**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 pf 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.

A picture containing logo

Description automatically generated

Fig.1 Flowchart of BERT

**3.Sentiment Analysis:**

Sentiment Analysis is widely used 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.

Topimodels

Language Modelling (BERT

, LDA)

Clean data

Reviews from the dataset

Data Set

clustering

Models (SVM, LST M)

Cleaned Reviews

Word Vectors

Topic Models

Predicted as ‘0’ or ‘1’

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 revie 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

**A picture containing table

Description automatically generated**

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

After cleaning the reviews.

Text

Description automatically generated

Observing positive and negative reviews.

Graphical user interface, text, application

Description automatically generated

Bar graph to analyze dataset features visually

Chart, bar chart

Description automatically generated

1. **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

Table

Description automatically generated

Confusion matrix generated:

Chart, treemap chart

Description automatically generated

1. 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:

Table

Description automatically generated

Chart, scatter chart

Description automatically generated

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 tunning 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.

Input Layer

Output Shape (None,130,100)

Embedding Layer

Output Shape (None,130,32)

Conv1D

Output Shape (None,65, 32)

Max\_Pool1D

Output Shape (None, 65,32)

Dropout

Output Shape (None, 100)

LSTM

Output Shape (None, 1)

Output Layer, sigmoid activation

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.

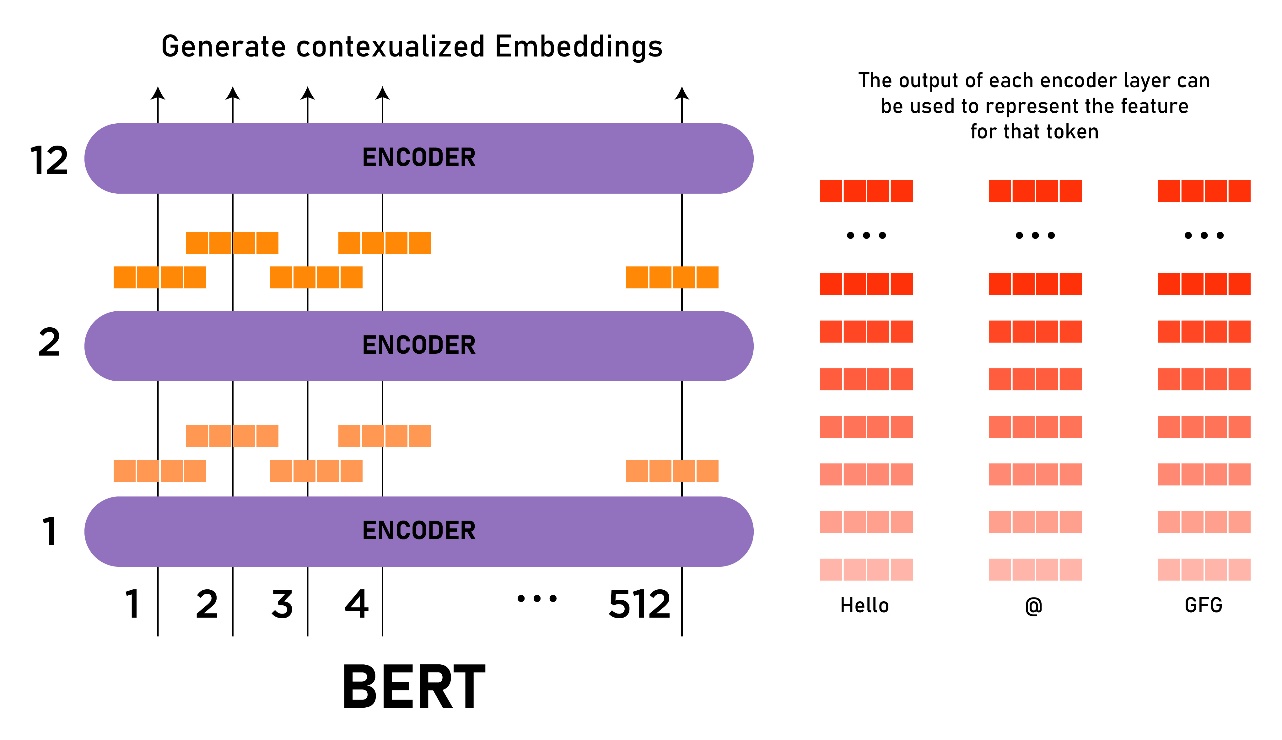
Below figure shows the model summary of our final model.

Table

Description automatically generated

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 a 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 embediings
* 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.

Predicted as ‘0’ or ‘1’

Topic models

Models (SVM, LST M)

Clean data

Reviews from the dataset

Data Set

clustering

Cleaned Reviews

Word Vectors

Topic Models

Language Modelling (BERT

, LDA)

Topic 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.

Graphical user interface, text, application

Description automatically generated

**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’.

Chart, bar chart

Description automatically generatedGraphical user interface, text, application

Description automatically generated

**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.

Text

Description automatically generated

Below image shows before and after output of data passing through ‘clean\_data’ function.

A picture containing text

Description automatically generatedTable

Description automatically generated

*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.

Graphical user interface, text, application

Description automatically generated

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

Before stemming:

Text

Description automatically generated with medium confidence

After stemming:

Text

Description automatically generated

*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.

Text

Description automatically generated

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.

Remove\_stopwords()

It removes ‘to’ , ‘the’ , ‘in’ , ‘a’, etc.

Simple\_stemmer()

This converts ‘writing’ to ‘write’, ‘reading’ to ‘read’, etc., to their ‘stem words’.

Clean\_data()

This removes ‘\*\*\*’, ‘ ’, ‘+’, makes text to lower cases, ‘removes digits’, etc.

output

input

output

input

input

output

Output:

awesome Worth watch

It is awesome Worth watch

It is awesome Worth watching

Input:

It is awesome!!!!!!!!!! Worth watching!!

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.

Chart

Description automatically generated

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

A picture containing graphical user interface

Description automatically generated

**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.



**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(word | topics) and P(topics | 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 coutvectorizer and these will be converted into respective vectors.

Graphical user interface, text, application, email

Description automatically generated

And this is how they are assigned with frequencies.

Table

Description automatically generated with medium confidence

Topics and their frequency of words:

A picture containing calendar

Description automatically generated

1. *Tfidf word vetcors:*

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:

Graphical user interface, text, application

Description automatically generated

Tfidf vectors:

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

Table

Description automatically generated with medium confidence

1. *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.

A picture containing table

Description automatically generated

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.

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

***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.

Graphical user interface, text, application, email

Description automatically generated

We also did apply it on tfidf vectors.

Graphical user interface, text, application, email

Description automatically generated

***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().

Text

Description automatically generated with medium confidence

***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.

Graphical user interface, text, application, email

Description automatically generated

***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.

Text

Description automatically generated with medium confidence

***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.

***Graphical user interface

Description automatically generated***

***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.

***Graphical user interface, text, application

Description automatically generated***

**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.

Chart, bubble chart

Description automatically generated

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

**A picture containing text

Description automatically generated**

***BERT:***

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

Table

Description automatically generated with low confidence

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

Chart, scatter chart

Description automatically generated

***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.

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

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

**Chart, line chart

Description automatically generated** **Chart, line chart

Description automatically generated**

We also generated classification reports for both the models:

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

Table

Description automatically generated Table

Description automatically generated

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

**Chart

Description automatically generatedChart, treemap chart

Description automatically generated**

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.

Graphical user interface

Description automatically generated with medium confidence**A picture containing text

Description automatically generated**

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.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**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=2ahUKEwj4ooHgyMX0AhUJlWoFHVyFDlkQ_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>