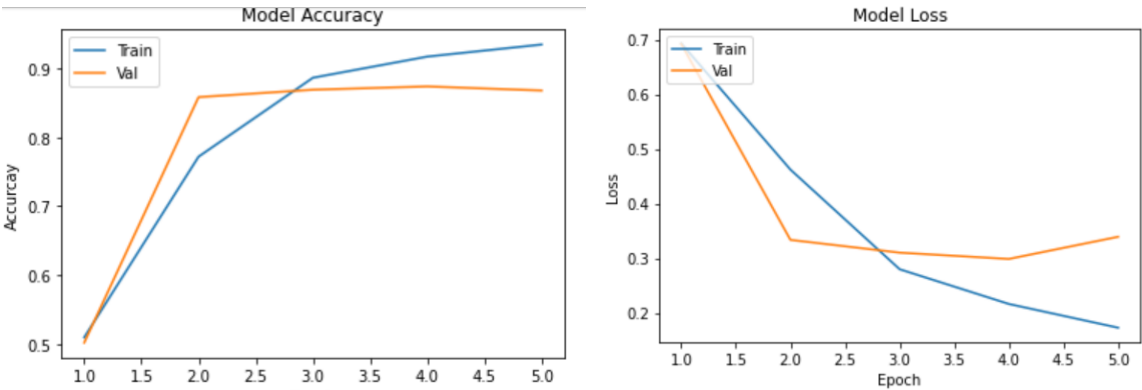
Table for accuracies:

	Naïve Bayes	SVM	LSTM	CNN+LSTM
Accuracy	81%(BoW)	84%(BoW)	86% (word vectors)	88% (word vectors)
	84%(tfidf)	88%(tfidf)		

Above table shows the comparison of accuracies for different models and different features. When we initially started with SVM model, we assumed that LSTM should give better results than SVM. After our research and experiments and from the literature which we referred in background module, we analyzed that SVM with bagwords gives less accuracy in cases compared to SVM with tfidfs. So, we started running experiments this time with base as Naivey Bayes model which have us 81% accuracy with BoW features. We took this as base model and developed SVM with BoW features which increase accuracy to 84%. We then tuned our LSTM model with LSTM layers and fine tuning parameters and achieved accuracy of 86%. In tuning the LSTM model, most of the time the model is overfitting. Now we want to increase our models accuracy to compare with tfidf accuracy of SVM. When we are researching for LSTM hyper tune for accuracy, we want to implement LSTM+CNN to increase accuracy. Finally, after many experimental hyper tunings we were able to achieve accuracy of 88% which is greater than all the other models accuracy and nearly equal to SVM tfidf model accuracy. Below are the images for validation and test accuracy and loss in LSTM model.



Below are the images for the images for accuracy and loss in LSTM+CNN model:



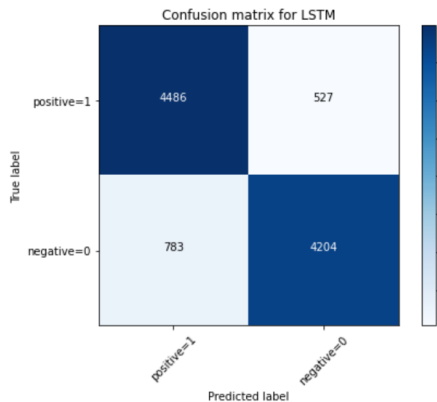
We also generated classification reports for both the models:

88% accuracy is for the CNN+LSTM model and 87% is for LSTM model.

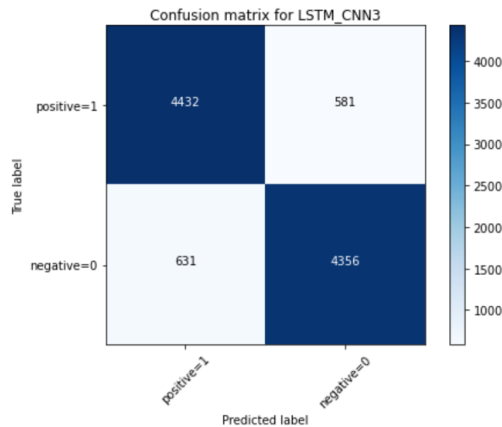
[: print(classification_report(y_test,y_pred_4))					:] print(classification_report(y_test,y_pred))				
	precision	recall	f1-score	support		precision	recall	f1-score	support
0	0.88	0.88	0.88	5013	0	0.85	0.89	0.87	5013
1	0.88	0.87	0.88	4987	1	0.89	0.84	0.87	4987
accuracy			0.88	10000	accuracy			0.87	10000
macro avg	0.88	0.88	0.88	10000	macro avg	0.87	0.87	0.87	10000
weighted avg	0.88	0.88	0.88	10000	weighted avg	0.87	0.87	0.87	10000

The confusion matrix for both the models are as shown below:

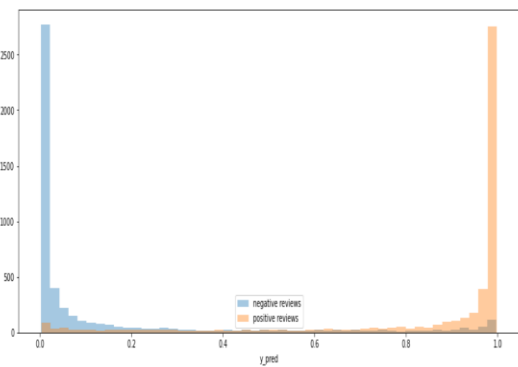
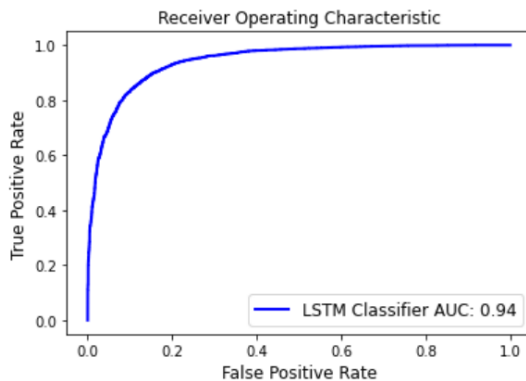
Confusion matrix
[[4486 527]
[783 4204]]



Confusion matrix
[[4432 581]
[631 4356]]



We also plotted ROC curves for both the models it AUC 94%, which is good percentage. We generated a plot for looking at the true results overlapped with predicted results. This shows the false prediction from the model are almost near to zero.



We finally gave real-time reviews to our model to predict, which results are shown below, it estimated a 6/10 rating review as positive, 1/10 rating as negative, and a review 8/10 as positive class.

For this we developed a function to preprocess the input and make the dimensions to fit in to our model, that is converting text into vectors and dimensions to predetermined ones. Below image show the real-time review analysis. These reviews are from [this link](#).

```

3]: #This review is originally rated 6 on the website

Custom_review1 = """ Why did you come here? What do you expect? The fast Saga - a movie that makes not much sense, but
Well no rules that the real world would set - boundaries and all that. I mean even Paul Walkers character is still a
80s movie reference aside this is what fans should and probably will expect. Including the long awaited and complete
Again it doesn't make much sense (or any to be fair), but it is fun to watch. And why should you not have fun ... th

4]: predict_CNN_review(Custom_review1)

4]: 'Positive(1) review'

5]: # 8 rated review
Custom_review3=""" I think the people in charge of IMDB should add Sci-Fi to the genres and I understand th

6]: predict_CNN_review(Custom_review3)

6]: 'Positive(1) review'

7]: #Below is the 1 rated review on the website

Custom_review2 = """ I am a huge fan of this franchise. And i watched all 9 movies again before F9. Here are my
(1) original 8/10... (2) 2 fast 7/10... (3) tokyo drift 3/10... (4) fast & furious 7/10... (5) fast 5 - 8/10...
story is useless. They tried to have character moments, but they didn't work for me. Which is odd because i'm al
My other biggest gripe: rewriting history. Apparently the trailer spoiled this, which is why i don't watch trail
did i mention it's really dumb? Unfathomably dumb. You know the overused phrase "i could feel myself getting dum
What a waste of charlize theron. I was so happy she survived f8. She's a BEAST of an actress (FURIOSA!), and i l
Racking my brain trying to come up with a positive. All i got is the musical score. Or at least one track was aw
My entire theater clapped when it finished. Ppl are stupid. But that means we'll probably get another one. Hopef
There is so much more i can talk about. I can't remember it all. But i'm sad and disappointed. My wait for good
8]: predict_CNN_review(Custom_review2)

8]: 'Negative(0) review'

```

Project Management:

Work Plan:

SPRINTS	Module	Due
---------	--------	-----

SPRINT0:	Work Plan.	Nov 2
SPRINT1:	Data Analysis, Preprocessing	Nov 6-7
SPRINT2:	Build Algorithms for Sentiment Analysis, Train and Tune	Nov 13-29
SPRINT3:	Build Algorithms for Topic Modelling, Train and Tune	Nov 13-29
SPRINT4:	Test Models performance, test on custom reviews.	Nov 27-30
SPRINT5:	Fine Tuning models	Nov 25-29
SPRINT6:	Final Delivery	Dec 1

Data Preprocessing:

In this stage of process, we initially cleaned the data and continued with all SVM and Naive bayes model, BERT models, then we observed that data needs to be more preprocessed. Then we started developing other functions to make data stemmed and cleaned. In this stage we generated visualizations of data and cleaning the data. We consolidated all the ideas and generated a single function initially for cleaning, then updated with more functions.

Responsibility (Task, Person)

Responsibility (Data Preprocessing, Gayatri, Harsha)

Generating Features:

Initial idea is to generate single type of features for all the models, to experiment more we started generating three types of features to give more scope for analyzing.

Responsibility (BoW vectors, Harsha)

Responsibility (Tfidf vectors, Gayatri)

Responsibility (word vectors, Gayatri)

Developing Naïve Baye, SVM:

In this module we developed Naïve Baye and SVM models and evaluated them, we even try to predict with real-time data.

Responsibility (Naïve Baye, Gayatri)

Responsibility (SVM, Harsha)

Developing LSTM, LSTM+CNN:

This module is we fine-tuned our models, we tried different models with different epochs from 3 to 20 and analyzed the results. We analyzed that more than 5 epochs models are overfitting. We saved trained models and used it for further evaluations.

We divided the tasks among us to try training the model with different hyper tune parameters. We came up with best fitting models.

This part of the project took us more time in analyzing the results, running the models and to select the best fitting hyper parameters. Generating visualization and evaluation metrics is also done in this module.

Responsibility (LSTM, LSTM+CNN models train and tune, Gayatri, Harsha)

Developing BERT, LDA:

This module is distributed between us. In this we wanted to try umap, pca, tsne embeddings which reduces the dimensionality, which we tried by spending lot of time for LDA. Because of timing and for the dataset we are using we shifted our scope more on Sentiment Analysis rather than Topic modelling. We generated topics using both models and are analyzed.

Responsibility (BERT, LDA models train and tune, Harishitha, Rakhsith)

Integrating, report, presentation:

After creating different models and different cleaning techniques on a whole all the project is integrated and generated results.

Responsibility (Report, Harsha, Gayatri, Rakshith, Harshitha)

Responsibility (Integration, Gayatri)

Responsibility (Presentation, Gayatri)

- Contributions (members/percentage):
- Gayatri: 35%
- Harsha: 35%
- Rakshith: 15%
- Harshitha: 15%

Issues/Concerns:

- Issues we encountered is cleaning the data, initially we did not realize that some of the cleaning techniques are not iterating through the entire data frame. We realized it halfway through and fixed code and developed again.
- As everyone is new python coding, we spend lot of time to understand the codes that needs to be applied and keep on updating and analyzing the codes.
- Due to lack of our knowledge on how to properly hyper tune parameters we spent lot of time running the models.
- One issue we encounter is running the models, mainly during integration of the project. Sometimes we ran out of memory. Many models took more than 2 hours we tried. But fortunately, the final model we can run quickly.
- One more issue is developing our idea of topic modelling, which initially we thought we will collect a dataset from web scrapping and perform topic modelling. But when we changed the plan to use the readily available dataset, as we started understanding that the scope of this project with the dataset is more on sentiment analysis part rather than topic modelling. But still we tried our best to explore both the concepts and run through few experiments in topic modelling.

Code Files:

Sentiment Analysis: SentimentAnalysis_final.ipynb

Topic Modelling: TopicModellingBertLDA.ipynb & TopicModellingBert.ipynb

TopicModellingBert.ipynb has 20 topics generated from the topics

The visualizations we developed for topic modelling is accessed through colab only.

References:

Sentiment Analysis paper:

[1] <https://iopscience.iop.org/article/10.1088/1757-899X/1077/1/012001/pdf>

[2] <https://link.springer.com/article/10.1007/s40747-021-00436-4>

Topic Modelling paper:

[3] https://web.stanford.edu/class/cs224n/reports/final_reports/report017.pdf

BERT Architecture:

[4] <https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/>

Dataset:

[5] <https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

Unidecode:

[6] <https://pypi.org/project/Unidecode/>

Image source:

[7] https://www.google.com/search?q=naive+bayes+posterior+probability&rlz=1C1ONGR_enUS939US939&sxsrf=AOaemvIj8qfa2xe5jJRzm-ITnV7TFy-qjg:1638463652788&source=lnms&tbm=isch&sa=X&ved=2ahUKEwj4ooHgyMX0AhUJIWoFHVyFDlkQ_AUoAnoECAEQBA&biw=1280&bih=609&dpr=1.5#imgrc=kwLT20eBUyxVdM

LSTM:

[8] <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Link for reviews:

[9] https://www.imdb.com/title/tt5433138/reviews?ref_=tt_sa_3

DataSet:

[10] <https://ai.stanford.edu/~amaas/data/sentiment/>

[11] <https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

Resource help/code help:

[12] <https://google.com>

[13] <https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0>

- [14] <https://medium.com/@mrunal68/text-sentiments-classification-with-cnn-and-lstm-f92652bc29fd>
- [15] <https://ieeexplore.ieee.org/document/9117512>
- [16] <https://iq.opengenus.org/naive-bayes-on-tf-idf-vectorized-matrix/>
- [17] <https://towardsdatascience.com/topic-modeling-with-bert-779f7db187e6>
- [18] <https://www.analyticsvidhya.com/blog/2021/06/part-2-topic-modeling-and-latent-dirichlet-allocation-lda-using-gensim-and-sklearn/>
- [19] <https://towardsdatascience.com/unsupervised-sentiment-analysis-a38bf1906483>
- [20] <https://github.com/AnushaMeka/NLP-Topic-Modeling-LDA-NMF/blob/master/Topic%20Modeling.ipynb>
- [21] <https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0>
- [22] <https://towardsdatascience.com/naive-bayes-and-lstm-based-classifier-models-63d521a48c20>

