**Topic modelling and Sentiment Analysis using Deep Learning (RNN-LSTM) on Movie Reviews Dataset**

Thesis submitted in partial fulfillment of the

requirements for

**Post Graduate Diploma in Data Science**

By

**Anurag Vishwakarma**

18325760003

Under the guidance of

Mohan Silaparashetty

Head of Department

Manipal Prolearn

Bengaluru



**MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL**

**Topic modelling and Sentiment Analysis using Deep Learning (RNN-LSTM) on Movie Reviews Dataset**

Thesis submitted in partial fulfillment of the

requirements for

**Post Graduate Diploma in Data Science**

By

**Anurag Viswakarma**

18325760003

**Examiner 1** **Examiner 2**

Signature: Signature:

Name: Name:



**MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL**

**CERTIFICATE**

This is to certify that the project work titled

**Topic modelling and Sentiment Analysis using Deep Learning (RNN-LSTM) on Movie Reviews Dataset**

is a bonafide record of the work done by

**Anurag Vishwakarma**

18325760003

In partial fulfilment of the requirements for the award of **Post Graduate Diploma in** **Data Science** under Manipal Academy of Higher Education, Manipal, Manipal and thesame has not been submitted elsewhere for any kind of certification/recognition.

Mohan Silaparashetty

Head of Department

Manipal Prolearn

Bengaluru

**ACKNOWLEDGEMENTS**

We express our deep gratitude and regards to **Mohan Silaparashetty**, Internal Guide and Professor, Head of Department for his encouragement and valuable guidance in bringing shape to this dissertation.

I am thankful to all the **Professors and Faculty Members** in the department for their teachings and academic support and thanks to **Technical Staff and Non-teaching staff** in the department for their support.

**Anurag Vishwakarma**

18325760003

**Table of Contents**

1. **Acknowledgments**
2. **Table of contents with page number**
3. **Abstract**
   1. **Introduction**

|  |  |
| --- | --- |
| **1.1.** | **Problem Statement** |
| **1.2.** | **Project Goal and Scope** |
| **2. Approach** | |
| **2.1.** | **Data Pre-processing** |
| **2.2.** | **Exploratory Data Analysis** |
| **2.3.** | **Feature Engineering** |
| **2.4.** | **Topic Modelling** |
| **2.5.** | **Model Training and Evaluation** |

1. **Data Pre-processing**
2. **Exploratory Data Analysis 4.1. Data collection**

**4.2. Data exploration**

**4.3. Complexity of Data**

**4.4. Data cleaning**

**4.5. Data Transformation**

**5. Topic Modelling**

**6. Modelling**

**6.1. Selection of model/technique**

**6.2. Model Performance Report**

**7. Analysis of the results**

**8. Conclusion**

**9. References**

**10. Appendix**

**ABSTRACT**

Sentiment analysis or opinion mining is the computational study of people’s opinions, sentiments, attitudes, and emotions expressed in written language. It is one of the most active research areas in natural language processing and text mining in recent years. Its popularity is mainly due to two reasons. First, it has a wide range of applications because opinions are central to almost all human activities and are key influencers of our behaviours. Whenever we need to make a decision, we want to hear other’s opinions. Second, it presents many challenging research problems, which had never been attempted before the year 2000. Part of the reason for the lack of study before was that there was little opinionated text in digital forms. It is thus no surprise that the inception and the rapid growth of the field coincide with those of the social media on the Web. In fact, the research has also spread outside of computer science to management sciences and social sciences due to its importance to business and society as a whole. In this talk, I will start with the discussion of the mainstream sentiment analysis research and then move on to describe some recent work on modelling comments, discussions, and debates, which represents another kind of analysis of sentiments and opinions.

Sentiment classification is a way to analyse the subjective information in the text and then mine the opinion. Sentiment analysis is the procedure by which information is extracted from the opinions, appraisals and emotions of people in regards to entities, events and their attributes. In decision making, the opinions of others have a significant effect on customers ease, making choices with regards to online shopping, choosing events, products, entities. The approaches of text sentiment analysis typically work at a particular level like phrase, sentence or document level. This paper aims at analysing a solution for the sentiment classification at a fine-grained level, namely the sentence level in which polarity of the sentence can be given by three categories as positive, negative and neutral.

**1. INTRODUCTION**

The Sentiment analysis is part of natural language processing. Natural language Processing is used for data analytics purpose, to extract meaningful information from lots of data. This is one of the methods to get information about current trend in the market of what people are thinking or talking on social media. There are so many practical applications present in the current world like in election which party is favourable or gaining popularity or a customer watching for reviews before actually buying something online. These are few of the applications which are getting harder to solve as size of data keeps on increasing. Big part goes to arrange this data into something meaningful before analysing it. This part of arrangement of data is called Text Classification. Sentiment classification and analysis is performed in python using nltk module. Python has special module NLTK to do tasks in natural language processing. It supports multiple languages like English, Hindi, Chinese etc to do classification of text or data into something meaningful. Text Classification can be performed in following ways:

1. Sentiment-Classification
2. Features-based-Sentiment-classification
3. Summarization-of-sentiments

These classifications classify the complete document in accordance with the sentiments or opinions listed in the text. Feature based approach however, classifies the sentiments based on specifications of the entity (Noun) listed in the text. This approach reveals about good or bad quality about certain entities based on the details listed with it. Opinions summarization is similar to text summarization but opinion summarization gives a clear indication about the sentiment attached with the text. It outputs the sentiment precisely not in the form of substring of the given text, It mentions the text in the positive or negative words about the entities so that a whole document can be best described in few words without losing the abstract of the document. These types of classification can be performed before actually analysing the text. After text classification, it performs tagging with the words. Sentiment classification can be performed at different level.

1. Document Level

2. Sentence level

3. Word level

English is one of the most preferred language to work for natural language processing. This project is based on opinions in English language, does not support other languages at all. Consider an example: "I watched the movie burger. The movie was very good and the actor did an awesome job." "When Modi returned from U.S.A., I got my 15 lakhs as promised by PM Modi" It clearly tells about the movie and the actor stating positive review. However the sentiment classifier is still not able to classify sarcasm. It is still a big problem for data analytics and a topic of research. How to perform this in a machine language is much harder. There are approaches which perform such operations

1. Linguistic approach

2. Machine Learning

**1.1 Problem statement:**

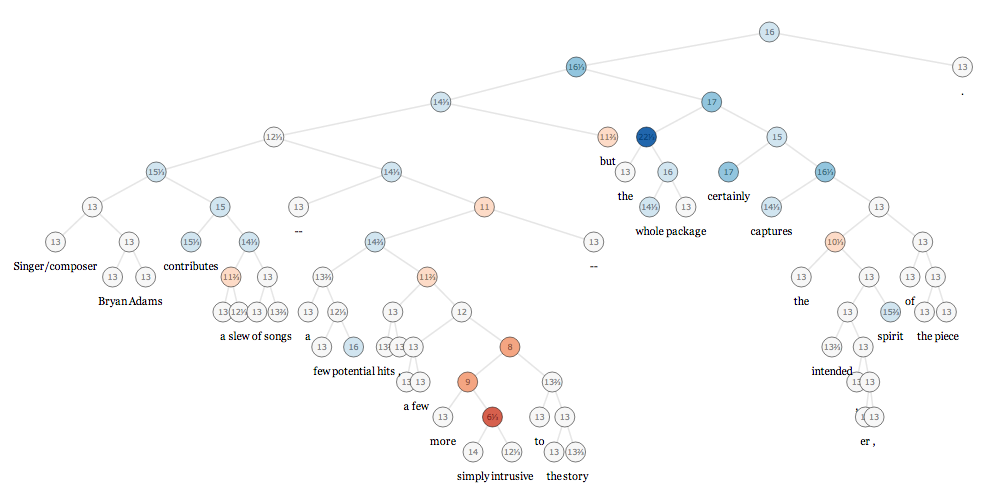
Users have written over hundreds of reviews for each movie. The reviews are expressed in the natural language, along with a self-annotated score describing the overall sentiment of that review. To make a better-informed decision, user has to go through each of them, which is a time-consuming activity that user is highly unlikely to invest time in.

In this project I am helping users making better informed decision, not only based on the aggregate of the self-annotated data but also by calculating the semantic orientation and polarity of each review individually.

**1.2 Project Goal and Scope:**

"There's a thin line between likably old-fashioned and fuddy-duddy, and The Count of Monte Cristo ... never quite settles on either side."

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee. In their work on sentiment treebanks, Socher et al. used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This competition presents a chance to benchmark your sentiment-analysis ideas on the Rotten Tomatoes dataset. You are asked to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. Obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very challenging.



**2. Approach**

Approach to be followed to fulfil the project requirement.

**2.1 Data Pre-processing**.

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing.  
  
Data pre-processing is used database-driven applications such as customer relationship management and rule-based applications (like neural networks).

Steps involved in Data Pre-processing.

* Data Cleaning
* Data Transformation
* Data Reduction

**2.2** **Exploratory Data Analysis**.

In statistics, exploratory data analysis (EDA) is an approach to analysing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

**2.3 Feature Engineering.**

**Feature engineering** is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. The need for manual feature engineering can be obviated by automated feature learning.

Feature engineering is an informal topic, but it is considered essential in applied machine learning.

Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.

The feature engineering process is:

* [Brainstorming](https://en.wikipedia.org/wiki/Brainstorming) or testing features.
* Deciding what features to create.
* Creating features.
* Checking how the features work with your model.
* Improving your features if needed.
* Go back to brainstorming/creating more features until the work is done.

**2.4 Topic modelling.**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), a **topic model** is a type of [statistical model](https://en.wikipedia.org/wiki/Statistical_model) for discovering the abstract "topics" that occur in a collection of documents. Topic modelling is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently: "dog" and "bone" will appear more often in documents about dogs, "cat" and "meow" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, in a document that is 10% about cats and 90% about dogs, there would probably be about 9 times more dog words than cat words. The "topics" produced by topic modelling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

In this project I am using **Latent Dirichlet Allocation (LDA)** for topic modelling.

In [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), **latent Dirichlet allocation** (**LDA**) is a [generative statistical model](https://en.wikipedia.org/wiki/Generative_model) that allows sets of observations to be explained by [unobserved](https://en.wikipedia.org/wiki/Latent_variable) groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a [topic model](https://en.wikipedia.org/wiki/Topic_model).

**2.5 Model training and Evaluation.**

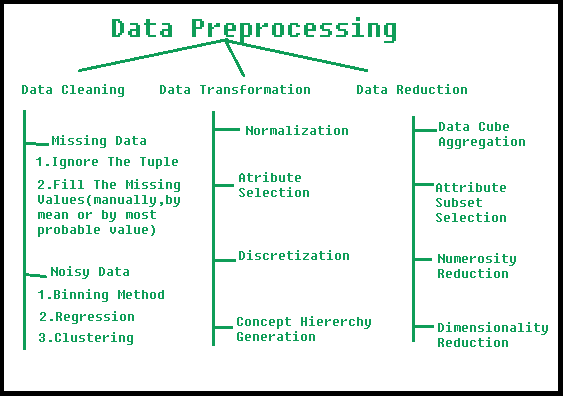
I have used multiple algorithms to know and get maximum accuracy.

Following are the algorithms used in this project.

1. Multinomial Naïve Bayes
2. Logistic Regression
3. Random Forest
4. Support Vector Machine
5. Recurrent Neural Network (RNN-LSTM)

**3. Data Pre-processing**

Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format.



**Steps Involved in Data Pre-processing:**

**1. Data Cleaning:**  
The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

* **(a). Missing Data:**  
  This situation arises when some data is missing in the data. It can be handled in various ways.  
  Some of them are:
  1. **Ignore the tuples:**  
     This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
  2. **Fill the Missing values:**  
     There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.
* **(b). Noisy Data:**  
  Noisy data is a meaningless data that can’t be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways:
  1. **Binning Method:**  
     This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
  2. **Regression:**  
     Here data can be made smooth by fitting it to a regression function. The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).
  3. **Clustering:**  
     This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

**2. Data Transformation:**  
This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization:**  
   It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)
2. **Attribute Selection:**  
   In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
3. **Discretization:**  
   This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
4. **Concept Hierarchy Generation:**  
   Here attributes are converted from level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

**3. Data Reduction:**  
Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. **Data Cube Aggregation:**  
   Aggregation operation is applied to data for the construction of the data cube.
2. **Attribute Subset Selection:**  
   The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute. The attribute having p-value greater than significance level can be discarded.
3. **Numerosity Reduction:**  
   This enable to store the model of data instead of whole data, for example: Regression Models.
4. **Dimensionality Reduction:**  
   This reduce the size of data by encoding mechanisms. It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Component Analysis).

**4. Exploratory Data Analysis**

In statistics, exploratory data analysis (EDA) is an approach to analysing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

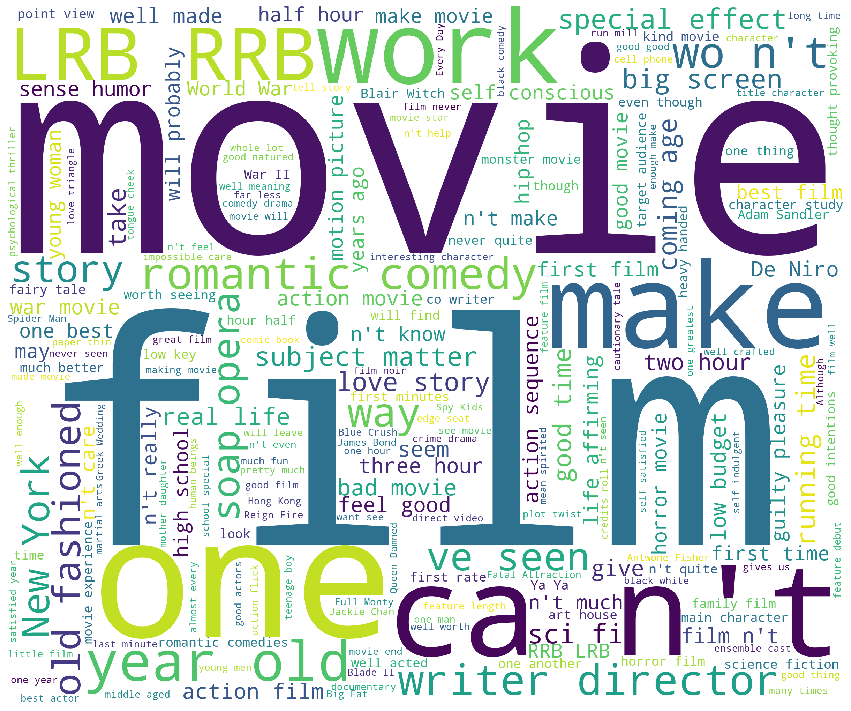
**4.1 Data Collection**

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data.

* train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence.
* test.tsv contains just phrases. You must assign a sentiment label to each phrase.

The sentiment labels are:

0 - negative  
1 - somewhat negative  
2 - neutral  
3 - somewhat positive  
4 - positive

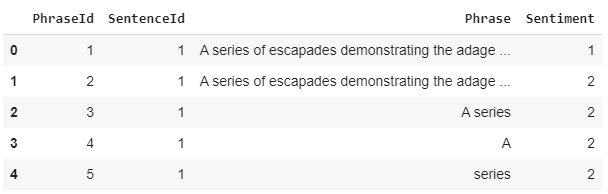


**4.2 Data Exploration**

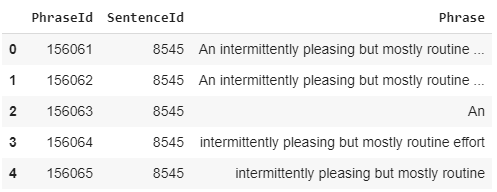
The data consist of two tables:

* 1. train.tsv
  2. test.tsv

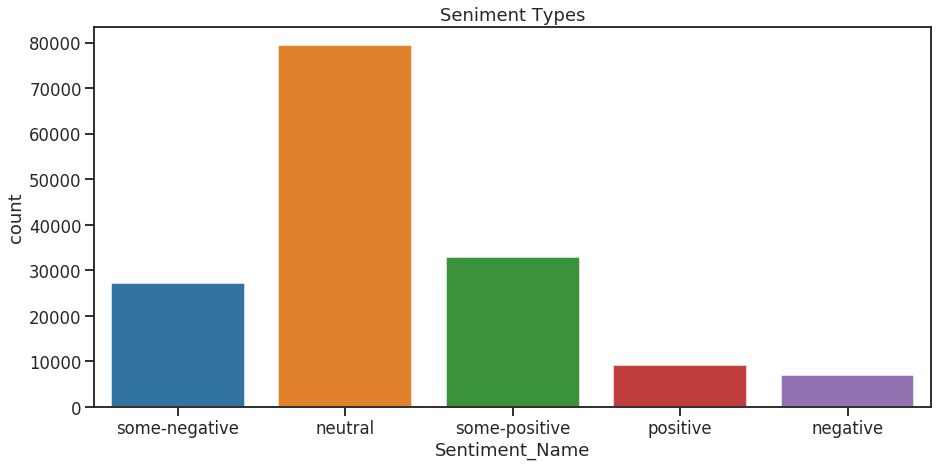
**test.tsv** contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence.



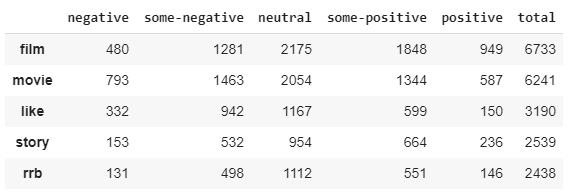
**test.tsv** contains just phrases. You must assign a sentiment label to each phrase.



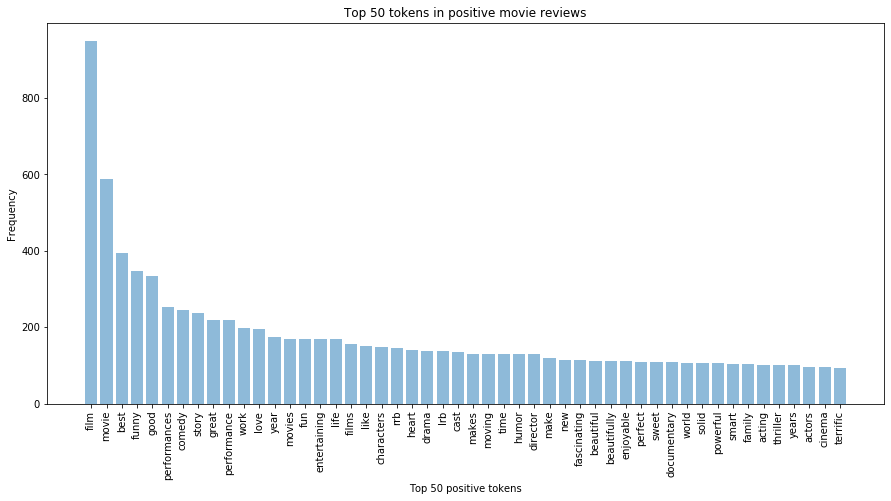
This Graphs show the distribution of data according to the sentiment class.



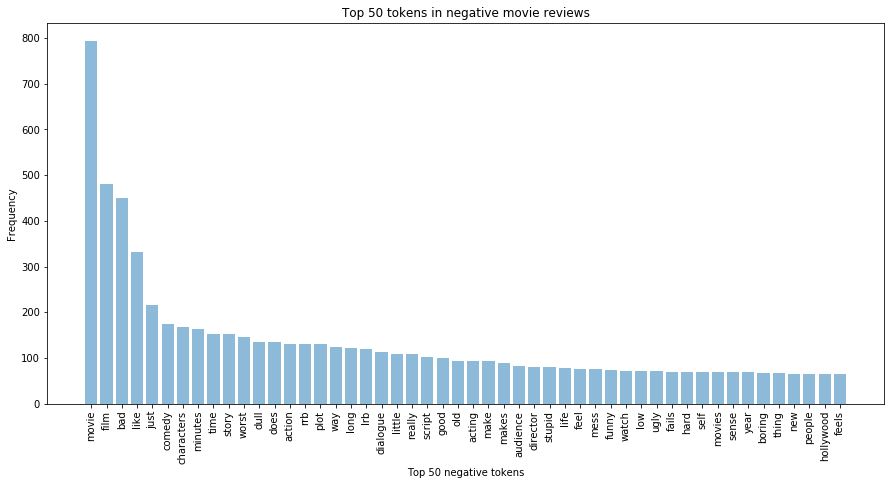
This tables shows the Term frequency of each word.



This graph shows top 50 tokens in Positive Sentiment.



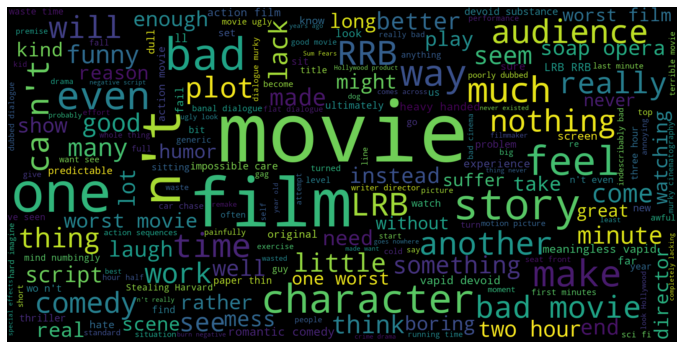
This graph shows top 50 tokens in Negative Sentiment



This WorldCloud show the Positive Sentiment.



This WorldCloud show the Negative Sentiment.



**4.3 Complexity of Data**

There are two tables present in the data.

* train.tsv contains a shape of (156060, 4)

- 156060 number of rows.

- 4 number of columns

* test.tsv contains a shape of (66292, 3)

- 66292 number of rows.

- 3 number of columns.

**4.4 Data Cleaning**

Data Cleaning is not required as the data is already cleaned and does not contain null values.

**4.5 Data Transformation**

As the data contains the only the text data, so we need to convert that text data into vector(numerical) form, in order to make it acceptable to the model or algorithm we are using.

In order to convert text data into vector form, I am using two types of vectorization technique.

1. Count Vectorization.
2. TFIDF Vectorization.

**Count Vectorization**

Consider a Corpus C of D documents {d1,d2…..dD} and N unique tokens extracted out of the corpus C. The N tokens will form our dictionary and the size of the Count Vector matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D(i).

Let us understand this using a simple example.

D1: He is a lazy boy. She is also lazy.

D2: Neeraj is a lazy person.

The dictionary created may be a list of unique tokens(words) in the corpus =[‘He’,’She’,’lazy’,’boy’,’Neeraj’,’person’]

Here, D=2, N=6

The count matrix M of size 2 X 6 will be represented as –

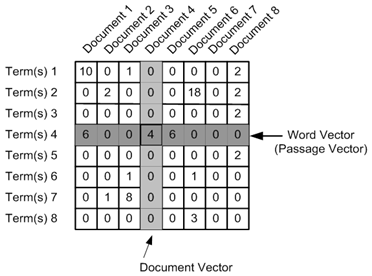
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | He | She | lazy | boy | Neeraj | person |
| D1 | 1 | 1 | 2 | 1 | 0 | 0 |
| D2 | 0 | 0 | 1 | 0 | 1 | 1 |

Now, a column can also be understood as word vector for the corresponding word in the matrix M. For example, the word vector for ‘lazy’ in the above matrix is [2,1] and so on.Here, the *rows* correspond to the *documents* in the corpus and the *columns* correspond to the *tokens* in the dictionary. The second row in the above matrix may be read as – D2 contains ‘lazy’: once, ‘Neeraj’: once and ‘person’ once.

Now there may be quite a few variations while preparing the above matrix M. The variations will be generally in-

1. The way dictionary is prepared.  
   Why? Because in real world applications we might have a corpus which contains millions of documents. And with millions of document, we can extract hundreds of millions of unique words. So basically, the matrix that will be prepared like above will be a very sparse one and inefficient for any computation. So an alternative to using every unique word as a dictionary element would be to pick say top 10,000 words based on frequency and then prepare a dictionary.
2. The way count is taken for each word.  
   We may either take the frequency (number of times a word has appeared in the document) or the presence(has the word appeared in the document?) to be the entry in the count matrix M. But generally, frequency method is preferred over the latter.

Below is a representational image of the matrix M for easy understanding.



**TFIDF Vectorization**

This is another method which is based on the frequency method but it is different to the count vectorization in the sense that it takes into account not just the occurrence of a word in a single document but in the entire corpus. So, what is the rationale behind this? Let us try to understand.

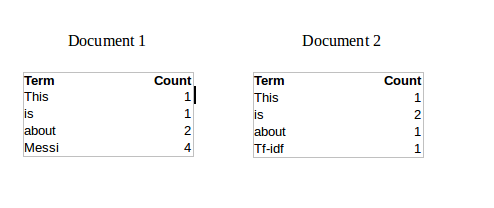
Common words like ‘is’, ‘the’, ‘a’ etc. tend to appear quite frequently in comparison to the words which are important to a document. For example, a document **A** on Lionel Messi is going to contain more occurrences of the word “Messi” in comparison to other documents. But common words like “the” etc. are also going to be present in higher frequency in almost every document.

Ideally, what we would want is to down weight the common words occurring in almost all documents and give more importance to words that appear in a subset of documents.

TF-IDF works by penalising these common words by assigning them lower weights while giving importance to words like Messi in a particular document.

So, how exactly does TF-IDF work?

Consider the below sample table which gives the count of terms(tokens/words) in two documents.



Now, let us define a few terms related to TF-IDF.

TF = (Number of times term t appears in a document)/(Number of terms in the document)

So, TF(This,Document1) = 1/8

TF(This, Document2)=1/5

It denotes the contribution of the word to the document i.e words relevant to the document should be frequent. eg: A document about Messi should contain the word ‘Messi’ in large number.

IDF = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.

where N is the number of documents and n is the number of documents a term t has appeared in.

So, IDF(This) = log(2/2) = 0.

So, how do we explain the reasoning behind IDF? Ideally, if a word has appeared in all the document, then probably that word is not relevant to a particular document. But if it has appeared in a subset of documents then probably the word is of some relevance to the documents it is present in.

Let us compute IDF for the word ‘Messi’.

IDF(Messi) = log(2/1) = 0.301.

Now, let us compare the TF-IDF for a common word ‘This’ and a word ‘Messi’ which seems to be of relevance to Document 1.

TF-IDF(This,Document1) = (1/8) \* (0) = 0

TF-IDF(This, Document2) = (1/5) \* (0) = 0

TF-IDF(Messi, Document1) = (4/8)\*0.301 = 0.15

As, you can see for Document1 , TF-IDF method heavily penalises the word ‘This’ but assigns greater weight to ‘Messi’. So, this may be understood as ‘Messi’ is an important word for Document1 from the context of the entire corpus.

**5. Topic Modelling**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), a **topic model** is a type of [statistical model](https://en.wikipedia.org/wiki/Statistical_model) for discovering the abstract "topics" that occur in a collection of documents. Topic modelling is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently: "dog" and "bone" will appear more often in documents about dogs, "cat" and "meow" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, in a document that is 10% about cats and 90% about dogs, there would probably be about 9 times more dog words than cat words. The "topics" produced by topic modelling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

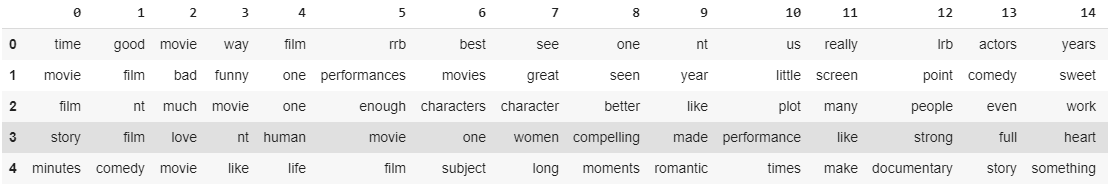
In this project I am using **Latent Dirichlet Allocation (LDA)** for topic modelling.

In [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), **latent Dirichlet allocation** (**LDA**) is a [generative statistical model](https://en.wikipedia.org/wiki/Generative_model) that allows sets of observations to be explained by [unobserved](https://en.wikipedia.org/wiki/Latent_variable) groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a [topic model](https://en.wikipedia.org/wiki/Topic_model).

Steps involved in Topic Modelling using LDA.

* Preparing Documents
* Cleaning and Pre-processing
* Preparing Document-Term Matrix
* Running LDA Model
* Result

The Final result of Topic Modelling using Latent Dirichlet Allocation (LDA) is shown below.



**6. Modelling**

The trainset was split in the ratio 80:20 train and validation set respectively. In every execution the textual data was transformed in TF – IDF matrix.

**Machine Learning models and TF – IDF as feature extraction:**

TF - IDF is an information retrieval technique that weighs a term’s frequency (TF) and its inverse document frequency (IDF).

The trainset and the test set were converted via the TF – IDF vectorizer from sklearn. we can see that due to the fact the ngram\_range is from 1 to 3 the columns of the TF - IDF matrix vectorizer is extremely huge. After applying TF-IDF vectorizer the shape of train set is increased to 301627. This may lead us to slow down the Machine Learning models to fit the data.

For all the ML models the random state will be set to 42 in order to the models be reproducible and create the same results in every run.

**6.1 Algorithms and Techniques:**

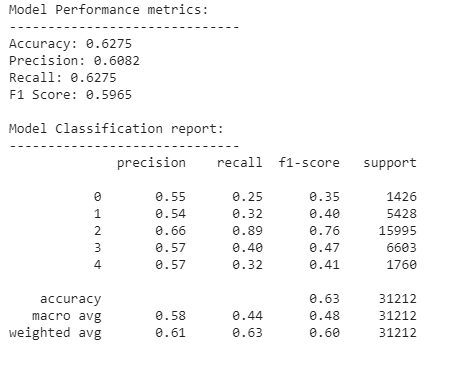
The machine learning techniques that are used are:

* Logistic Regression
* Random Forest Classifier
* Linear SVC
* RNN-LSTM

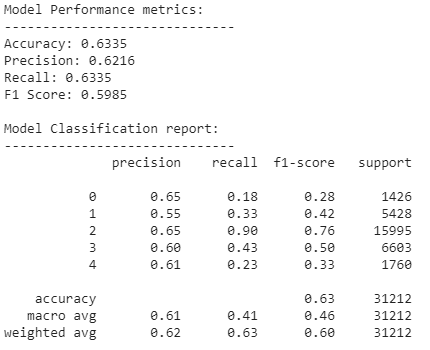
**6.2 Model Performance Report**

**Logistic Regression**

Model Performance metrics of Logistic regression model on Count Vectorizer.

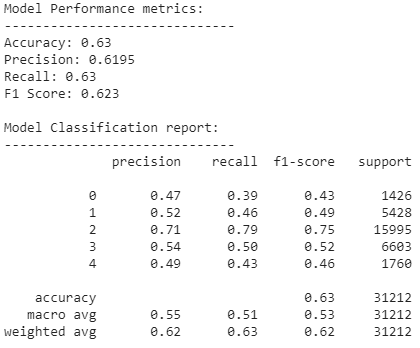


Model Performance metrics of Logistic regression model on TF-IDF Vectorizer.

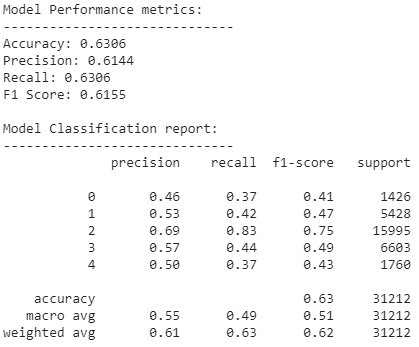
****

**Random Forest Classifier**

Model Performance metrics of Random Forest Classifier model on Count Vectorizer.

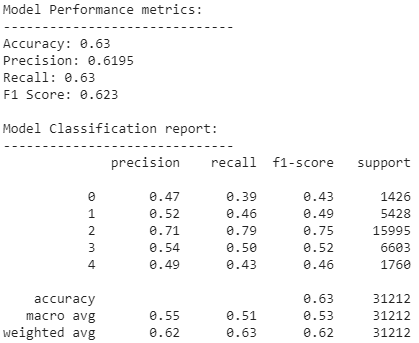


Model Performance metrics of Random Forest Classifier model on TF-IDF Vectorizer.

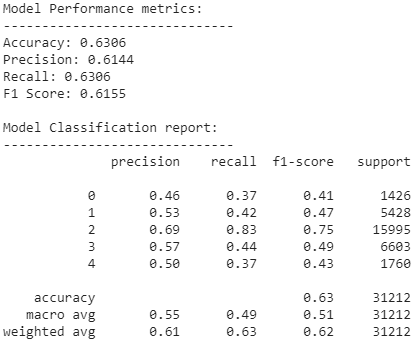
****

**Support Vector Machine Classifier**

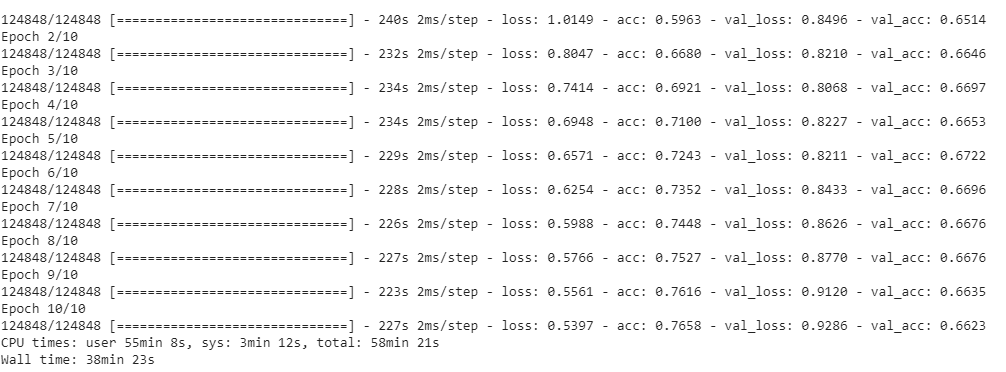
Model Performance metrics of Support Vector Classifier model on Count Vectorizer.



Model Performance metrics of Support Vector Classifier model on TF-IDF Vectorizer.

****

**RNN-LSTM Model**

****

**7. Analysis of the model Performance**

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (CountVector)** | **Accuracy(TF-IDF)** |
| Logistic Regression | 0.6275 | 0.6335 |
| Random Forest Classifier | 0.63 | 0.6306 |
| Support Vector Machine Classifier | 0.63 | 0.6306 |
| RNN-LSTM | 0.7658 | |

As we can see, the maximum accuracy is given by the Deep Learning (RNN-LSTM) model with an accuracy of 76.58%. So we will be using deep learning model to deploy and use for our Sentiment Analysis requirement.

**8. Conclusion**

After performing multiple model training and testing, we consider the best model to be picked according to the accuracy score is the Deep Learning model with an accuracy of 76.28% as compared to other.

This means our model is predicting proper sentiment with an accuracy of 76.28%.

**9. References**

Kaggle - <https://www.kaggle.com/c/movie-review-sentiment-analysis-kernels-only/data>

Google – <https://www.google.com>

**10. Appendix.**

GitHub - https://github.com/anuragvishwakarma/Final-Project.git

**framework to be followed as appropriate for your project>>**

Project title



**Project Report Specifications**

Paper size : A4

Margins :

Left : 1.25"

Right : 1.25"

Top : 1.5"

Bottom : 1"

Fonts : Times New Roman

Fonts Size : Headings-16, Sub headings: 14 & Content (Paragraph): 12

Line Spacing : 18pt

Alignment : Center (Body Text: The body text should be aligned Centre

in the page and content should be justified)

Header : 0.7"

Left side of Header to contain the title of the project

Footer : 0.5"

Left side of the footer to contain the name Manipal Academy of Higher Education, Manipal

Right side of the footer to contain the page number

Total size of the report (in pages): Minimum 40 pages of Project Content excluding title pages, Bibliography and Table of contents Source code to be supplied in CD

References and Bibliography: (APA Style)

Drucker, P. F. (1994). Title:" The Age of Social Transformation. *Atlantic* *Monthly*, *274*(5).

Report to have Covering Sheet, Certificate, and Acknowledgment and Index each in separate pages, followed by the document.

Please note that the report should be submitted in the form of **“BOOK HARD** **BINDING” ONLY**.

**Binding Specifications:** Note that the color of Book binding should be Black with thegolden color font embossed on it (first sheet of project i.e., Roll No., Name, Program name, Project Title, Guide name and Designation). Spiral Binding will not be accepted.



Manipal Academy of Higher Education, Manipal Page **1** of **1**