MSc. DATA SCIENCE & ANALYTICS THESIS

Sentiment Analysis of Bollywood Songs

Author:

SANLEY PATHROSE OOMMEN

Student ID: 21250765

Supervisor:

JOSEPH TIMONEY

A thesis submitted in fulfillment of the requirements for the degree of MSc in Data Science and Analytics 2021-2022

in the

Department of Computer Science

Maynooth University

August 8, 2022

DECLARATION

I hereby submit for evaluation on the programme of study for the MSc. Data Science and Analytics qualification, that this content is entirely my own work and has not been generated from the work of others, unless it has been appropriately cited and acknowledged within the text of my work.

Signed: SANLEY PATHROSE OOMMEN Date: 08.08.2022

ACKNOWLEDGEMENT

I would like to thank my supervisor, Dr. Joseph Timoney, who gave me this opportunity to work on this project and has provided valuable feedback and continuous support throughout the course of time. Finally, I would like to thank my parents from the bottom of my heart because, without their help, this project would have been successful.

ABSTRACT

Users from all around the globe now have access to ever-growing collections of music genres, including Indian music, as the paradigm of music consumption shifts toward streaming services. A song's melody and its lyrics are the two primary components that make up a general musical composition, and this holds true across all sorts of musical genres. There have been several articles that have been published categorizing the feeling conveyed by the lyrics of a song that are written in English. In this paper, the classification of song mood of Bollywood songs is analyzed, solely based on their lyrics on both Hindi and English language. A variety of unsupervised learning methods is used, including the use of Hindi SentiwordNet, a combination of the Word2Vec model and the K-means clustering algorithm, and a combination of Vader Sentiment algorithm and BERT model. mBERT which is a supervise learning approach is implemented and compared with the other unsupervised methods. The original Hindi lyrics and their English equivalents have been included to a new dataset that has been developed to include 1051 Bollywood songs. The primary goal of lyric-based song sentiment categorization is to place songs into one of two categories, good or negative, based on the sentiments conveyed in their lyrics (represented as 1 and -1 respectively). The accuracy and other computational metrics are taken and compared between the models. It was observed that supervised learning algorithm obtained a higher accuracy (60%) than unsupervised algorithms and the VADER-BERT approach observed the highest accuracy among the unsupervised methods with around 58%. It was concluded that there is difficulty in getting a good model performance using unsupervised methods for Hindi songs and a proper dataset organization and parameter tuning could increase the performance.

Table of Contents

List of Figures

List of Tables

Chapter 1: INTRODUCTION

Humans are capable of thinking, comprehending, deliberating, and expressing their ideas on almost anything in the world. Because humans are able to understand and make choices, practically everything affects their emotions, which may be impacted either positively or negatively by the circumstance. These human emotions are studied in Natural Language Processing (NLP) through sentiment analysis.

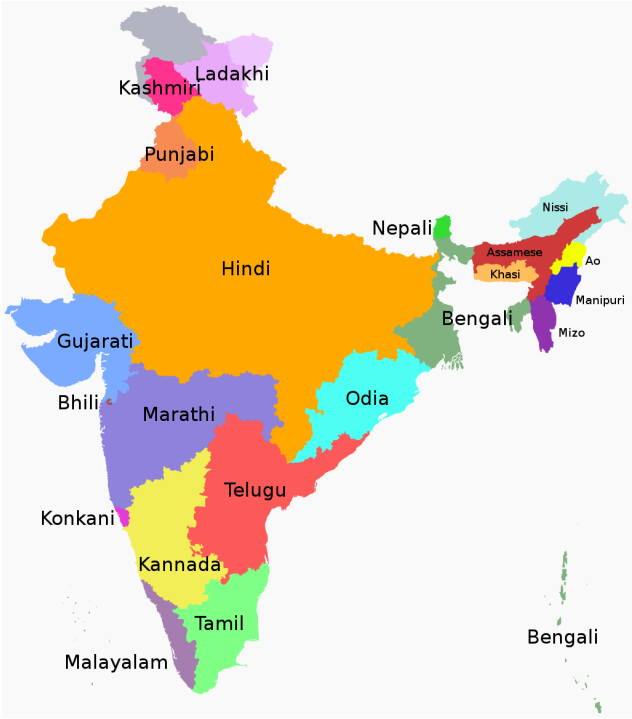

Figure 1.1: Map of India with commonly spoken first Language. Source (Wikipedia)

1.1 Indian Languages

Many people favor using their mother tongue when commenting or expressing their viewpoint because it is the language in which they are most fluent and most ease. The Constitution of India recognizes a total of 22 official languages, and Hindi is the national language. It is estimated that over 42 percent of Indians, as shown in Figure 1.2, speak Hindi

or one of the several varieties of the Hindi language, making Hindi the most widely spoken language in India (Source: (India)). It is evident that consumers are interested in offering their ideas and thoughts in their native language as revealed from a recent survey conducted throughout the world. With the advent of Unicode (utf-8), there is a notable surge in the online content in languages other than English. Google Search engine also offers search using Hindi script. Also, several social networking websites and micro blogging services offer Hindi scripts. This has led to Hindi is being used on the web, in the form of weblogs, blogs, reviews etc.

1.2 Indian Music Industry

According to the IMI or Indian Music Industry, “the Indian Recorded Music Industry has bounced back and exhibited substantial growth of 20.3% in 2021 versus previous year. Further, the Indian Recorded Music Market has been increasing at an annual compound growth rate of 15.78% over the previous five years“ and they are currently ranked 17th in the world, as per IFPI (International Federation of the Phonographic Industry) standards. According to (Pastukhov, 2022) “80% of the country’s music earnings are apparently produced by soundtracks for Bollywood films”. The recorded music sector played a crucial role in helping people cope with the epidemic last year when restrictions were at their peak. The profits of the Indian music business are on a fast increase over the previous several years — mostly driven by the country's rising internet population. Non-film music and independent musicians gained appeal. Local streaming services have achieved general acceptance and launched a new music distribution strategy. As of December 2018, there were reportedly 150 million customers of streaming services in India. All of these indicate that India has one of the most rapidly expanding music industry in recent years.

1.3 Motivation

There have been many analysis done on other music industries, including not surprisingly, industries from USA and England, but only a handful has been done on Indian music, especially in the native Hindi language. Also, most of the sentiment analysis done on Hindi language use supervised learning methods. This project is aimed at analysing the sentiment of Bollywood songs, especially Hindi lyrical songs and advising unsupervised methods which

can be used for this analysis. Supervised learning methods are also used and compared with the unsupervised methods. The results of these models are recorded and compared to see which model would work better.

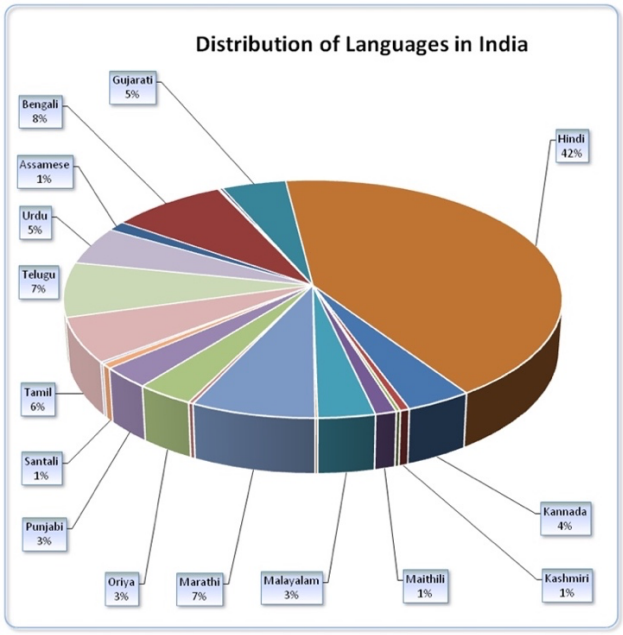

Figure 1.2: Distribution of languages in India. Source (India)

A total of three unsupervised methods were used for this project. The first method included the use of a Hindi SentiwordNet, which allowed to cross check each of the words in the Hindi lyrics with each word in its dictionary and give it a sentiment score. The second method uses a combination of Word2Vec model and K-means clustering method to calculate the overall sentiment of each Bollywood song. The last method uses a combination of the famous VADER sentiment analysis tool from the NLTK library and the BERT model from the transformers library to get the sentiment. Different parameters were used for each method to get the maximum accuracy score. As a first step, the data, which contained Bollywood song lyrics in Hindi language was taken from github sources and was pre-processed in R and python to obtain a clean dataset which can be further utilized to perform the analysis. The Hindi lyrics were also translated to their English version using Google Translate API. The analysis using the Hindi SentiwordNet and Word2Vec – K means clustering methods were applied on the Hindi lyrics, while the VADER – BERT method was used on the English translations of the lyrics.

Furthermore, the dataset was manually analysed and scored by music enthusiasts, giving '+1' for positive sentiment and '-1' for negative sentiment. Neutral sentiment was omitted from this project due to limitations in the analysis.

1.4 Thesis Outline

In the next chapter, background literature review, we are going to look into the recent works done in the field of Sentiment Analysis, with papers referring to analysis on song lyrics and also analysis on Hindi texts. In chapter 3, the overall challenges faced during the collection of data and during the implementation of the models were described. In chapter 4, the methodologies used in the research is described which also includes the metrics and tools used for the project. Chapter 5 describes about the data being used and the pre-processing done on that data for the analysis. In chapter 6, the implementation of how the analysis was done using the combination of models were specified. In chapter 7, the evaluation of the model metrics and graphs was compared and the results were discussed, In chapter 8, which is the final chapter, we come to a conclusion based on the work that was done and some future works that can be done from this project was discussed.

Chapter 2: Background Literature Review

Since the 1990s, researchers in the field of audio signal processing have been looking at song sentiment classification, and the majority of the research efforts that have been done on the topic rely on audio signals to make decisions using machine learning algorithms. There has been a significant amount of investigation into the analysis of music based on its acoustic characteristics; yet, lyrical analysis has not been investigated nearly as extensively. Music mood classification has been the subject of an interesting study, in which the researchers classified the mood of music only based on the lyrics (He, 2008). There are 1000 songs in the database with English lyrics. As part of a categorical approach to mood categorization, the following four moods are considered: happy, sad, angry, and relaxed as well as their complementary categories that are not happy, not sad, not angry and not relaxed. Different supervised learning algorithms have been explored for mood classification, but SVM has consistently produced the best and most accurate results, hence it is often regarded as the best classification model in this context. For classification purposes, the authors employed typical bag-of-words properties such as unigram, bigram and trigram combinations to identify a high-order bag of words. The mood tags have also been analyzed by the writers.

In another study that was carried out (Shukla, et al., 2017) the researchers reviewed how lyrics may demonstrate their usefulness in mood categorization by utilizing linguistic lyric feature set and text stylistic feature set. They have examined the effectiveness of the lyric feature sets both on their own and when combined in a number of different ways. In order to evaluate the effectiveness of the merged feature set, fusion techniques such as feature concatenation and late fusion were utilized.

Dhanashree et.al (Kulkarni & Rodd, 2022) published an article which offers an exhaustive survey of the work done in Hindi Sentiment Analysis. They concentrate on the methods, such as lexicon-based, machine learning methodology, deep learning, and hybrid approach, as well as the capabilities of these approaches to perform sentiment analysis using Hindi reviews as the data. This article also explains the various Sentiment Analysis issues such as sarcasm, negations, and discourse relations that are handled in the Hindi language. It also discusses levels of analysis, performance evaluation measures, datasets, and so on, with respect to the techniques that handle it along with its future scope of research in this area. They came to

the conclusion that there is a requirement for the development of new organised and larger datasets and resources for the Hindi language, which might assist in the implementation of an effective Sentiment Analysis system. According to the outcomes of their research, there is also a need to develop more hybrid systems and sophisticated deep learning models capable of handling all of the issues related with Hindi Sentiment Analysis, as well as a need to increase the accuracy of lexicon-based techniques.

Using a binary support vector machine (SVM) classifier, Xiao et al (Hu, et al., 2009), presented a lyric-based method to mood categorization. In the paper, a vector space model feature set is proposed, which, when paired with existing statistical textual characteristics like part-of-speech tags, results in a more accurate analysis. A private dataset of 5,585 songs is used to analyze the findings, and the 18 mood categories described in (Hu, 2008) are used to categorize the songs. According to the findings, utilizing a mix of lyrical and acoustic characteristics can result in improved classification performance. In later work (Hu & Downie, 2010), the authors investigate several lyrical characteristics and modifiers. These lyrical features and modifiers include stylistic elements as well as features acquired from the ANEW dataset. According to the findings, a classifier that is based on lyrics may perform better than one that is based on audio for some categories of mood.

Revathy et.al (Rajendran, et al., 2022) propose an approach to identify the importance of lyric features in songs using transfer learning and combining pre-trained BERT model. Using state-of-the-art algorithms, lyrical patterns that are important for recognising four emotions were learned. These emotions include: happy, sad, relaxed, and angry. After that, those characteristics were put to use in order to make predictions regarding the emotions included within the lyrics of the Music4All dataset from the Music Emotion Recognition (MER) dataset. It is a large dataset that comprises various information about songs, such as audio, lyrics, tags, and genre, all of which were obtained from the listening histories of 15,000 users. The results reveal that the implementation of BERT embeddings considerably increased the accuracy of all emotions on the dataset. An experimental study is carried out by Frederico et. al. (Souza & Filho, 2022) that includes BERT models trained with Brazilian Portuguese corpora and the multilingual version. This research also includes numerous aggregation techniques and open-source datasets with predefined training, test, and validation divisions to make it simpler to reproduce the results. This study takes into account five datasets of user reviews, each of which is partitioned into a training set, a test set, and a validation set. The pre-trained

BERTimbau embeddings performed significantly better than those associated to the multilingual m-BERT, which is a significant observation from this experiment. Some of the BERT embedding configurations obtained high results out of all the models for all the datasets, with the exception of the UTLC-Movies dataset.

Kim et al. (Kim, 2011) also explore lyric-based mood classification. The method employs partial syntactic analysis in order to extract feelings or moods from a dataset consisting of 500 Korean songs that have been manually labelled. In this study, the authors present a strategy that makes use of innovative aspects such as detection of negation, time of emotion, and changes in emotion. It managed to acquire an accuracy of 60 percent. The researchers Kumar et al. (Kumar & Minz, 2013) take a different method. They utilized SentiWordNet to extract mood characteristics from 185 lyrics that have been tagged with one of the following four mood categories: happy, angry, love, or sad. In this study, three different types of classifiers were compared: k-nearest neighbours, support vector machines, and naive bayes. Of the three, the naive bayes classifier was shown to be the most accurate, with a classification accuracy of up to 81 percent. Joshi et al. (Joshi, et al., 2010) developed Hindi SentiWordNet which is a lexical resource with the help of the existing English one. They created it by using two lexical resources namely the English SentiWordNet and English-Hindi WordNet linking. The Hindi SentiWordNet was developed by replacing the English words with its Hindi counterpart. They achieved an accuracy of about 78%.

The publication (Nanda, et al., 2018), discusses about the Sentiment Analysis of movie reviews in Hindi language. For classification, machine learning techniques such as Random Forest and SVM are employed. As for filtering reviews, however, the Hindi Sentiwordnet is utilized. The method categorized the reviews as having either a positive or negative sentiment, and the performance was assessed using a variety of metrics, including Precision and Recall. They come to the conclusion that the strategy that should be employed is machine learning methods since these methods offered them greater accuracy and other metric values when compared to other ways. Bruno et.al (Ohana & Tierney, 2009) proposes the approach of applying the SentiWordNet lexical resource to the problem of automatic sentiment classification of film reviews. By adding the positive and negative phrase scores, the sentiment orientation is calculated. The method also involves compiling a data set consisting of relevant

characteristics, with SentiWordNet serving as the source, and then applying this data set to a machine learning classifier. This method makes use of the polarity dataset that was presented in the (Pang & Lee, 2004) research. paper. The results of basic word counting were comparable to previous findings obtained using manual lexicons, demonstrating that SentiWordNet outperforms human resources on this task. Furthermore, employing SentiWordNet as a source of features for a supervised learning technique outperformed pure term counting.

The authors (Biancofiore, et al., 2022) developed a processing pipeline with the intention of trying to extract domain-related information from the text. They created a unique unsupervised method for extracting emotional aspects from texts pertaining to a certain domain by first identifying aspect terms and then locating aspect categories to which those terms belonged. They defined a method of document encoding called Aspect-Based Sentiment Embedding, which is based on the sentiment values that are calculated for each Aspect Category. The dataset from Spotify served as the basis for the analysis. The findings were compared with various sentiment based embeddings, such as Doc2Vec, on the song lyrics on Spotify, and improvements in prediction were found, particularly on the Mode and Valence characteristics. On top of that, a T-student test was carried out on the data that was predicted, which, in turn, validates the results that were acquired.

Vinithra et.al (Vinithra, et al., 2015) also explores sentiment analysis on Hindi reviews and does a comparison with different classifiers. On the model, they apply a number of different classifiers, including Naive Bayes, Logistic Regression, and the Random Kitchen Sink method. In order to validate the quality of the classifier that is being utilized, a 10-fold cross-validation is carried out. The F1 score is used to determine the test accuracy, taking into account both precision and recall. All of the models that have been discussed are subjected to an in-depth analysis that compares the accuracy of their unigram and bigram characteristics. When bigrams were utilized, all of the algorithms achieved a higher level of accuracy, as is evident from the results. It has been noticed that, among the other classifiers that were used, Naive Bayes was concluded as the best classifier with an accuracy of 70 percent. The dataset contains a total of 2000 reviews, each of which has been separated into categories of 1000 positive and 1000 negative texts. Out of these 2000 reviews, 1800 reviews were retrieved

from many leading Hindi review websites and the remaining were manually gathered from websites like jagran.com, webdunia.com, bollywoodbhaskar.com and few other websites.

The paper (Denecke, 2008) determines the sentiment of a text in a multilingual framework. The English SentiwordNet is utilized in this suggested method. The document in a different language than English is translated into English using standard translation software. After that, the translated document is evaluated with regard to its sentiment and placed into either the "positive" or "negative" class, as suitable. For sentiment classification, a document is searched for sentiment bearing words like adjectives, so that it may be classified according to that sentiment. A statistical polarity classifier based on n-grams is compared to the approach on the basis of German movie reviews taken from Amazon. SentiWordNet is used to determine the positivity and negativity scores for these words. The document's polarity can subsequently be determined by interpreting the scores. The acquired accuracy in polarity classification is 66 percent, as observed. They conclude that by integrating a suite of current technologies, conventional translation software, and available English sentiment analysis tools, it is possible to categorize texts according to their sentiment with high accuracy.

Jin et al. (Akaishi, et al., 2022) suggested a technique for inferring the emotional content of a song from its Japanese lyrics using a Web search engine as part of a proposal system that matches the listener's mood, but with more accuracy. This is accomplished by the use of technology that infers the emotions portrayed in language. The data came from popular Japanese songs, from which the words were retrieved. For each trial, a different dataset was utilized. The emotion dictionary, which was an expansion of the conventional emotion word dictionary in Japanese, "Emotional Display Dictionary," was also used in the study. They employed a Web search engine to analyse the sentiment of terms that were not in the dictionary, in addition to matching with a basic emotion dictionary. When compared to the traditional dictionary technique, the search engine method improved accuracy by 4 percent. Furthermore, pre-processing on the input system was performed, in which the system corrects omissions of the verbs or particles and inverted phrases, which are commonly utilized in Japanese songs, into regular sentences. According to the findings, the accuracy might be enhanced by roughly 4 percent with this method.

(Anwaar, 2020) explores the sentiment analysis using K-means clustering algorithm where it discusses about the sentiments of women about cloths by using machine learning algorithm k-means clustering in ORANGE, which is a data mining tool. Tokenization, data cleaning and

stop word removal was done on the corpus before sentiment analysis. The sentiment was only divided into positive and negative sentiment. They have also used wordcloud which shows the word frequency in the dataset.

In the publication (Fauzi, 2019), the author uses Support Vector Machines to analyze the sentiment of product reviews written in Indonesian using the features determined by Word2Vec model They evaluated the classification performance of Wor2Vec to that of Bag of Words using Binary TF, Raw TF, and TF.IDF. The experiment used 772 product reviews retrieved from the FemaleDaily website, where the text reviews and their scores were carefully collected and labelled. The experiment demonstrates that the best results may be obtained by combining Bag Of Words characteristics with TF.IDF, as demonstrated by an accuracy of 85 percent. In contrast, the suggested Word2Vec model provided the lowest accuracy, which was still satisfactory at 70 percent. They have arrived at the conclusion that Word2Vec may be utilized for sentiment analysis; however, the accuracy will be influenced depending on the size of the dataset. Word2Vec requires a huge amount of data in order to learn the word representation and put words that are similar in closer proximity. Otherwise, in a small dataset, there too many examples to move the words into the better place.

The research by Hutto and Gilbert (Hutto & Gilbert, 2022) explains about VADER or Valence Aware Dictionary for sEntiment Reasoning in sentiment analysis compares its effectiveness to LIWC, ANEW, the General Inquirer, SentiWordNet, and machine learning oriented techniques relying on Naive Bayes, Maximum Entropy, and Support Vector Machine (SVM) algorithms. A standard list of lexical features along with their sentiment measures are first created and validated and then combined with consideration for five general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity. From their research, they also implemented the model to assess the sentiment of tweets, where it was found that VADER out individual human experts with F1 and accuracy of 96% and 84% respectively. It also performed as well as or even better than the other highly regarded sentiment analysis tools.

(Kumar & Meera, 2022) performs sentiment analysis using k-means clustering and using two sentiments, positive and negative. Twitter data is taken for this experiment. They have found that only when the optimal value for k is 3, can a three – way sentiment analysis is possible. But from the experiments, they have found that this is not always the case where the results when using the dataset were optimal when the k value was 4. They finally conclude that only

when the conditions for Sentiment analysis and k means are met, can the method give good results.

(Ortega, et al., 2013) proposed an unsupervised system, SSA-UO, whose main objective is to analyze the sentiment of tweets. The proposed system includes three stages: data preprocessing, contextual word polarity detection and message classification. The preprocessing phase comprises treatment of emoticon, slang terms, lemmatization and POS-tagging. Word polarity detection is carried out taking into account the sentiment associated with the context in which it appears. Finally, the overall sentiment of the tweet is determined using a rule-based classifier. The experiment showed that the method would be accurate for Twitter sentiment analysis. Two unlabelled datasets were used for evaluation, one composed of messages from Twitter and other of SMS messages. The results obtained were considered to be satisfactory, considering that the system is unsupervised.

From all the above research reviewed, a clear structure on how sentiment analysis should be implemented was conveyed. Also, the importance of SentiWordNet and how it should be used in sentiment analysis was understood. The papers also specified how word vectors representations can be used to boost the model performance for performing the task. An overview of the BERT model was studied and the results show higher accuracy than other models. This was taken into consideration for this project. The process of K-means clustering and the use of clusters in sentiment analysis was also studied from previous works.

Chapter 3: SYSTEM REQUIREMENTS AND CHALLENGES

3.1 OVERVIEW

In natural language processing, sentiment or emotion analysis can be challenging since machines must be educated to assess and comprehend emotions as effectively as the human brain. This chapter describes the obstacles encountered throughout the whole project and the solutions used.

3.2 CHALLENGES IN DATA

One of the most important factor which needs to be considered while doing any form of analysis is the data. There is an abundance of open-source datasets to do research on for any form of analysis. The quality of the data that is being used is equally important as the quantity of the data being used. Most of these datasets would be well organized and structured which would in turn make the pre-processing and the model implementation part easier and give better results. As discussed during Chapter 1, there is an abundance of well-organized and cleaned datasets in English language to perform sentiment analysis. Hence, one of the main and probably the most critical challenge in this project was the data collection. Not many datasets were available which had song lyrics in Hindi. There were Bollywood songs data which were available and used for Sentiment Analysis, but they used the English translation of the songs for the analysis. The dataset had to be manually formed from two git sources which are discussed in Chapter 3. These sources contained the song lyrics in separate files, i.e, each song lyric was in a separate file. These files had to be manually read and appended into the dataframe using Python. Figure 3.1 shows the dataset contents.

The subjective aspect of the text may be identified by a sentence's grammatical structure, which plays a significant part in the sentence. Because English is a language with a specific order, a sentence in English must begin with the subject, then the verb, and finally the object. This is because English is a language with a fixed order. On the other hand, Hindi is a language with a free order, which means that the subject, verb, and object can appear in any sequence. When it comes to determining the importance of a text, this factor plays a very significant

role. The polarity of the text can be shifted by using slightly different wording or rearranging the sequence of the words.

	Title	lyrics
0	aa-jao-na	(तुम थे यहीं, फिर भी तुम गुम थे और मैं लापता अब जो मिले हो तो साथी में रह जाओ ना) x 2 थोड़े थोड़े से पुरे और थोड़े अधूरे ये वादे रहे क्या पता पुरे होंगे कई ख्वाब रह जायेंगे कुछ अधूरे अभी क्या पता (आ जाओ ना आ जाओ ना आ जाओ इतना भी क्या सोचना) x 2 थोड़ा सही इक दुसरे में आ रह ले कहीं आ जी भी ले कब क्यूँ कहाँ कैसे सोंचे नहीं आ चल वादों के भटके हुए जुगनूओं को दिखा दें सही रास्ता आ चल सोते सितारों को होले से सहला के रोशन करें आसमां (आ जाओ ना आ जाओ ना आ जाओ इतना भी क्या सोचना) x 2 आ.. ओ.. शामें कई होंगी ठहरी हुई बातें कई होंगी रूठी हुई छोटी सी जिद्द होगी लम्बी सी रातें फिर भी प्यार रह जायेगा रहता हमेशा तो कुछ भी नहीं फिर भी न जाने क्यूँ मुझको यकीं सब बिताने पर भी सब छूटने पर भी ये प्यार रह जायेगा तुम थे यहीं, फिर भी गुम थे और मैं लापता अब जो मिले हो तो साथी में रह जाओ ना थोड़े थोड़े से पुरे और थोड़े अधूरे ये वादे रहे क्या पता पुरे होंगे कई ख्वाब रह जायेंगे कुछ अधूरे अभी क्या पता (आ जाओ ना आ जाओ ना आ जाओ इतना भी क्या सोचना) x 3
1	aa-re-pritam-pyare-hindi	अये...बंकस आ रे प्रीतम प्यारे बंदूक में ना तो गोली मेरे आ रे प्रीतम प्यारे बंदूक में ना तो गोली मेरे आ रे प्रीतम प्यारे बंदूक में ना तो गोली मेरे सब आग तो मेरे कुर्ती में रे ज़रा हुक्का उठा ज़रा चिल्लम जला पल्लू के नीचे छुपा के रखा है उठा दूं तो हंगामा हो पल्लू के नीचे दबा\xa0के रखा है उठा दूं तो हंगामा हो बंकास... जोबन से अपने पल्लू हटा दूं तो कौले कांवरों का चेहरा खिले हाय मैं आँख मारूं तो नोटों की बारिश हो लॉक जो हिला दूं तो झिल्ला हिले झिल्ला हिले हिले हिले हिले हिले झिल्ला हिले\xa0हिले हिले हिले हिले झिल्ला हिले\xa0हिले हिले हिले हिले ज़रा टूंटी बजा ज़रा ठुमका लगा पल्लू के नीचे छुपा के रखा है उठा दूं तो हंगामा हो हो हो पल्लू के नीचे दबा\xa0के रखा है उठा दूं तो हंगामा हो बंकास..
2	aafreen	तुम रूबरू जो यूँ आ गए हो हमको ना जाने क्या हो गया आँखों से ये दिल धड़का रहे हो सीखा कहाँ से जादू नयासाए सा तेरा साथ हूँ मैं हर कहीं तेरे बिन मैं हूँ भी क्या कुछ भी नहीं आफरीन, आफरीन, आफरीन, आफरीन आफरीन, आफरीन, आफरीन.. बेखबर हम रहे वक़्त से आफरीन उम्र भर हम रहे बस यूँही आफरीन आफरीन, आफरीन, आफरीन, आफरीन आफरीन, आफरीन, आफरीन, आफरीन.. तू जो है मेरे पास में और चहुँ कुछ भी मैं यहाँ जब तू हँसे मेरे वास्ते मैं बिछा दूंगी वहां मेरी जान साए सा तेरे साथ हूँ मैं हर कहीं तेरे बिन मैं हूँ भी क्या कुछ भी नहीं आफरीन, आफरीन, आफरीन, आफरीन आफरीन, आफरीन, आफरीन, आफरीन.. बेखबर हम रहे वक़्त से आफरीन उम्र भर हम रहे बस यूँही आफरीन आफरीन, आफरीन, आफरीन, आफरीन आफरीन, आफरीन, आफरीन, आफरीन
3	aahista	तुम मेरे हो रहे या हो गए या है फ़ासला पूछे दिल तो कहूँ मैं क्या भला दिल सवाल से ही ना दे रुला होता क्या है आहिस्ता आहिस्ता मेरे होना आहिस्ता आहिस्ता मैंने बात ये तुमसे कहनी है तेरा प्यार ख़ुशी की टहनी है मैं शाम सहर अब हँसता हूँ मैंने याद तुम्हारी पहनी है मेरे होना आहिस्ता आहिस्ता होना क्या है आहिस्ता आहिस्ता
4	aainda-hindi	आइन्दा से हम दा इस जहाँ से दुआ से, दजुआ से बेखबर आइन्दा से हम फना इस तरह से रिहा कारवाँ से इस क़दर आइन्दा मेरे दिन तुम्ही से है रौशनी तुम्ही से रात भर दुनिया से परे हवाओं से लड़े तेरी आँखों में जिए तेरे होंठों पे मरे दुनिया से परे फलक से यूँ जुड़े तेरी बाहों में जिए तेरी बाहों में मरे दुनिया से परे.. आ ओ.. आइन्दा से वक़्त भी इक तरह से मेहेरबान सा रहेगा इश्क पर ओ.. आइन्दा से मंजिलें रास्तों पे फैसले रास्तों पे हमसफ़र ओ.. आइन्दा इल्तेजा तुम्ही से हाँ हर वजह तुम्ही से उम भर.. दुनिया से परे हवाओं से लड़े तेरी आँखों में जिए तेरे होंठों पे मरे दुनिया से परे फलक से यूँ जुड़े तेरी बाहों में जिए तेरी बाहों में मरे ओ.. दुनिया से परे..

Figure 3.1: the initial dataset before preprocessing

It is difficult to include all of the possible occurrences of a word in a lexicon since the same word might appear more than once with the same meaning but under a different spellings. When training a model, dealing with all of these spelling variations adds an extremely high level of complexity.

3.3 CHALLENGES DURING IMPLEMENTATION OF MODELS

Most of the models used for normal sentiment analysis are supervised models, mostly Naïve Bayes, Random Forest etc., in the sense that, the algorithm has to be taught in a supervised manner on how each word in a sequence corresponds to the outcome of overall sentence being negative or positive. This approach requires manually labelled data, consisting of input features and a target value. The target value in this case would be the sentiment of the songs. However, as this dataset was a made from scratch, there was no way of including the sentiment values in the target set. Hence, unsupervised learning algorithms had to be used for this project. Also there was not many unsupervised algorithms which could be used for foreign languages. The data had to be translated to English text before using these models. Hence, a sizable challenge was encountered in selecting the proper models for this project.

During the implementation of the Word2Vec – Kmeans clustering method, the data could not be used directly in Hindi language before initializaing the Word2Vec model;. A way had to be found out in using the song lyrics without converting it to English words. Hence, the Hindi words had to be unidecoded to their corresponding ASCII text, which could then be used for the rest of the analysis. Training and testing with the google BERT model was highly time consuming where training for just 10 epochs took approximately five and half hours and the training process needed a lot of memory usage. It is said that In order to run these benchmarks, or be able to fine-tune BERT Large with 4x GPUs, we'll need a system with at least 64GB RAM.

Chapter 4: METHODOLOGIES USED

Sentiment Analysis involves a number of techniques and algorithms to classify the data. It involves theories and techniques from machine learning, computer science, artificial intelligence and statistics. This project uses a number of algorithms or models and methods from machine learning along with software tools like Python and R to get the predictive results.

4.1 Sentiment Analysis

Natural Language refers to the human way of communicating, that is, through speech and text. Natural Language Processing or NLP refers to the branch of Artificial Intelligence that allows machine to understand human language. Sentiment analysis is the contextual mining of text used in Natural Language Processing that detects and extracts subjective information from source material, therefore supporting businesses in determining how others feel about their brand or product. In layman's terms, sentiment analysis aims to provide computers and machines the capability to comprehend the feelings of a user. It monitors conversations and analyzes language using a scoring technique to determine attitudes, opinions, and emotions related to the task.

There are mainly three approaches used in sentiment analysis:

- Rule-based or Lexicon-based: This approach examines text without the use of any training data or machine learning models. This method produces a set of guidelines for categorizing text as positive, negative, or neutral. These guidelines are referred to as lexicons.
- Automatic: Unlike the rule-based approach, this method uses machine learning techniques to learn from the data in order to determine the substance of the text.
- Hybrid: This approach is a combination of both lexicon – based and automatic approach

We are going to use the Lexicon – based approach in finding the polarity using Hindi SentiwordNet, an automatic approach in Word2Vec – K means clustering method and more of a hybrid approach in Vader – BERT model. Polarity categorization is an important part of

sentiment analysis. The overall mood expressed by a certain sentence, phrase, or word is referred to as polarity. This polarity can be quantified as a "sentiment score." This score can be computed for the entire text or for a particular phrase.

A basic sentiment analysis for text classification has the following steps:

1. Split the text into smaller parts, such as sentences, phrases, tokens and POS or parts of speech.
2. Identify the sentiment or the emotion of each component.
3. Assign a sentiment score to each of the component. The most prominently used scores are ‘+1’ for positive, ‘0’ for neutral and ‘-1’ for negative.
4. Combine the scores of the components to get the overall sentiment of the text.

There are mainly two types of text used for sentiment analysis, subjective and objective text. Objective text represents normal words or facts devoid of any emotion, sentiment, or mood. Subjective text consists of text typically expressed by a human with normal emotions, moods, and sentiments. Typically, material having a subjective context works better for sentiment analysis than text with merely an objective context.

There are three levels for which Sentiment Analysis can be done, as shown in Figure 4.1:

1. Document Level
2. Sentence Level
3. Aspect level

The purpose of document-level sentiment analysis is to ascertain the document's overall opinion. This level implies that each document represents a single entity. Document sentiment classification is considered the most basic sentiment analysis work because it approaches sentiment classification as a standard text classification issue with sentiment orientations or polarities as classes. Consequently, any supervised learning technique may be easily applied to solve the problem. In document level classification, rather of assessing multiple entities, opinion about a single entity is identified. Because it is more difficult to reliably classify three sentiments (positive, negative, and neutral), the neutral sentiment is omitted from the most of document-level sentiment classification.

The aim of classification at the sentence level is to classify each sentence in a document as having a positive, negative, or neutral opinion output. Classification at the sentence level is

comparable to the document level since sentences may be treated as shorter version of documents. However, classification at the sentence level is typically more difficult since the information included in a normal sentence is less than that contained in a typical document because they are much shorter than document texts.

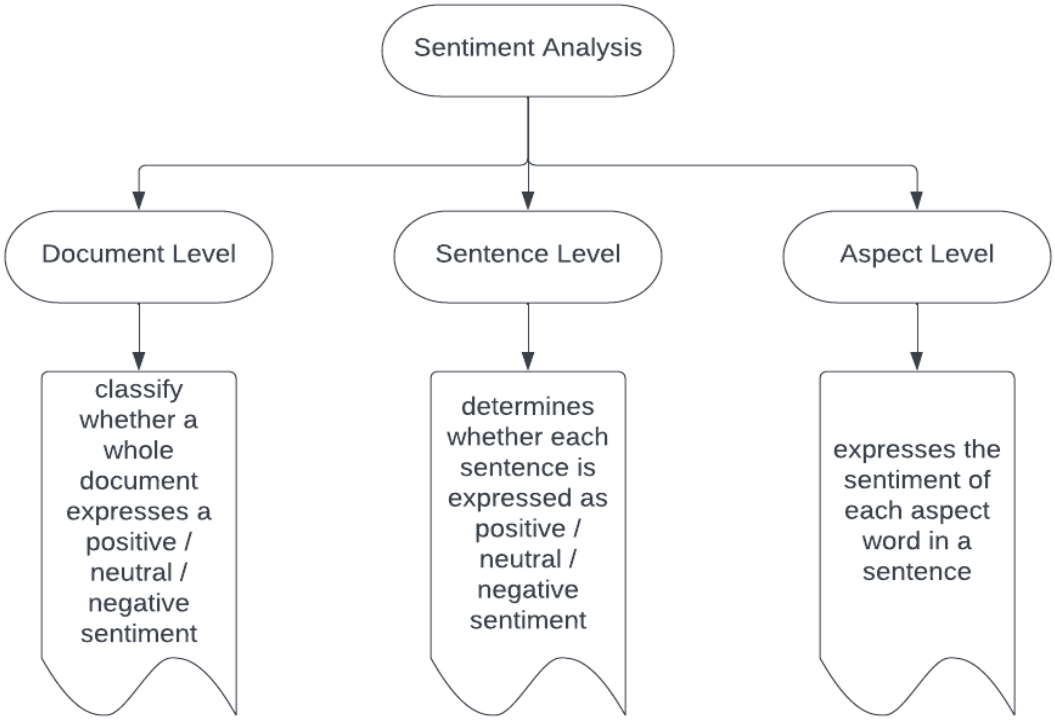

Figure 4.1: levels of sentiment analysis

Due to the variety of sentence forms, such as comparison sentences, conditional sentences, and sarcastic statements, etc., conducting an analysis at the sentence level might be challenging. For sentence level classification, the neutral sentiment cannot be ignored, as an opinion text may contain several neutral sentences. Neither the document level nor the sentence level analysis can determine precisely what people's emotions. Aspect-based sentiment analysis is a more refined classification task that may determine the sentiment polarity of numerous elements in a sentence. Aspect level can obtain a more precise categorization of sentiment by mining the sentiment patterns of many aspects inside comment text, which, to put it simply, implies that instead of looking at the language constructs, it directly analyses the opinion itself. This is mostly founded on the notion that each opinion comprises of a positive or negative mood and a target (of opinion).

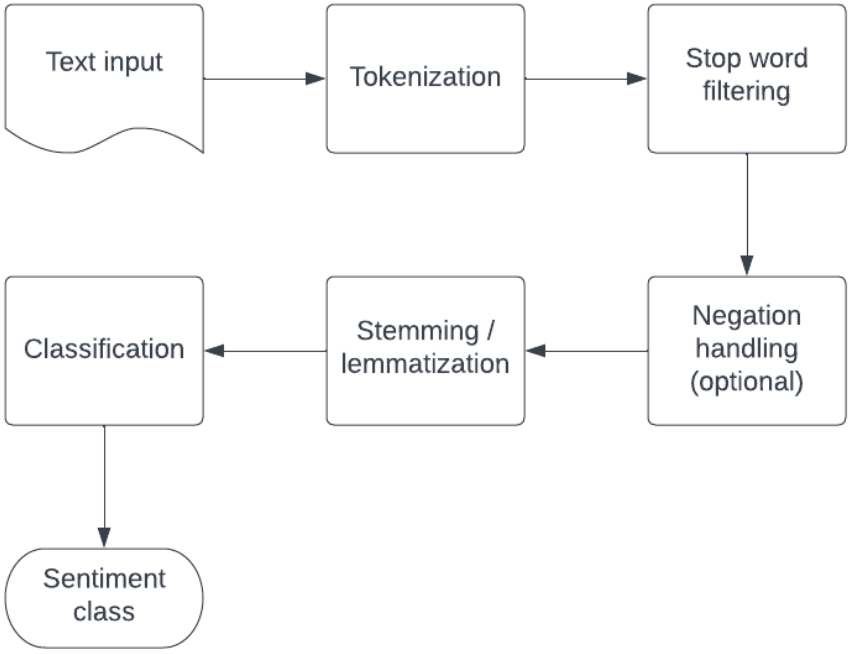

Figure 4.2: general work flow of sentiment analysis. Source (Brown , 2021)

Figure 4.2 represents the general work flow of a normal sentiment analysis. Almost all of these processes are done before classification. First the input is tokenized into individual words using a tokenizer. These individual words are called tokens. These tokens are then filtered to remove stop words. The filtered tokens are then passed through to process the negation terms. Negation handling is the process of automatically identifying the degree of negation and flipping the polarity of opinionated words affected by a negation. The portion of a phrase that is affected by negation is referred to as the proximity or scope of negation. Negative words consist of no, should not, never, not, cannot, would not, etc. Once this is done, the words are then stemmed or lemmatized to get the common base form of the words. All these processes are done during the preprocessing stage. This preprocessed data is then sent to the classification algorithm or model which then classifies the text into positive, negative and sometimes neutral sentiment classes. The explanations for tokenization, stop words filtering and stemming / lemmatization will be discussed further in this chapter.

4.2 HindiSentiWordNet

One of the most important and useful tools for conducting sentiment analysis on any language is a lexicon of emotions. A large number of researchers make use of sentiment lexicons in order to construct unsupervised sentiment models. It can also be used as training features in order to train machine learning algorithms in supervised methods. A sentiment lexicon is a collection of words that are paired with their positive or negative emotional orientation. These terms are also referred to as opinion words.

This project uses Hindi SentiWordNet (HSWN) which is the lexical resource for Hindi language, created by IIT Bombay (Kulkarni & Rodd, 2022). This resource was developed utilizing the English lexical resource 'SentiWordNet' and the English–Hindi WordNet link. SentiWordNet is an opinion lexicon that was developed from the WordNet database, in which each phrase is tagged with numerical scores that indicate positive and negative sentiment information. SentiWordNet is constructed using a method that is semi-supervised, and it has the potential to be a useful resource for carrying out opinion mining tasks. It provides a readily available library of word sentiment data for the English language and has the potential to replace the manual process of constructing ad hoc opinion lexicons. Moreover, SentiWordNet is based on a procedure that is semi-automated, which means that it is easily adaptable to future versions of WordNet and to other languages for which equivalent lexicons are available. It is constructed by assigning each WordNet synset to one of these three groups: positive, negative, and neutral. It assigns a numerical score between 0 and 1 for each term, indicating the degree to which that term applies. They assign polarity at the syntactic level, which means that the polarities of the words are ranked according to the parts of speech that the words belong to, namely, nouns, adjectives, verbs, and adverbs.

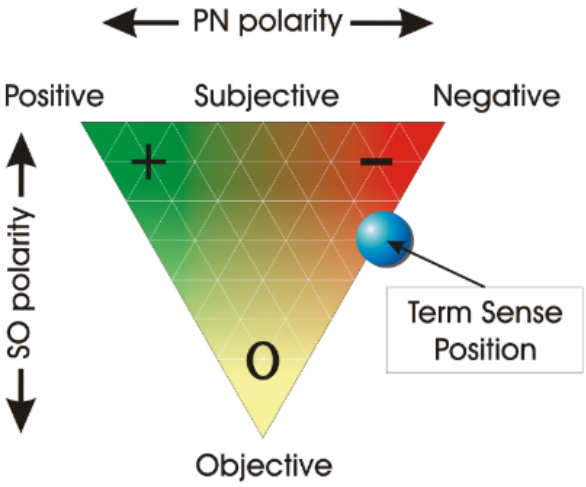

Figure 4.3: The graphical representation adopted by Senti-WordNet (Roul, 2021)

Figure 4.3 represents the opinion related properties of term sense, wherein if the position of the term sense specifies which sentiment is the term more towards to. The PN polarity is the extent of positive and negative scores that the word will possess and SO polarity is used to tell whether a term indeed indicates the presence of an opinion or a subjective term or not. An entry of equivalent scores is produced in the HSWN database for each synset that matches both SentiWordNet and the English–Hindi WordNet. The polarity of every phrase in the document is gathered from HSWN, and the overall polarity is determined using the results based on voting. Despite the fact that the HSWN approach has the lowest accuracy, there is a significant amount of room for improvement in HSWN.

4.3 Word2Vec Model

The Word2Vec model is utilized for analysis since machine learning or deep learning models cannot accept text directly and require some type of numerical representation in order to interpret the input. Word2Vec is a shallow neural network with two layers that is trained to reconstruct the linguistic contexts of words. It contains a single hidden layer, and its purpose is to adjust the weights in order to minimize the loss function that occurs during training.

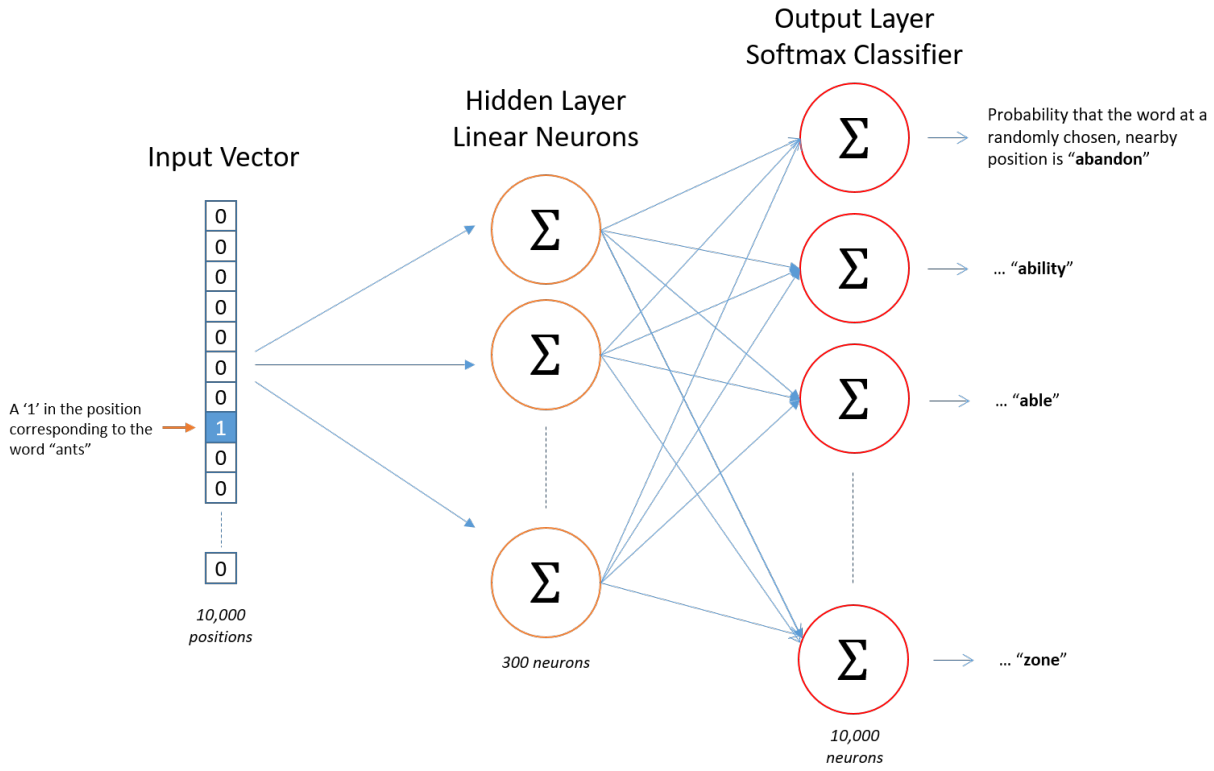

Figure 4.4: architecture of Word2Vec Skip-Gram model. Source (Megret, 2018)

Figure 4.4 represents the Word2Vec Skip gram model with one hidden layer. The skip-gram model is a basic neural network with a single hidden layer that is trained to predict the likelihood of a given word's occurrence when an input word is present. A corpus can be represented as an N-size vector, where each element corresponds to a corpus word. During the training phase, a pair of target and context words are extracted, and the input array will have the value 0 for all entries except the target word. The value of the target word will be 1. Each word's embedding representation will be learned by the hidden layer, resulting in a d-dimensional embedding space. The final layer or output layer has a softmax activation function. The output layer will provide a vector of the same size as the input vector, with each element having a probability. This probability shows the similarity between the target and corpus words. (Vatsal, 2021).

A large corpus of words is taken as the input and creates a vector space that has several dimensions, with a matching vector in the space being assigned to each unique word in the corpus. These are called word embeddings. The placement of word vectors in the vector space is done in such a way that words that are found in the corpus to have similar contexts are placed in close proximity to one another in the space. The Word2Vec method is generally

involved in lots of deep learning methods, even though there is no deep learning involved in generating word embeddings. However, numerous deep learning applications involving text have improved by including Word2Vec embeddings as features.

There are mainly two types of Word2Vec model:

1. Continuous Bag-Of-Words (CBOW) model:-

   The target word(s) is predicted from the surrounding context as shown in Figure 4.5. The word “jumps” is predicted using the surrounding words. This model is useful for smaller datasets.

2. Skip-Gram model:-

   The surrounding context words are predicted from the target word(s) as shown in Figure 4.6. The context words including “brown”, “over” etc. are predicted using the word “jumps”. This model tends to do better when a larger dataset is used.

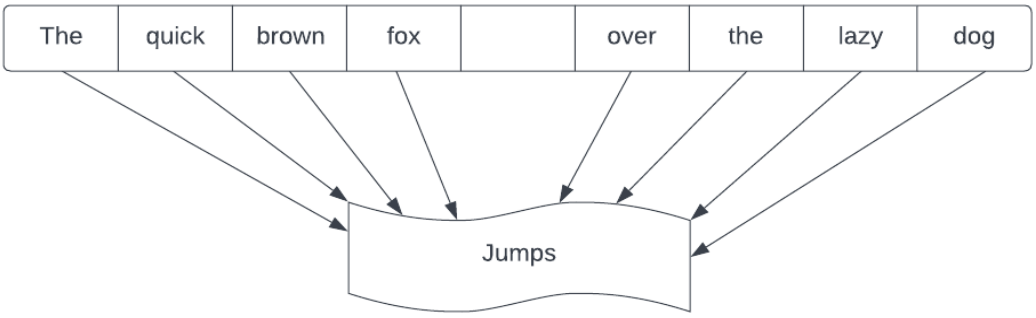

Figure 4.5: example of CBOW method

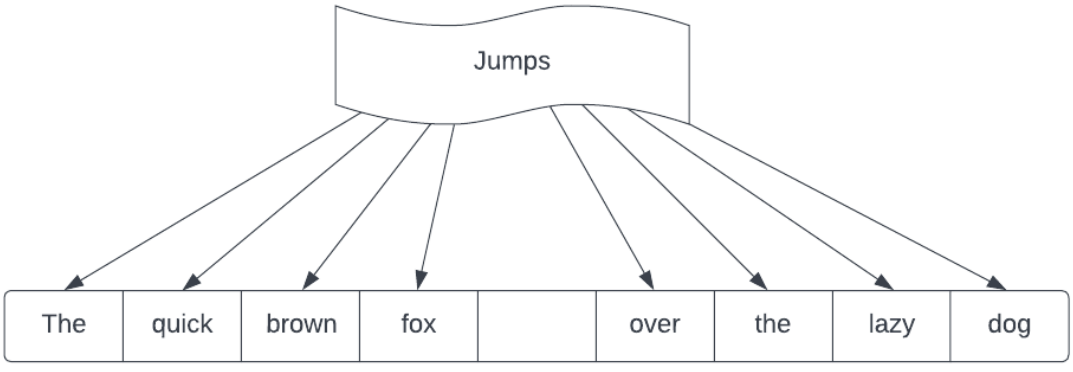

Figure 4.6: example of Skip-gram method

4.4 K-means clustering

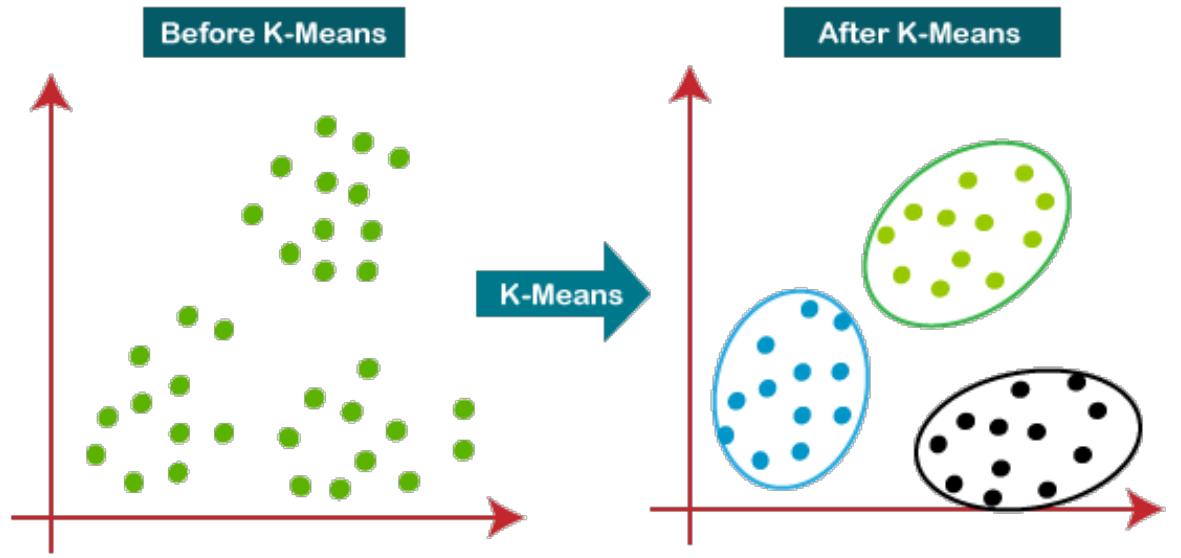

Figure 4.7: clusters formed during K-means clustering. Source (Patel, 2021)

K-means clustering is one of the most widely used methods for unsupervised machine learning. In its simplest form, the K-means clustering algorithm attempts to group similar objects into clusters as specified in figure 4.7. K represents the total number of clusters. A cluster is a collection of data points that have been grouped together due to their similarity patterns.

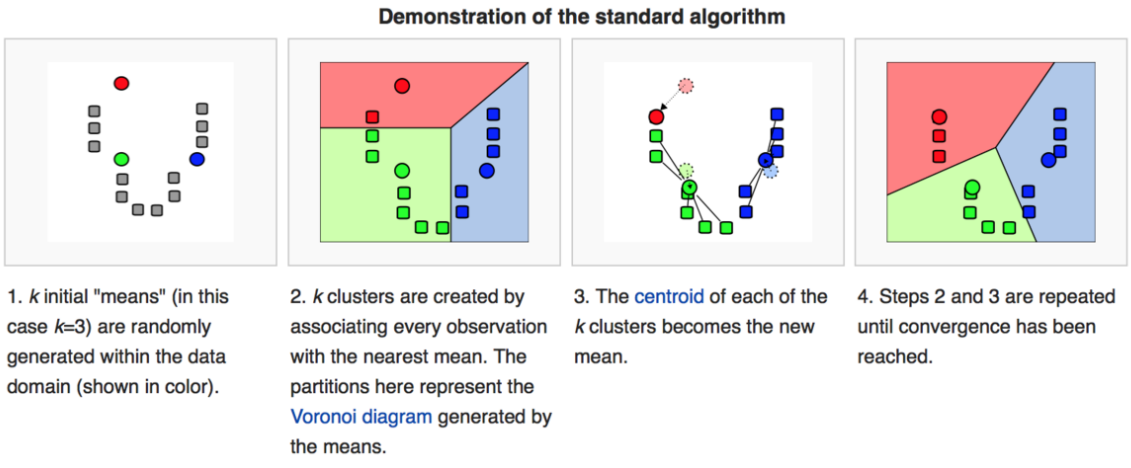

Figure 4.8:steps involved in k means clustering. Source (Gong, 2019)

The basic steps of a K-means clustering algorithm as generally described in figure 4.8 is as follows:

1. A target number k is defined, which refers to the number of centroids you need in the dataset.
2. Initializing centroids, A centroid is the imagined or actual position that represents the cluster's center. Initially, the exact center of data points will be uncertain. Thus, random sites are chosen and specified as the cluster centroids.
3. Assign data points to the nearest cluster, Once the centroids are initialized, the next steps is to assign data points to their closest cluster centroid, where the distance to the data points is minimum.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

   The distance is calculated between the data point X and centroid C using Euclidean Distance metric.

4. Re-initialize centroids, the centroids are re-initialized by calculating the average of all data points of that cluster

$$Ci = \frac{1}{|Ni|} \sum xi$$

5. Repeating steps 3 and 4 until optimal centroids are identified and the data points in each cluster no longer change, or when the defined number of iterations has been reached.

The *'means'* in the K-means refers to the finding the average of the data; that is, to locate the centroid.

The k-means algorithm attempts to minimize the objective function.

$$\mathrm{TWSS} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{m}_i)^2$$

TWSS is the objective function which is the total within cluster sum of squares. Here d refers to Euclidean distance, and m_i is the centroid of the i^{th} cluster.

4.5 VADER (Valence Aware Dictionary and sEntiment Reasoner)

Text sentiment analysis based on VADER was first introduced in 2014 and employs a human-centric strategy. This technique combines qualitative analysis and empirical validation by employing human raters. VADER is a lexicon and rule-based analysis tool that is specifically sensitive to suppositions transmitted in web-based media. VADER makes use of a variety of words that are, marked by their semantic orientation as positive or negative. The VADER sentiment analysis system make use of a dictionary that assigns numerical values, known as sentiment scores, to various lexical features. The results of the VADER sentiment analysis are reported as a score that ranges from -1 to 1, with -1 being the most negative and 1 being the most positive. A text's sentiment score may be calculated by adding up the emotional weight of each individual word included inside the text.

Word	Sentiment Rating
tragedy	-3.4
rejoiced	2.0
disaster	-3.1

Table 4.1: word and sentiment rating in VADER vocabulary

Table 4.1 shows the extract from the Vader vocabulary, where more positive filled words have higher positive score and more negative filled words have higher negative scores.

4.6 BERT Model

BERT, or "Bidirectional Encoder Representations from Transformers," is a natural language processing (NLP) framework that was developed to pre-train deep bidirectional representations from unlabelled textual data by concurrently conditioning on both left and right context. As a result of this, the pre-trained BERT model may be fine-tuned by adding only one further output layer to provide state-of-the-art models for a variety of different NLP tasks. It is pre-trained on a large corpus of unlabelled text including the entire Wikipedia and Book Corpus, which has a total of 3,000 million words. BERT is dependent on a Transformer, which comprises of an encoder to read the text input and a decoder to provide a prediction for the job. There are two models for BERT, BERT base and BERT large. The BERT large has double the layers compared to the base model. By layers, we indicate transformer blocks. BERT-base has 12 layers, 12 attention heads, and 110 million parameters while BERT-large has 24 layers, 16 attention heads and, 340 million parameters as specified in figure 4.9. The BERT-large model is used in this paper.

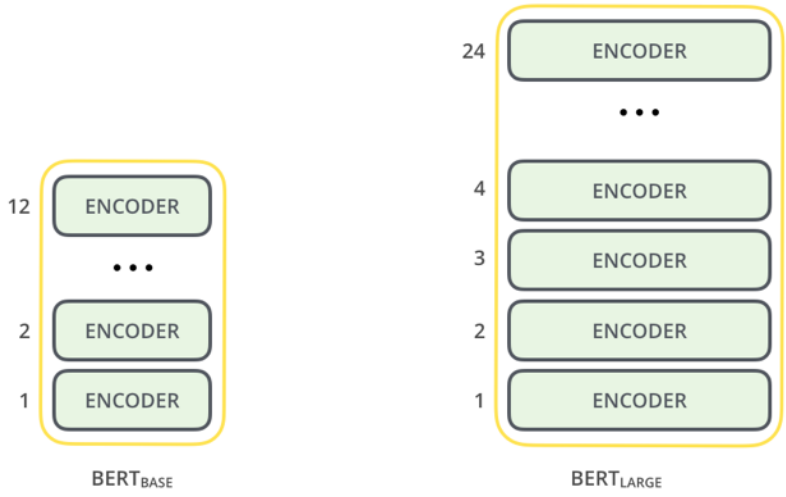

Figure 4.9: source BERT illustration. Source (Akhtar)

Encoder reads the text, while decoder outputs the prediction. Because the objective of BERT is to produce a language representation model, it is only required to include the encoder component. The input for the encoder that BERT employs is a sequence of tokens in succession. Before being sent on to the neural network for processing, these tokens first go through a transformation that turns them into vectors. [CLS] is a special token that appears at

the start of the first sentence. [SEP] is added at the end of every sentence. Figure 4.10 depicts the processing and conversion of these tokens.

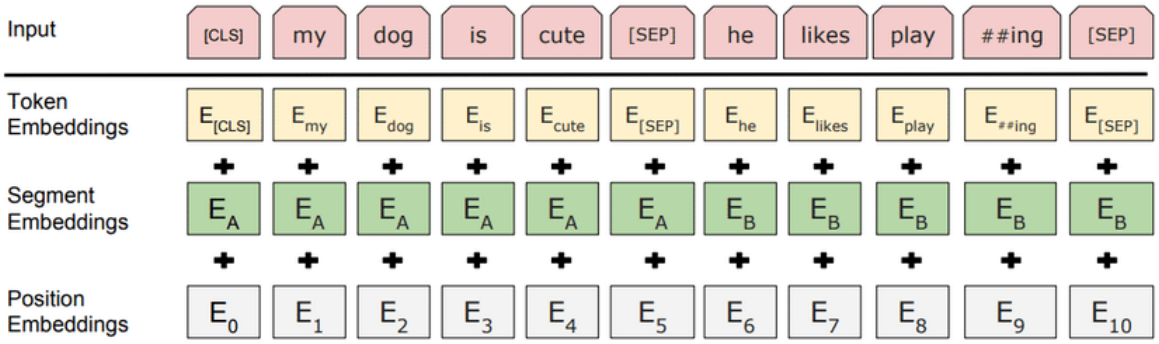

Figure 4.10: BERT adding special tokens. Source (Devlin, et al., 2018)

The accumulation of the three embeddings is the very last component of data that is sent into the BERT Encoder. After receiving an input sequence, BERT will begin an iterative process of moving up the stack. After going through a self-attention layer at each block to begin, the information is then supplied into a feed-forward neural network. It is initially run through a self-attention layer at each block, and then it is fed into a feed-forward neural network. It is sent to the next encoder to process. In the end, each position will produce an output vector with the value hidden size, which will be 768 in our case for BERT Base. This vector will be the word embedding. BERT employs two training methods: pre-training and fine-tuning.

4.6.1 Special Tokens

There are two main special tokens which are processed in BERT as explained by (McCormick , 2019)

[SEP] : The special token [SEP] is appended after every sentence. This token is an artifact of two-sentence tasks, where BERT is given two separate sentences and asked to determine something.

[CLS] : For classification tasks, the special token [CLS] is prepended to the beginning of every sentence. This token has special significance. BERT consists of 12 Transformer layers as described in Figure 4.11. Each transformer takes in a list of token embeddings, and produces

the same number of embeddings on the output but with the feature values changed. On the output of the final transformer, only the first embedding (corresponding to the [CLS] token) is used by the classifier.

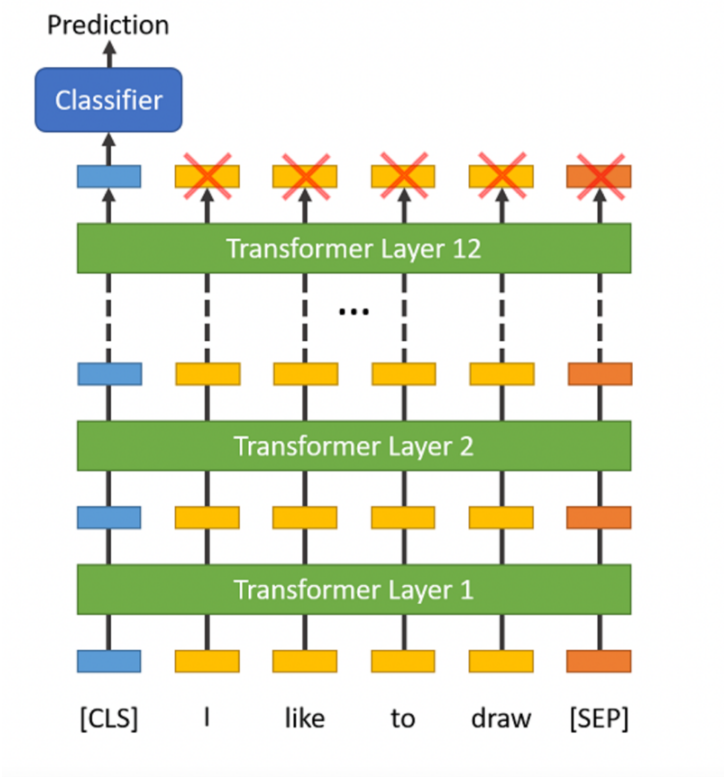

Figure 4.11: Process diagram of BERT with 12 transformers. Source (McCormick , 2019)

A big dataset is used for pre-training, during which time the model is trained to extract patterns. An unsupervised learning challenge like this often requires a large, unlabelled dataset like Wikipedia to train a model. Downstream tasks like as text generation, language translation, classification and question answering etc. are trained on the model during fine-tuning. It is possible to transfer-learn a model that has already been trained to new data using a pre-trained model.

4.7 Multilingual BERT

Multilingual BERT (mBERT) was released along with BERT, supporting 104 languages. It is just a BERT model trained on text from numerous languages in a self supervised manner. In particular, it was trained on Wikipedia content using masked language modelling, with a shared vocabulary across all languages. Small languages were oversampled and major languages were under sampled in an effort to rectify the content imbalance on Wikipedia. mBERT came from the thought that the approaches of Machine Translation of language to english and/or training one BERT model per language would be very time consuming and expensive. The model will learn an internal representation of the languages that are included in the training set. This may then be used to extract features that are helpful for fine-tuned tasks. The MBERT or Multilingual BERT was pretrained with mainly two objectives as per (Hugging Face):

1. Masked language modelling: the model uses masked language modelling (MLM), which involves masking 15% of the words in an input sentence and then running the complete masked sentence through the model to predict the masked words. It enables the model to understand bidirectional representation of the sentence. In simple terms, the model will be able to predict which word can occur in a specific part of a sentence. For example, if the target sentence is “Paris is the [MASK] of France.”, where [MASK] is the word which needs to be predicted, the result shows that the word prediction frequency of “capital” is the higher than other word prediction.
2. Next sentence prediction: Two masked sentences are used as pre-training inputs for the models' next sentence prediction model. A lot of times, they aren't exactly like the original text, but occasionally they are. The model then has to determine whether or not the two sentences are related.

This analysis is done using the pre-processed song lyrics and the polarity column which contains the polarity determined manually by experts. The dataset is first split into train and test data using the train_test_split() from the sklearn library.

4.8 TOKENIZATION

Each sentence derives its meaning from its constituent words. Therefore, by studying the words in the sentence we may quickly comprehend the its meaning. As tokens are the building elements of Natural Language, token-level text processing is the most common method. NLTK Tokenization is used for parsing a large amount of textual data into smaller parts called tokens. The tokenized words can be turned into a data frame and vectorized. The default method for creating tokens is to use a space delimiter. Figure 4.12 depicts an example of tokenization.

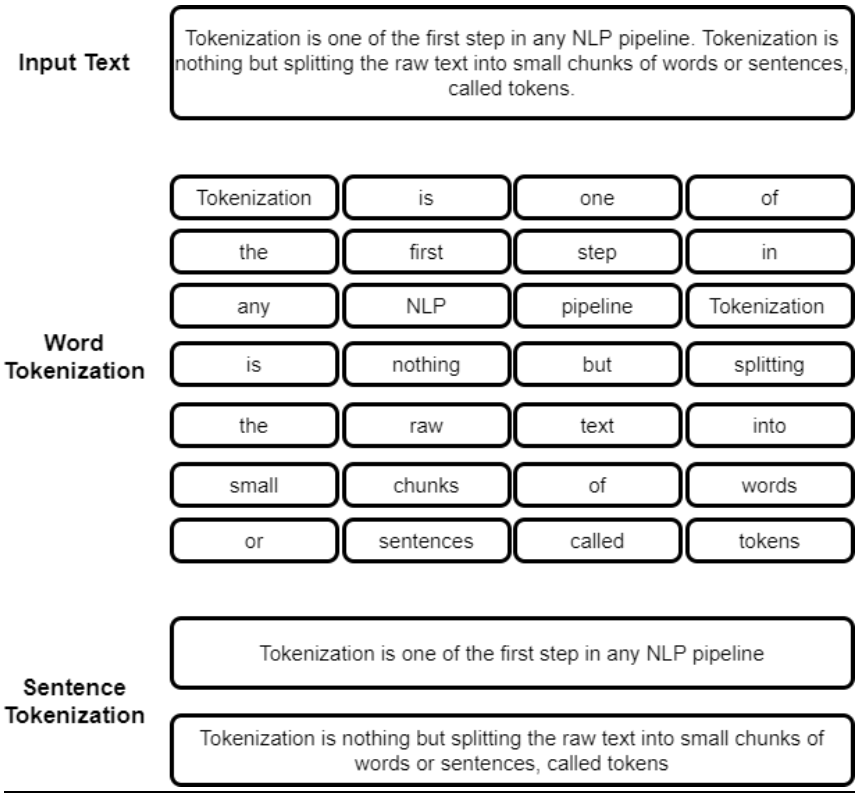

Figure 4.12: Implementation diagram of Tokenization

The Input text, “Tokenization is one of the first step in…” is split into word tokens using word tokenization. The word tokens split here include “Tokenization”, “is”, “one”, “of” etc. Whereas in sentence tokenization, the text is split into sentences like “Tokenization is one of the first step in any NLP pipeline” etc.

Tokenization is the first stage in text data modelling and is conducted on the corpus to produce tokens. After that, the tokens are used in constructing a vocabulary, which is the

collection of distinct tokens included inside the corpus. Word Tokenization is the most used tokenization algorithm. The most often used delimiter is whitespace, which is used to break up long passages of text into smaller chunks or words. Different word-level tokens are created based on delimiters. The accuracy of word tokenization is determined on the language it has been trained on. Tokenization has a hard time separating unfamiliar or Out Of Vocabulary terms. Unknown words can be marked with a simple token to indicate that they are unintelligible. This is not an ideal method, especially considering that the three 'unknown' word tokens may represent three entirely unique unknown words or they could all be the same word.

4.9 TF-IDF (Term Frequency-Inverse Document Frequency)

It is a measure that analyses how important a word is to a document in a collection of documents. (Kumawat, 2020) provides a brief explanation regarding the two terms TF and IDF.

TF or Term Frequency is the contribution of the word in the document where the relevant words in the document should be frequent.

$$TF = \frac{Number\ of\ times\ the\ term\ appear\ in\ the\ document}{Number\ of\ terms\ in\ the\ document}$$

IDF or Inverse Document Frequency is the term used in case that if a word is in the document, then the word might not be relevant to a particular document. However, if the word is in a subset of the documents then the word might be of some relevance to the document it is present in.

$$IDF = \log_{\varepsilon}\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ that\ contain\ the\ term}\right)$$

4.10 STOP WORDS

Stop words are most frequently used words in a language. These words include some of the most often used terms in any language, such as prepositions, articles, pronouns, and

conjunctions; yet, the addition of these words to the text does not bring much information to the sentence. Stop words common in the English language include common nouns and pronouns like "a," "the," "is," and "are". The elimination of these words most probably would not appear to have any adverse effects on the model that we are training on. In other words, removal of stop words will get rid of the low level information in the text, which will allow more focus to be placed on the important information. The elimination of stop words can minimize the size of the dataset, which in turn can cut down on the amount of time spent on training due to there being fewer tokens engaged in the training process.

4.11 STEMMING

Stemming is the process of extracting the base or root form of a word by removing the affixes from that word. It basically use a fixed set of rules such as remove “able”, “ing” etc. to derive a base word The stemmer or stemming algorithm refers to a program that is used to remove stems from words. The operation of stemming algorithms involves removing either the beginning or the end of the word. This is done while taking into consideration a list of typical prefixes and suffixes that are present in stressed words. The process of stemming is included in language studies, including morphology, as well as in information retrieval and extraction used in artificial intelligence.

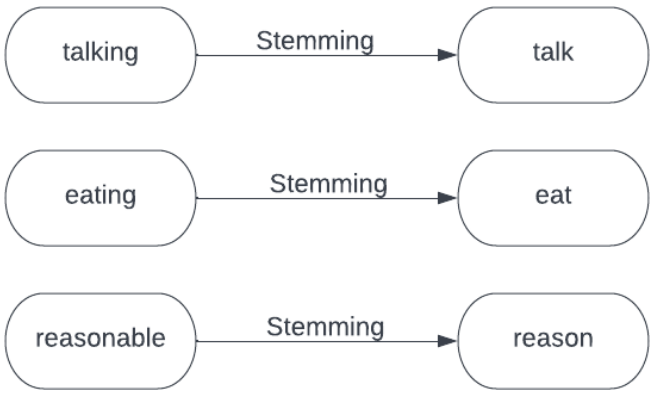

Figure 4.13: example of stemming process

The Figure 4.13 provides some examples of words once they are stemmed to their root word. The stemming algorithm takes in the input word, here say, the word “ talking” and then cuts off the suffix in the word, which in this case, is “ing” and hence forms the root word “talk”.

4.12 METRICS USED

4.12.1 CONFUSION MATRIX

The majority of error measures in a classification model will compute the overall error in a model, however this cannot be utilized to determine the individual errors that occur in a model. Also, during classification the limitation of accuracy has to be overcome. When it comes to classification problems, accuracy might cause ambiguity. A model could have a high accuracy score despite its poor performance in predicting the majority class in every example if there is a severe class imbalance. To solve an issue of this nature, the confusion matrix is employed.

A confusion matrix is a table that helps to visualize the outcomes of a classification task by presenting a table layout of the many outcomes of prediction and findings from the problem. It simply assists in determining whether or not a model's predictions for various particular classes include any faults or whether or not they are accurate. The rows in a confusion matrix corresponds to what the machine learning algorithm predicted and the column corresponds to the known truth.

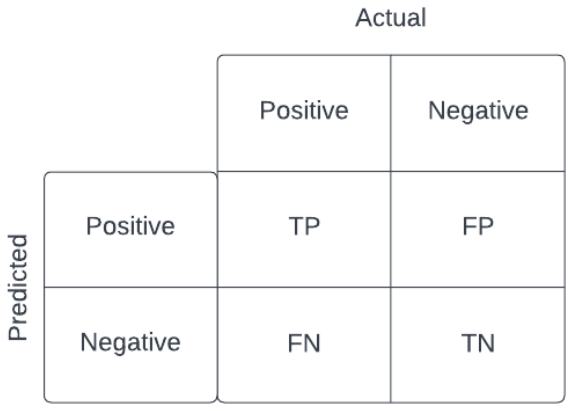

Figure 4.14:Representation of confusion matrix

Figure 4.14 shows how the confusion matrix is represented.

- TP is the True Positive is the number of times the actual positive values are equal to the predicted positive values

- TN is the True Negative which is the number of times the actual negative values are equal to the predicted negative values
- FP is the False Positive which is the number of times the model wrongly predicts negative values as positives
- FN is the False Negative is the number of times the model wrongly predicts positive values as negative values

The confusion matrix can be used to find out four main metrics:

1. Accuracy
2. Precision
3. Recall
4. F1-Score

The Precision, Recall and F1-score are metrics used to measure the accuracy of the model. When a model is built to predict a certain class or category, there is a need to measure how accurate the predictions are. Precision, Recall and F1-score are used for this purpose. The article (Suresh, 2020) gives a brief explanation on the metrics used.

4.12.2 ACCURACY

The accuracy is used to find the percentage of correctly classified label. It is the sum of all true values divided by the total values. The accuracy should be a high value. The higher the accuracy, the better the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The accuracy tells how often the classifier is right.

4.12.3 PRECISION

Precision is used to calculate the model's ability to classify the positive values correctly. It basically answers the question: "When the classifier or model predicts a positive value, how often it is right?"

It is the true positives divided by the total number of positive predicted values.

$$Precision = \frac{TP}{TP + FP}$$

The higher the precision, the better the model.

4.12.4 RECALL

Recall is used to calculate the model's ability to predict positive values. It would be "How often does the model actually predict the correct positive values?"
It is the true positive divided by the number of actual positive values.

$$Recall = \frac{TP}{TP + FN}$$

The model must aim for high precision and high recall, where it avoids as many mistakes as possible, doing a good job at correctly predicting both positive and negative values.

4.12.5 F1-SCORE

If the model is really good at predicting one class however, has problem predicting the other, then it is misleading to look at precision and recall separately. This is when F1-Score is used. It is the harmonic mean of Recall and precision. It is useful when there is a need to take both precision and recall into account.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

A balance of precision and recall is what F1 scores on. The higher the F1 score the better the classifier would be, with 0 being the worst and 1 being the best.

4.13 TOOLS USED

The project was mainly implemented in Python as it provided a better and more stable environment for implementing the unsupervised models. R was also used for some part of data pre-processing. The following libraries/packages were the most commonly used for this project.

- Pandas : it is a an open source data analysis and manipulation tool
- Numpy : a library used for working with arrays
- Sklearn : a library used for machine learning projects.
- Nltk : open source library for tokenization, classification, stemming, tagging etc. It is a platform for constructing Python programs that operate with human language data for use in natural language processing.
- Re : library used to work with regular expressions
- String : it is a module that constitutes of some constants, classes and utility functions for string manipulation
- Transformers : it provides API's to download and use state of the art pre trained models
- Google Translate API : use to translate texts using Google's neural machine translation technology. It can translate texts to more than 100 languages
- Matplotlib : library used for data visualization and graphical plotting
- Wordcloud : library used to initialize word cloud for representation

Chapter 5: DATASET PRE-PROCESSING AND EXPLORATION

This chapter discusses about how the data was collected and how the dataset was formed. It also discusses about preprocessing of the data in general and for separate preprocessing for each model. The preprocessed data is analysed to look for any trends or patterns in the lyrics and in the sentiments which was included using human raters.

5.1 DATASET COLLECTION

It was hard to find a well-organized and structured dataset of songs in Hindi language as they were not that popularly used for analysis, unlike other languages. Most of the Sentiment analysis done on Hindi or even Indian language was by translating the text first to English. Hence, the dataset had to be manually created from scratch.

The data was taken from two individual sources. The first was a collection of files from Bhavya Pant's "bolly_song_lyrics" git repository (Pant, 2020) which contained over 1200 Bollywood songs in Hindi, with the song title as their file names and the song lyrics as the content within the corresponding file. The files were in txt format, and hence the filename would be "song-name.txt". The data contained some English language data, due to the fact that Bollywood songs are bilingual. Most of the Bollywood songs contained some kind of English words at the beginning or end of the song. A temporary dataset was created by reading each file content, which is the song lyric, and the file name excluding the txt extension, into the dataframe in the 'lyrics' and 'title' column respectively, in each iteration. The encoding used was 'utf-8', which was the supported type for Hindi language.

The second source was from the git repository (Stephan, 2019), which contained around 20 songs, categorized into different folders based on four main emotions, angry, sad, relaxed and happy. The title of the song was taken from the file named 'info' situated in each folder. This file contained all the meta-data of the songs in that folder, including the title, singer, name of the film etc. The folder contained Hindi and English version of the songs in individuals files, with the file name of the English version having '_E' at the end.

The songs from the two sources were combined to make up the dataset used for this project. The final dataset contains around 1047 songs, after pre-processing. The initial dataset

includes the title of the song with column name as “Title”, the song lyric in Hindi language as “lyrics” and the English translation of the lyric as “lyrics_eng”. The English version of the set of songs from the first data sources was obtained by translating the Hindi versions using Google Translate API in Google Sheets. As for the songs in the second data source, the English version was taken directly from the folders. The polarity of each song was also manually analysed by different music enthusiasts for the songs which were taken from “bolly_song_lyrics” git repository. They gave each song a score of ‘-1’ if the song portrayed negative sentiment and ‘+1’ if the song portrayed positive sentiment. The polarity of the other data source was scored based on the category of emotion it was included in. For instance, “Angry” and “sad” emotion songs were marked ‘-1’ (negative sentiment) and “happy” and “relaxed” songs were scored ‘+1’ (positive sentiment).

5.2 DATA PRE-PROCESSING

As this was a very new data to work with, it was complicated to properly organize and clean the data to how it needed to be. The first and most important part was check if there was files which were not songs and if it had any other junk data. While investigating the files of the first data source, it was found that some of the song files did not have the complete song lyrics, some even having only a few words. For example, a file in the raw data source, named “aaj-ki-party-hindi.txt” only had 2 lines, with around 8 words in total. This would not do justice to the analysis, hence, they had to identified and removed. Hence, a threshold limit of 6 lines was set in order to consider the song for analysis and this threshold was used when creating the final dataset. This would mean that all the songs which contained less than 6 lines would not be included. Further observation revealed that some songs lyrics were coming more than once. This was because there were files containing the remix or reprise version of the songs. Remix or re-prise version has the same lyrics of the original song and their audio features would be the main difference. An instance during the observation were the files “tu-jo-mila-hindi” and “tu-jo-mila-reprise-hindi”, having the same lyrics but different file names. The data also had many files where the content were not lyrics but some other instances from the movie. This was found in a row where the song had no lyrics. It only had the dialog from the movie, which is the starting point of the song in the movie. All these were removed from the dataset. The shape of the dataset is 1047 rows and 4 columns.

The combined dataset was then loaded into Rstudio and some part of the pre-processing was done using the gsub() in R. It is a function which is used to replace all the matches of a pattern in a given string with the replacement string provided. If the pattern is not found in the source string, then it will be returned unchanged. Even though python's string library has the feature to remove string punctuations, it seemed that most of the punctuations used in the dataset were sort of different from the normal ones. Apart from the conventional symbols like (.") some other symbols (…"") were found in the dataset, which looked almost similar to the conventional ones but are different. This might have been due to the encoding done on the dataset. This can proved by their Unicode representation. For instance, the normal period symbol (.) has a utf-8 code unit of "2E" however, the character found in the dataset (…) had a utf-8 unit of "E2 80 A6". Another instance can be found where the conventional double quotes character (") has a utf-8 unit of "22" while the encoding of the symbol found in the dataset was "E2 80 9C". This meant that the normal string punctuation function in the python library could not detect these type of symbols. Hence, they had to be removed separately. The symbol (…) could not be detected in Python, hence, those parts of the pre-processing were done in R. The following pre-processing steps were done overall:

1. Removed some of the punctuations individually because some of them could not be detected by Python's string punctuation function. Some of these punctuation include … , " , " etc.
2. Unicode characters where found in the dataset, like "\xa0", which is used to indicate a non-breaking space. This is also a result of the encoding when the dataset was created. These characters were removed
3. Extra spaces occurring between sentences or words were removed
4. Normal string punctuations were removed using the Python string library
5. Normally when writing song lyrics, the sentences in the song that are repeating would be represented by the sentence and a specific number after the sentence depicting the number of times that phrase needs to be repeatedly sung. Hence, it is essential to remove these numbers as part of pre-processing. As the dataset was in Hindi language, the numbers used in the lyrics were also in Hindi script. The number representation in Hindi is stated in Table 5.1
6. There were some English digits (0 – 9) in the dataset that was used instead of the Hindi number representation. These were also removed.

Hindi representation	Numbers
१	1
२	2
३	3
४	4
५	5
६	6
७	7
८	8
९	9
१०	10

Table 5.1: Hindi representation of numbers

All the English words in the songs were also removed and this was added to a new column "hin_cleaned". The same pre-processing steps were done on the English version and added to the new column "eng_cleaned". The shape of the final dataset was (1051,6).
The next phase is to pre-process data according to each method.

5.2.1 Hindi SentiWordNet

Other than common pre-processing steps, the most important step in data preparation is removal of stop words. There are some libraries which can be used in python to remove the stop words in most prominent languages. One of the most commonly used is the nltk library. Unfortunately, there is no library supporting Hindi language. Hence, for the analysis using Hindi SentiwordNet, a manual list of stop words had to be used. This list was downloaded from the git repository (goyal, 2018) which contained almost all of the stop words for the Hindi language.
Then tokenization is done on the data where the 'word_tokenize' function from the nltk library was used to convert the words in the lyrics to individual tokens. The word_tokenize() function to split a sentence (here the song lyrics) into words as shown in figure 5.1. The final output text after stop words removal and tokenization was used for further analysis

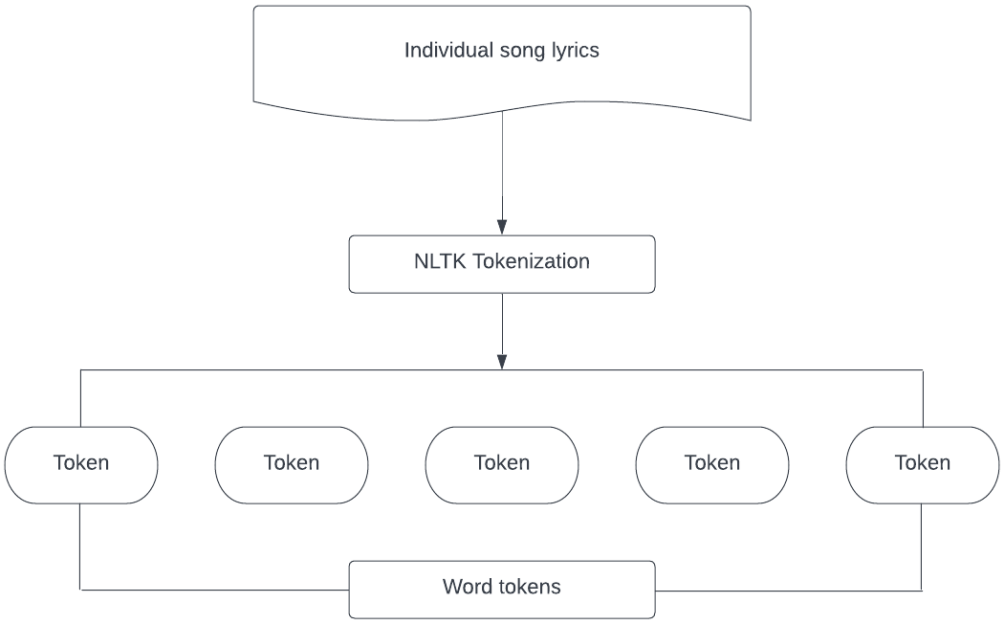

Figure 5.1: tokenization done on the song lyrics

5.2.2 Word2Vec – K means clustering

For this model, the dataset is first pre-processed using the normal steps as stated before. Then the Hindi stop words are removed from the lyrics as the same as that done for the Hindi SentiwordNet method.

The next phase is to decode the Hindi words using the unidecode module. The unidecode function accepts string values and *returns* a unicode string in Python. From the python projects website, "the function unidecode() takes Unicode data and tries to represent it in ASCII characters, where the compromises taken when mapping between two character sets are chosen to be near what a human with a US keyboard would choose." In laymen's terms, the function takes a non-ASCII string object and returns a string that can be encoded to ASCII. Some of Hindi words from the lyrics dataset after them being decoded is shown in Table 5.2.

Hindi word	ASCII format after decode
लापता	laaptaa
दुनिया	duniyaa
मिलेगा	milegaa
करदा	krdaa
दुआयें	duaayen

Table 5.2: Hindi words and their corresponding ASCII format

The text was decoded to ASCII inorder for the Word2Vec model to understand and build the vocabulary from the sequence of sentences. This is important for initializing the model. Finally the decoded lyrics are then split or tokenized into individual words using the text split function.

5.2.3 VADER – BERT and mBERT

As for the VADER – BERT and Multilingual BERT method, there was no extensive pre-processing done. The language used in this method had to be English, hence, the songs from the first data source were translated to English using the Google Translate API. The stop words from NLTK library was used to remove the common English stop words from the data.

5.3 Exploratory Data Analysis

In this section, the pre-processed data was analysed to look for any patterns or trends. For the data exploration, the following questions were determined:

1. Which is the most frequent words occurring in Hindi songs?
2. Which sentiment was mostly described by the songs based on human evaluation of the sentiment

To find out the most frequently used words, the Word Cloud method was used for graphically representation. According to the blog posted by boostlabs (Boost labs, 2014), “A word cloud is a collection, or cluster, of words depicted in different sizes. The bigger and bolder the word appears, the more often it’s mentioned within a given text and the more important it is”. This

representation was used on both the Hindi Words and their English translation to see if there was any difference when the words are translated. The figures 5.2 and 5.3 show us the frequency of words in both Hindi and English language.

Figure 5.2: Wordcloud in Hindi

It can be seen in the Hindi WordCloud representation that the most commonly used words are “प्यार”, “दलि”, “आँखों”, “दुनिया”, “रंग”, which means love, oppressed, eyes, world, colours. Hence, it can be assumed that most of the Hindi songs in the dataset convey a positive mood in the sense that most of the frequently occurring words are positive words. There is only one frequent negative word occurring, which is “दलि”, meaning oppressed, but most of the other negative words are not that frequently used in this dataset.

Figure 5.3: Wordcloud of english lyrics

When looking through the English representation, almost the same words, including love, heart, mother and eyes can be seen on both the Word cloud representations. There is no change in the sentiment when looking at the English translations, with the positive words like “heart”, “love”, “eyes” having high frequency. It can also be seen that there is Hindi-English mixed translation in the dataset. For example, “dil” as seen in the English representation, translates to “heart”, but was not translated correctly to English by Google Translate API. There are many such words, including “tera” means “yours” and “tu” means “you”, which were not translated properly. This can be seen as a drawback of Google Translate API.

As specified before, the dataset was manually analysed by music enthusiasts and was graded accordingly. Their outputs were written into the “polarity” column of the dataset. While doing the analysis, it was intriguing to find out how they graded the songs. It was seen that majority of the songs were graded positively, with the positive sentiment graded more (584) than negative sentiment (463).

The following histogram in figure 5.4 shows the difference in the word counts in both the sentiments. It can be seen that songs having word counts uptil 200 words having higher frequency than other songs. Also, the positive songs have higher frequency than the negative songs of the same range, with the frequency increasing to around 500 for positive sentiment for the songs with words ranging till 200 words, while the negative sentiment only has around 400 for the same word range.

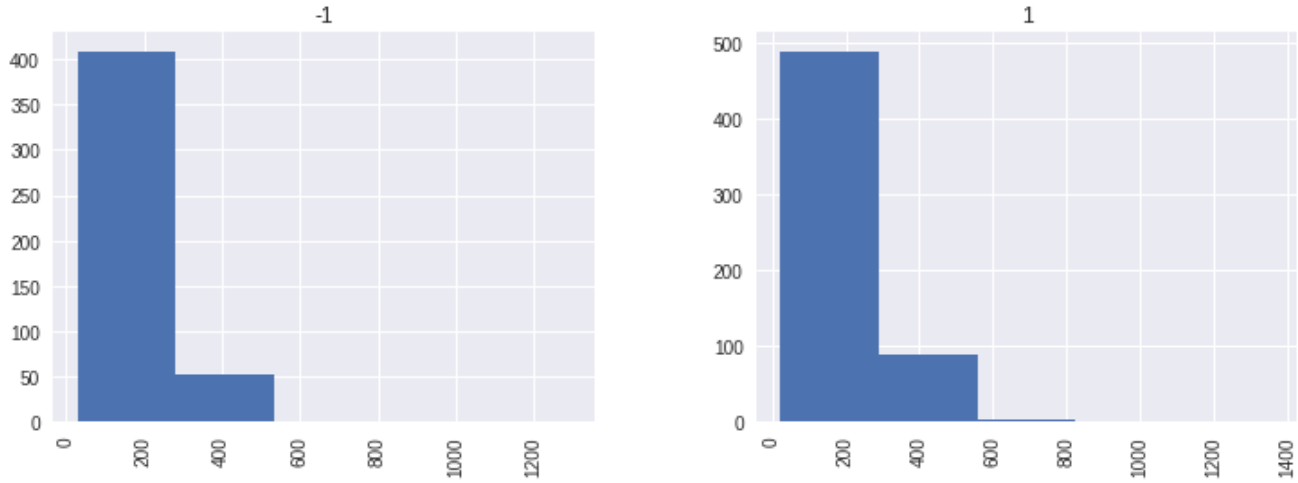

Figure 5.4: word counts based on sentiments

Chapter 6: IMPLEMENTATION

This chapter briefly describes how the different models were combined and implemented for calculating the sentiment. The methods used in this project include the following:

1. Hindi Sentiwordnet
2. Word2Vec, K-means clustering and tfidf weighting
3. Vader and BERT base model
4. mBERT or multilingual BERT model

6.1 Unsupervised methods

6.1.1 Hindi SentiwordNet

This method is used to predict the sentiment of the songs is to use the prior polarity of words present in data. In this method of classification, the Hindi SentiWordNet is used for developing this sentiment classifier.

The HSWN dictionary consists of list of words with each having a certain positive or negative sentiment score. A polarity is given to each word in a song, and that the overall polarity or sentiment is determined by the maximum of all of the scores. The emotion of a song is anticipated to be the polarity that covers the most number of words in that song. Following is a brief algorithm description of what was done in the project:-

Input:

1. The pre-processed song lyrics, **S** in string format. **S** = {p_1, p_2,.., p_i, n_1, n_2,.., n_j}, where i = number of positive songs and j = number of negative songs
2. Hindi SentiwordNet dictionary "**dict**" which contains a tuple for every word having its positive and negative polarity.

Output:

A list **P** containing polarity of the song lyrics

Algorithm:

Create an empty list for polarity P = []

FOR each pre-processed song lyric s_i in S

```
Tokenize the words in s_i
Make a list of votes v = []
Initialize two variables, pos_polarity=0 and neg_polarity=0
FOR each word w in s_i
        If w exists in dict
                Positive score and negative score are extracted from dict[w] to
                pos_score and neg_ score
                If pos_score > neg_score
                        append '1' to v
                        pos_polarity = pos_polarity + pos_score
                else
                        append '0' to v
                        neg_polarity = neg_polarity + neg_score
                else
                        ignore the word
ENDFOR
x = count of positive votes (1) in v
y = count of negative votes (0) in v
if x > y
        sentiment = 1 (here 1 denotes positive)
else if y > x
        sentiment = 0 (here 0 denotes negative)
else
        if pos_polarity < neg_polarity
                sentiment = 1
        else
                sentiment = 0
P.append(sentiment)
ENDFOR
```

The algorithm is applied on each of the song lyrics using the pandas apply() function.

6.1.2 Word2Vec - K means clustering – tfidf weighting

This approach is an unsupervised method, where the main idea is that no previous assumptions are given to the model about the outcome of the features which are fed into it during testing. The preprocessed data is inserted and the model learns the structure of the data by itself. The concept of transfer learning is used in this approach, where some model (in this case, the Word2Vec model) is first trained in an unsupervised manner and then fine-tuned on a certain task (sentiment classification). This trained model is also used in another model to produce better quality features. The unsupervised task (actually fake supervised) for a word2vec would be to predict the word based on the words that surround it or predicting the surrounding words based on the given word.

The main idea behind this approach as given by (Wójcik, 2019) is that negative and positive words are surrounded by similar words. For example, the word 'tedious' would be surrounded by the same words are 'boring', and sometimes, these words would have somewhere close relationship with the word 'didnt'. And it would be very unlikely that the word 'tedious' would have similar surrounding words with the word 'exciting'. When this is taken into consideration, it is possible for words to create clusters of negative words and positive words with similar surroundings, and some neutral words in between. There are several ways to make use of the word2vec model, such as using the vector representation of the words to feed into text classification or clustering algorithms, or in finding relationship between words in the data. Main steps include forming of negative and positive clusters in word vectors space using K Means clustering algorithm, which were then used to transform every sentence into vector of replaced sentiment scores for a given words in a sentence. Then the vector for given sentence was obtained through replacing all words in a sentence with their corresponding tfidf-scores. Final prediction was obtained as a dot product from these two vectors for each sentence, that is, if their dot product was positive, the overall sentiment was predicted as positive, and if dot product was negative, overall sentiment was predicted as negative.

Figure 6.1 describes the general work flow for this method. After the preprocessing stage which includes data preprocessing and stopword removal, the dataset is then saved to be used for tfidf scoring which will be discussed further down in this chapter. The next section is to generate bigrams or phrases for minimizing the amount of surrounding words.

Generating Bigrams

The Phrases package is used to automatically detect bigrams (common phrases) from a list of sentences. The Phrases() is used to cut down the consumption of memory from the Phrases function by discarding the state of the model which is not strictly needed for detecting bigrams.

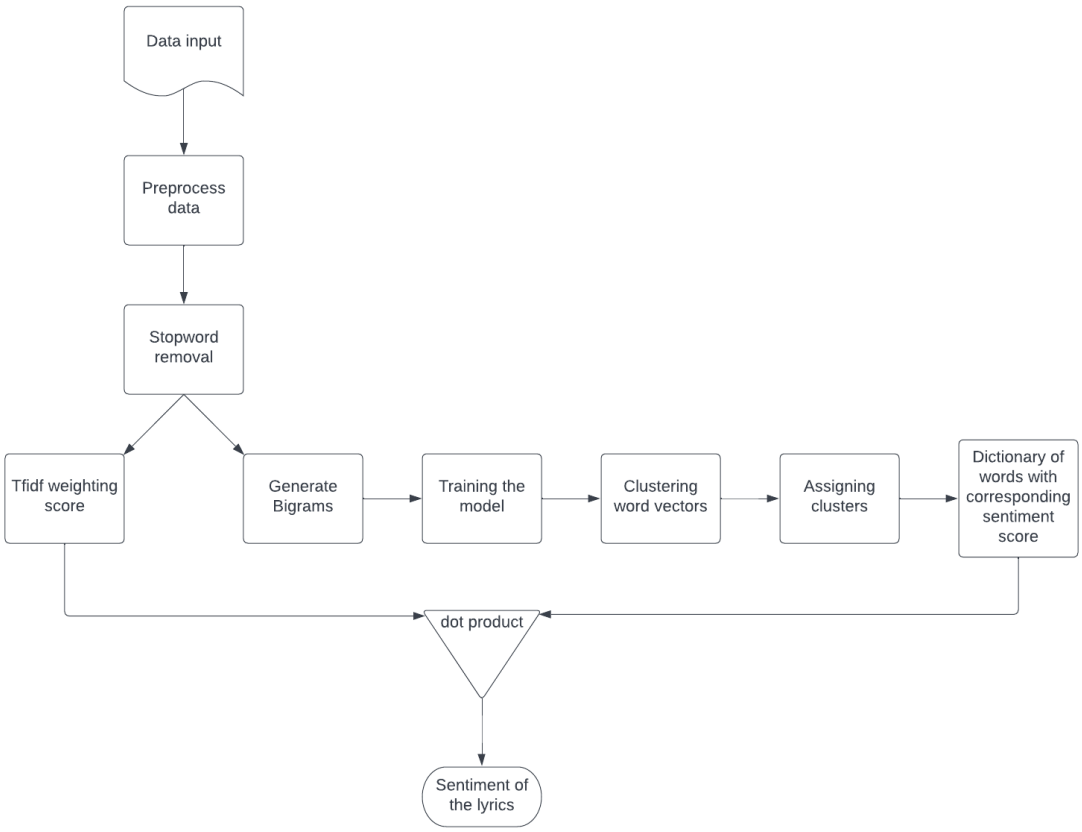

Figure 6.1: work flow diagram of word2vec - clustering - tfidf scoring

Training the Model

The word2vec model is trained on the phrases generated, by splitting the training phase to the following steps:

1. Setting up the parameters for the model one by one (Megret, 2018).
    - min_count – ignores all words with absolute frequencies less than the value
    - window – the maximum distance between the current and predicted word within a sentence
    - size – dimentionality of the feature vectors

- sample – the threshold for configuring which high frequency words are randomly downsampled
- alpha – the initial learning rate
- min_alpha – learning rate will linearly drop to the min_alpha as the trainig progresses
- negative – if greater than 0, negative sampling will be used. The value negative specifies how many 'noisy words' should be drown. If value is 0, then no negative sampling is used
- workers – use these many worker threads to train the model
- sg – if value is 1, then skip-gram method is used, else CBOW method is utilized

2. Building the vocabulary from the bigrams, which simply digests all the words and filter out the unique ones and does basic counts on them. Hence this initializes the model.
3. Training the model. The parameters used are:
    - total_examples – count of sentences
    - epochs – number of iterations

Assigning Clusters

The next task is to assign the word vectors into clusters. K-means clustering algorithm was used on the word vectors to cluster them into two: mainly clusters containing positive and negative words. The two clusters are then evaluated manually and the cluster containing positive words and negative words are determined. Changing the parameters when initializing the word2vec model influenced the words which were put in each cluster. The cosine distance is used to determine the sentiment of each cluster and the Euclidean distance is used to assign each to word to the corresponding cluster. The word sentiment score was assigned to each word, with the values being -1 and +1 based on which cluster they were on. The sentiment coefficient was then calculated by multiplying the sentiment score with the closeness score (which gives how close they were to their cluster). As a result, a comprehensive dictionary was developed, with a weighted sentiment score associated with each individual word. This sentiment score is then multiplied with the tfidf weighting, which is explained in the section.

tfidf weighting and prediction of sentiment

The Tfidf vectorizer was utilized in order to arrive at the tfidf scores for each individual word. It displayed the uniqueness of each word in each phrase and increased the positive or negative score associated with terms that are extremely specific to a given sentence compared to the whole corpus. Then, each lyric word was substituted with its respective tfdf and sentiment score. The dot product of these two vectors represented the song's overall emotion.

6.1.3 VADER – BERT

This approach first uses VADER sentiment tool to generate the sentiments for the training and validation sets in order to use them in BERT's fine tuning process. The sentiment column from our dataset which was manually labelled was dropped for the training and validation sets, but left in the test set for evaluating the efficiency of the model. The fine tuned BERT is then applied to the test set get the sentiment predictions. This is compared with the manually obtained sentiment values to get the overall accuracy of the model.

Figure 6.2 describes the work flow diagram of this approach which will be explained in the below sections.

Generating sentiment labels using VADER

The idea of neural networks is the ability to train themselves by looking at examples. If examples contain both source feature (song lyrics) and target labels (each lyric's sentiment), we have supervised learning like BERT. However, when the model needs to figure out everything without providing the correct labels, it is called unsupervised learning. So, we need the target labels to be able to train a model like BERT. The VADER algorithm is used for generating the sentiments from the unlabelled dataset and the algorithm used for this approach returns 0 for negative sentiments if VADER's positive score is less than the negative score and 1 for positive sentiment if VADER's positive score is higher than the negative score. The algorithm is used on both the training and validation sets and the results are put in a new column.

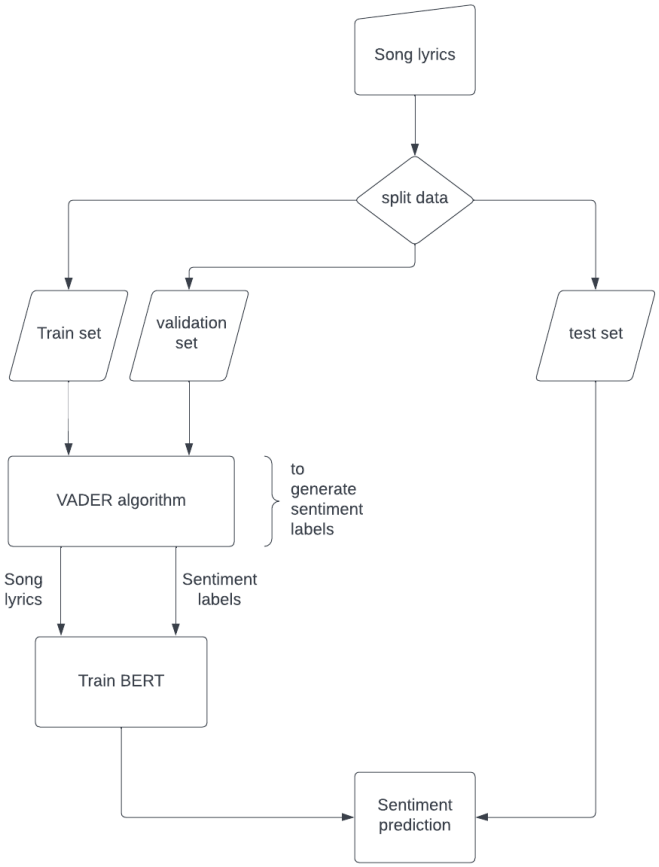

Figure 6.2: work flow diagram of VADER-BERT approach

Fine-tuning the pre-trained BERT model with Trainer function

The process starts with converting the data to a PyTorch dataset object to feed it to the BERT model. This is because the input needs to be transformed into the specific format that was used for pre-training the other core BERT models. The pre-trained BERT model that was used in this approach is the BERT large uncased model. BERT will encode the sequences and extract features for the classifier to make its decisions based on better representation.

Test set prediction

The best model from the BERT training is taken by loading the model from the checkpoint were the metrics were higher. The process is the same as while training the BERT model, where instead of train(), the predict() function is used from the Trainer object and the test set is passed into the Trainer predict() function.

6.2 Supervised methods

6.2.1 mBERT or multilingual BERT

The mBERT of multilingual BERT model is a supervised learning approache where the target labels are the manually generated sentiments. The data is split into train and test sets where the training set is used to train the model. 80% of the data is used as the train set and the rest 20% is used as the test set. The model is first initialized and is trained using the training set and that model is used for further analysis.

The following parameters were used to train the model:

- reprocess_input_data: If the value is True, the input data will be reprocessed even if there is already a cached file of the input data in the cache directory.
- use_cached_eval_features: The evaluation done during training makes use of the features that have been cached. Changing this to False will result in the features being recalculated at each step of the evaluation process.
- overwrite_output_dir: If the value is True, the trained model will be stored to the ouput dir, and any previously saved models in the same directory will be replaced with the new one.
- num_train_epochs: number of iterations the model will be trained
- silent: Disables progress bars

Once the model has been trained, it is assessed using the test set, and its accuracy is measured based on the proportion of incorrect predictions to total predictions on the test set.

$$accuracy = \left(1 - \left(\frac{wrong\ predictions}{total\ predictions}\right)\right) * 100$$

Chapter 7: EVALUATION AND RESULTS

7.1 Unsupervised methods

7.1.1 Word2Vec – K means clustering – tfidf weighting approach

Table 4 shows the results which obtained from the experiment conducted using the Word2Vec – k means clustering – tfifd weighting approach. A number of parameters were changed during the experiment, including the "sg" to check whether the model was better when using skip-gram method when compared to CBOW method.

min_count	window	sample	negative	Skip-gram (1) CBOW (0)	epochs	accuracy(%)	Precision score	Recall score	F1 score (%)
3	5	1.00E-03	20	1	30	55.7	0.69	0.36	48.1
3	5	1.00E-03	20	0	50	56.9	0.66	0.45	54.1
4	4	1.00E-04	10	1	50	56.2	0.68	0.39	50.1
3	3	1.00E-04	10	0	50	55.8	0.64	0.46	54.1
3	4	1.00E-05	10	0	50	56	0.55	0.99	71.7
4	**3**	**1.00E-05**	**20**	**1**	**100**	**58.8**	**0.66**	**0.53**	**58.8**
3	3	1.00E-05	10	1	100	57	0.65	0.52	58
3	5	1.00E-05	20	1	100	57.1	0.65	0.48	55.7
4	3	1.00E-05	20	1	150	57.4	0.66	0.49	56.1

Table 7.1:experiment results for w2v - k means clustering approach

The Table 7.1 shows that this approach is not getting an accuracy above 60%. Even when the parameters are changed drastically. It is maintaining an accuracy between 55% – 59%. Also, it can be seen that in this approach, the precision, recall and the F1 score are not on the high end.

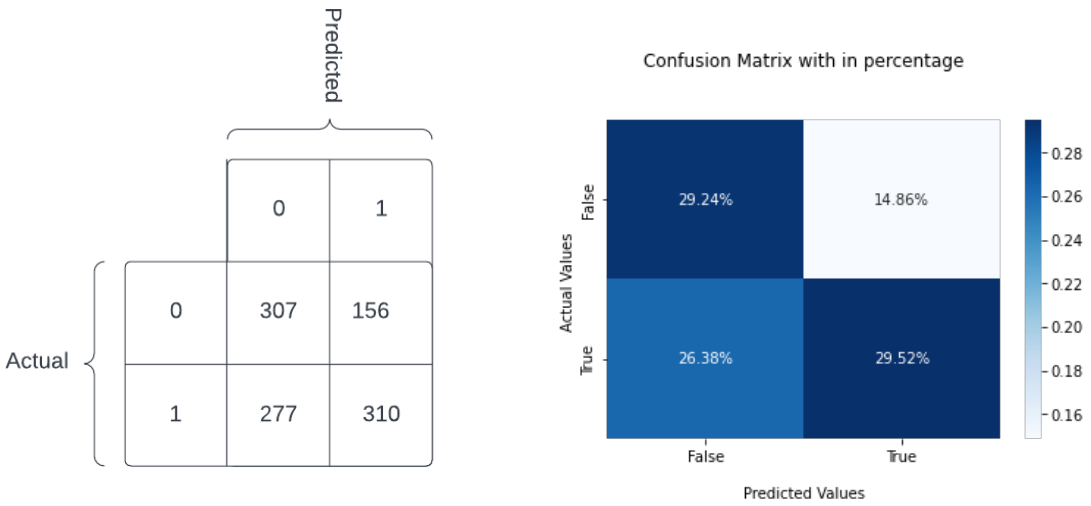

Figure 7.1: confusion matrix for w2v - k means clustering approach

The confusion matrix when the accuracy was highest is shown in Figure 7.1. It shows that the model is almost correctly predicting the negative sentiments with the model correctly predicting 307 out of 463 predictions. However, the positive sentiments are not accurately predicted with the misprediction almost equalling the predictions, which in the case of the best accuracy, the number of mispredictions is 277 while the number of correct predictions is 310. Because only one sentiment is being correctly predicted, it is not recommended to look at the precision and recall individually. Hence, we will look at the F1-score of each of the model. Hence, the best model will be the model with higher accuracy and have high F1-score. The best accuracy that was obtained from this approach is 58.8, using a negative sampling of 20 at 100 epochs, which is highlighted in yellow in Table 7.1. The F1-score can be seen to be higher than others. Therefore, we can conclude that this model is better at classifying the sentiments in the Word2Vec – clustering – tfidf weighting approach.

There is an exception case where there was higher accuracy (56%) with higher F1-score (71.7%). This is due to the high recall value (99.8%), which occurred because there was a very big imbalance during the clustering the word vectors. That is, when the word vectors were grouped into clusters, almost all the word vectors were clustered into one group than the other. This causes the very high imbalance when calculating the sentiment score and in turn the overall sentiment. This approach is not giving a very good metric score rather other supervised models specified in Chapter 2. One of the main reasons for this would be the data used. Unknown out-of-vocabulary words are one of the greatest hurdles for word2vec. If the model has never come across a certain term before, it will be completely clueless on how to understand that word or how to construct a vector for it. If this were to happen, it would mean that the model would have no choice but to make use of a random vector, which is not an ideal situation. Because the Hindi words in our dataset are unidecoded and grouped into phrases, it is quite likely that some of the words were not recognized by the model. This will cause the model to generate a random vector for that word and sometimes the vector would not adequately convey the true meaning of the word, which would in turn cause the problem when clustering the word vectors.

In addition, the fact that word2vec performs poorly on small datasets may be a significant factor. There may be a tiny number of cases in which the desired-to-be-similar words appear in similar contexts of adjacent keywords. There is no need to shift the pair to the same spot if there are no common occurrences of surrounding words; the internal Word2Vec task of predicting nearby words does not require them to be close to one another if they are each predicting separate words.

7.1.2 Hindi SentiwordNet approach

There was no model training or testing involved in this approach. The accuracy of this method was determined by comparing the sentiment predictions made using the HSWN with the sentiment predictions made by the human annotators.

The accuracy of this method was found to be around 53.6%, which is not that good of a result. The sentiment predictions in Table 7.2 shows that the predictions done using the HSWN obtained greater number of positive sentiments, almost double, whereas, the sentiment prediction by human annotators were more of balanced. One of the main reasons for these predictions going fairly positive can be because of the scope knowledge of the Hindi SentiWordnet. Using this approach would need to have a gigantic knowledge base that contain the lexicons which can provide the sentiment orientation. Even if the H-SWN used for this study is rather large, there is always a need to extend the lexicons and resolve the pertinent concerns of negation, morphological variants, etc., in order to improve the system's accuracy.

	HSWN prediction	human prediction
positive sentiment	732	587
negative sentiment	318	463

Table 7.2: sentiment count comparison

Another possible reason for the low accuracy could because there is human intelligence involved when manually scoring the sentiments. This means that some of the songs in the dataset might have more positive words than negative words, which would mean for the HSWN approach to give positive sentiment to those songs, however, that song might contain a negative vibe which the lyricist wanted to convey, which would be understood by humans but not this approach. This would cause a mismatch in the sentiments that human annotators and HSWN approach put for a song.

7.1.3 VADER – BERT base approach

As described in chapter 6, this approach used VADER sentiment analysis tool to generate the sentiment labels onto the "unlabelled" dataset for training and then used google's BERT model to train and predict the sentiments. The Table 7.3 shows the experiment results for different epochs. It can be seen from the results that the accuracy is not that good the highest accuracy being 59.8% at 15 epochs. The "eval_batch_size" parameter is the batch size per GPU/TPU core/CPU for evaluation. It can be seen from the results that the test accuracy increases when the number of epochs increases till 15, and then the accuracy starts to decrease after 15 epochs. The best model can thus be specified as the model trained at 15 epochs.

epochs	eval_batch_ size per device	Test accuracy(%) approx.	F1	Precision	recall	loss
40	32	48.4	0.5	0.39	0.68	3.35
20	64	52.6	0.6	0.54	0.67	1.46
15	**64**	**59.8**	**0.69**	**0.57**	**0.86**	**1.37**
10	64	53.9	0.62	0.59	0.64	0.85
15	32	59.5	0.69	0.59	0.83	1.08

Table 7.3: experiment results for BERT model

The best sought out model is giving a high recall but a moderate precision, which could mean that the model is only moderately good at classifying the values. F1-score is checked in this case, which in turn gives us a good score (69%) than compared to the other models.

Retrieving the best checkpoint for prediction

For the model to perform well on a given task, it must be loaded from a checkpoint that corresponds to that task. Checkpoints are snapshots of the working model during the training process. This section specifies how the particular checkpoint was loaded.

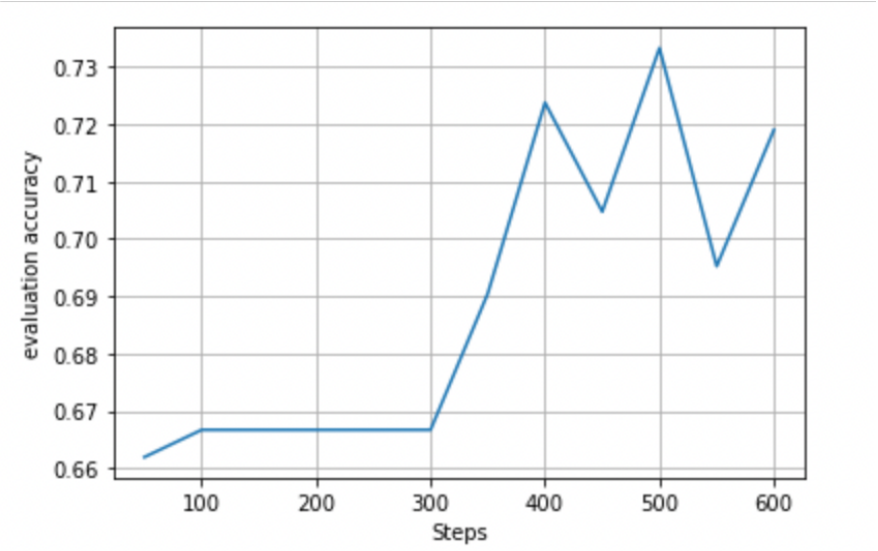

Figure 7.2: steps vs evaluation accuracy of the best model

The graph shown in figure 7.2, describes about the evaluation accuracy every 100 steps. One step equal to process one batch of data, while one epoch is completed when all the batches are processed. These steps are saved into checkpoints so that the model with the highest metric values can be loaded for predicting. From the graph, it can be seen that the evaluation accuracy maintains a steady line till 300 steps and then drastically increases to above 70% in the next 100 steps. There is a fluctuation seen after it reaches 70%, where the highest evaluation accuracy is obtained at 500 steps with around 73.3%. The model at this checkpoint is loaded to predict the sentiment.

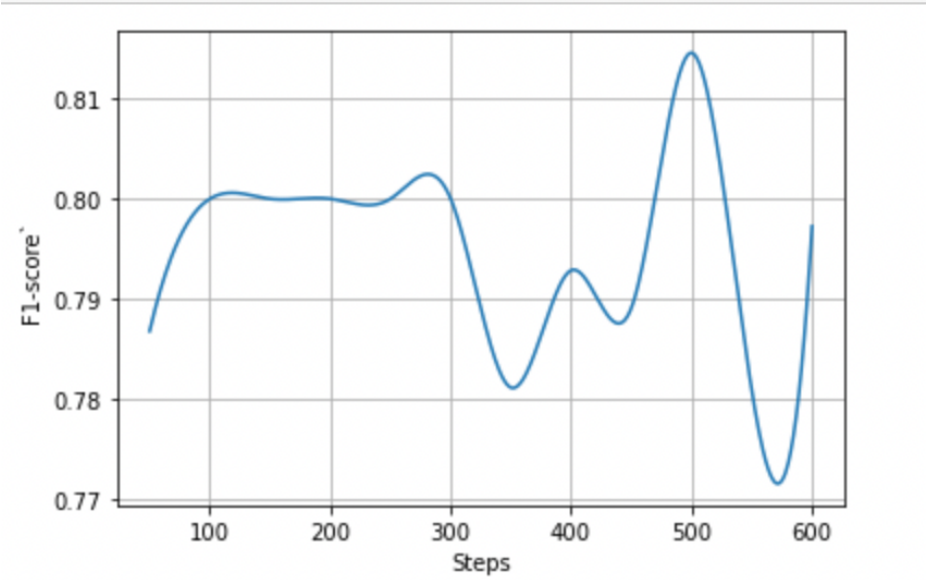

Figure 7.3: steps vs f1 score

Figure 7.3 shows the trend of F1-score after each 100 steps during the evaluation process. The f1-score remains fairly static for around 300 steps, then a steep decline can be observed. The value then increases steadily and reaches it's peak of 81.46% at 500 steps. The F1-score is usually more useful to look at than accuracy as it describes how well the model will predict the target values, which in our case, is the sentiments.

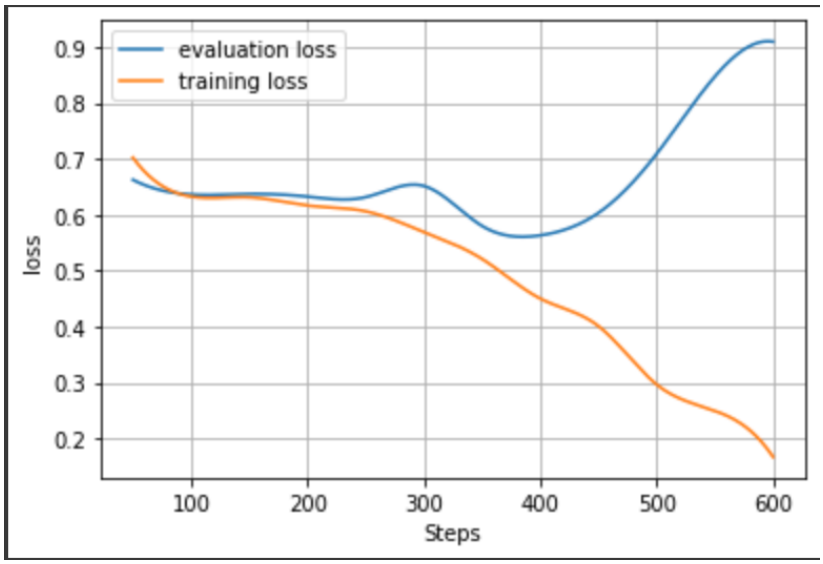

Figure 7.4: comparison between training loss and validation loss

Figure 7.4 describes about the training loss and evaluation loss per steps. When evaluating how well a model fits the training dataset, one statistic that is utilized is called the training loss. Validation loss, also known as evaluation loss, is the statistic that is utilized to evaluate how well the model performs on the validation set and is calculated after each epoch. The validation loss is comparable to the training loss in that it is determined by summing up all of the errors made in each example that is included in the validation set. It should be noted that the validation loss should always be greater than the training loss, if not, it would mean that the training dataset was not split correctly. We can see from the graph that the validation data is slightly decreasing along with the training loss however, it starts to increase while the training loss continues to decrease. This would mean the model is **overfitting**. This means that the model has become too specialized on solving for the training data and performs worse when validated on the test data. A significant reason which can be explained for this occurrence would be that the model may be too complex for the data or the model might have been trained for a long period.

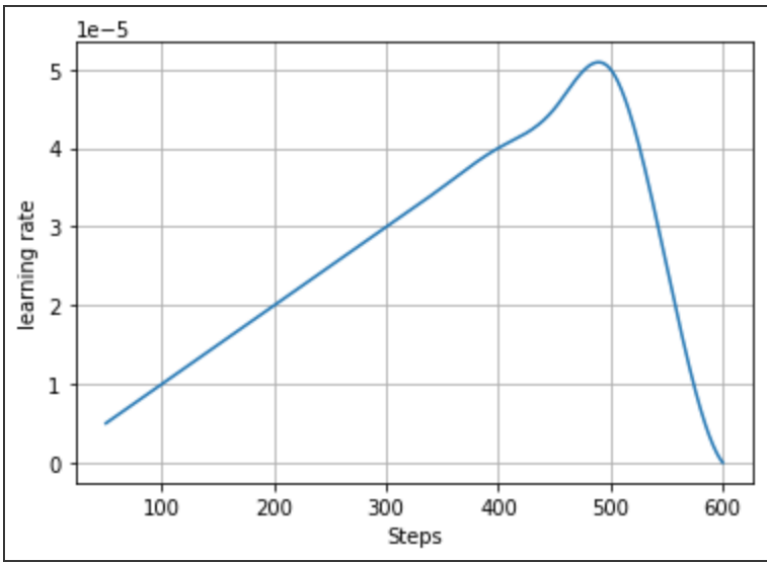

Figure 7.5: learning rate vs steps during training process

Figure 7.5 shows the trend of the learning rate of the during the training process. The learning rate is the hyperparameter that regulates how much of a change should be made in the model each time the weights are updated, in response to the estimated error. To put it another way, the learning rate determines how quickly the model has adjusted to the problem (Brownlee, 2019). The selection of the learning rate is a challenging task because a value that is too small

might result in a lengthy training process, which could get stuck, while a value that is too big could result in learning sub optimal set of weights quickly or an unstable training process. Because of these two characteristics, it might be difficult to select the optimal value.
Upon checking all the parameters, the model which was trained at checkpoint 500 is taken as the best model.

7.2 Supervised methods

7.2.1 mBERT

The multilingual BERT is the BERT model which is trained on multiple languages. The model uses supervised learning as described in Chapter 6. Table 7.4 shows the accuracy obtained using the model. The highest accuracy for this model was obtained with 4 epochs, where the accuracy is 60%.

epoch	Accuracy %
4	60
5	58.57
10	59.5
8	54.28

Table 7.4: experiment result for mBERT

7.3 Comparison between the models

When comparing the accuracies between the models used for this project, it can be seen that the supervised model mBERT is having higher accuracy (60%) than the all the unsupervised approaches. However, this can be justified as the models were trained using the manually annotated sentiments.
Out of the unsupervised methods, it can be seen that the VADER-BERT approach has a slightly better accuracy when compared to other methods, while HSWN lexicon approach obtained the least accuracy as specified in the Table 7.5. Moreover, VADER-BERT method

would also be able to classify better when compared to the Word2Vec – k means clustering – tfidf weighting method as the F1-score of the first approach is higher than latter.

Methods	Accuracy(%)	F1-score
HSWN	53.6	NA
VADER-BERT	59.8	0.69
W2V-Clustering	58.8	0.588

Table 7.5: comparison of accuracies

All the methods which were used in this project did not achieve good accuracy and performance metrics. The accuracy of the models were mostly obtained between the 50 – 60% range. The reasons for this issue and the solutions are discussed in the next chapter.

Chapter 8: CONCLUSION

Lyric based sentiment analysis of Hindi songs was done in this project. The dataset of 1050 songs was created from scratch containing the title of the song, song lyrics in Hindi language and the sentiment label which was manually annotated by human raters. Various unsupervised methods were used on the dataset, without including the sentiment labels. Methods including the lexicon based approach using Hindi SentiWordNet, transfer learning approach which is the combination of Word2Vec, K means clustering and tfidf weighting methods and a semi supervised method in VADER-BERT approach. The metrics including accuracy, precision, recall and F1-score was computed for the models, wherein it was concluded that the VADER-BERT approach was the best model out of the unsupervised methods with slightly better accuracy of 59.8% than the Word2Vec – clustering approach which had an accuracy of 58.8%.

mBERT, which is a supervised method, was also implemented to compare with the unsupervised methods and the it obtained a higher accuracy than the unsupervised approaches. This would be justified as the supervised methods used the sentiment labels manually annotated to train the model while the unsupervised methods have no such target label to train with and the model is being performance tested by comparing the predicted sentiment labels with the manually generated ones.

8.1 FUTURE WORKS

The approaches used in this project did not get a good accuracy while classification. This may be due to the lack of annotated datasets and resources for Indian languages, hence considerable focus and time to be given to creating and structuring a well organized dataset. Because the dataset was newly created from scratch, the contents of the dataset may not have been fully organized, including some of the song lyrics in the dataset starting with movie dialogs delivered just before that song and also some songs only having half the lyrics and so on. All these affect the model performance. Hence, a well-organized and structured dataset needs to be found out or created and used in these approaches to get a higher accuracy. Also, a larger dataset could be made or used which would enhance the performance of the model

because most of the algorithms used, like the word2vec and BERT would require a larger dataset to efficiently train the models.

Introducing a neutral sentiment would help in improving the performance of the models, including the Word2Vec clustering approach. Introducing the neutral sentiment as the third cluster would greatly improve in classifying the word vectors correctly to the clusters. During the pre-processing stage while creating word2vec, a spell checker could be implemented to avoid training too many embeddings of words. Proper stemming or lemmatization of the Hindi language could be done on the dataset to get more accurate word vectors.

Creation of bi-grams for the word2vec algorithm can be avoided for Hindi language, as sometimes it combines two Hindi phrases which do not have a meaning together. Also Hyperparameter tuning can be done on the word2vec algorithm based on the f1-score to achieve an increased model accuracy in the future. More techniques to build more efficient transfer learning approaches can be explored and experimented with in the future.

Also in the future, a classifier based on SentiWordNet can be combined with other approaches such as word vectors that may produce better results than each individual classifiers. Initializing more parameters with the BERT model will also go a long way in increasing the performance.

In the context of Indian languages, the majority of the languages have not been examined, with the exception of Bengali and Hindi, which have been the subjects of past research on sentiment analysis. The script, representation level, and linguistic qualities of Indian languages vary greatly. Therefore, a significant amount of effort has to be done in order to gain an understanding of the behaviour of Indian languages and to carry out the study of the same in an appropriate manner.

Bibliography

Akaishi, J. et al., 2022. Estimating the Emotional Information in Japanese Songs Using Search Engines. *Sensors 2022.*

Akhtar, Z., n.d. *BERT base vs BERT large.* [Online]
Available at: https://iq.opengenus.org/bert-base-vs-bert-large/
[Accessed 28 July 2022].

Anwaar, A., 2020. *Exploring Women 'S Reviews: A Sentiment Analysis Using K-Means Clustering.* s.l.:s.n.

Biancofiore, G. M. et al., 2022. *Aspect Based Sentiment Analysis in Music: a case study withSpotify,* Brno: SAC'22.

Boost labs, 2014. *Word Clouds & the Value of Simple Visualizations.* [Online]
Available at: https://boostlabs.com/blog/what-are-word-clouds-value-simple-visualizations/
[Accessed 25 July 2022].

Brown , R., 2021. *Sentiment Analysis: How it Works, Types & Everything You Need to Know.* [Online]
Available at: https://cogitotech.medium.com/sentiment-analysis-how-it-works-types-everything-you-need-to-know-822a1f2ddeaf

Brownlee, J., 2019. *Understand the Impact of Learning Rate on Neural Network Performance.* [Online]
Available at: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/
[Accessed 07 August 2022].

Denecke, K., 2008. *Using SentiWordNet for Multilingual Sentiment Analysis,* Hannover: IEEE.

Devlin, J., Chang, M. W., Lee, K. & Toutanova, K., 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,* s.l.: ARXIV.

Fauzi, A. M., 2019. Word2Vec model for sentiment analysis of product reviews in Indonesian language. *International Journal of Electrical and Computer Engineering,* p. 525~530.

Gong, S., 2019. *K-means Clustering for Customer Segmentations: A Practical Real-world Example.* [Online]
Available at: https://medium.com/@sygong/k-means-clustering-for-customer-segmentations-a-practical-real-world-example-196a10323b9f
[Accessed 4 August 2022].

goyal, S., 2018. *Sentiment-Analysis-On-Hindi-Reviews.* [Online]
Available at: https://github.com/shubham721/Sentiment-Analysis-On-Hindi-Reviews
[Accessed 30 May 2022].

He, H. a. J. J. a. X. Y. a. C. B. a. S. W. a. Z. L., 2008. Language Feature Mining for Music Emotion Classification via Supervised Learning from Lyrics. In: L. a. C. Z. a. Y. X. a. L. Y. Kang, ed. *Advances in Computation and Intelligence.* Berlin: Springer Berlin Heidelberg, pp. 426-435.

Hugging Face, n.d. *bert-base-multilingual-uncased.* [Online]
Available at: https://huggingface.co/bert-base-multilingual-uncased
[Accessed 15 July 2022].

Hutto, C. J. & Gilbert, E., 2022. *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text.* California, AAAI Press.

Hu, X. a. D. J. a. L. C. a. B. M. a. E. A., 2008. The 2007 MIREX audio mood classification task: Lessons learned. *ISMIR 2008 - 9th International Conference on Music Information Retrieval,* pp. 462-467.
Hu, X. & Downie, . S. J., 2010. *Improving mood classification in music digital libraries by combining lyrics and audio.* s.l., ACM, pp. 159-168.
Hu, X., Downie, S. J. & Ehmann, A., 2009. Lyric Text Mining in Music Mood Classification. *Proc. Int. Soc. Music Information Retrieval Conf.,* pp. 411-416.
India, F. P., n.d. *Language.* [Online]
Available at: https://sites.google.com/a/fargoschools.org/final-project-india/language
Joshi, A., Balamurali, A. R. & Bhattacharyya, P., 2010. *A Fall-back Strategy for Sentiment Analysis in Hindi: a Case Study.* India, Macmillan Publishers.
Kim, M. a. K. H.-C., 2011. *Lyrics-Based Emotion Classification Using Feature Selection by Partial Syntactic Analysis.* s.l., IEEE, pp. 960-964.
Kulkarni, D. S. & Rodd, S. S., 2022. Sentiment Analysis in Hindi—A Survey on the State-of-the-art Techniques. *ACM Transactions on Asian and Low-Resource Language Information Processing,* Volume 21, pp. 1-46.
Kumar, A. V. & Meera, K. N., 2022. *Sentiment Analysis Using K Means Clustering on Microblogging Data Focused on Only the Important Sentiments.* Nagpur, IEEE, pp. 1-5.
Kumar, V. & Minz, S., 2013. Mood classifiaction of lyrics using SentiWordNet. In: *2013 International Conference on Computer Communication and Informatics.* s.l.:IEEE, pp. 1-5.
Kumawat, D. C., 2020. *Introduction to Term Frequency — Inverse Document Frequency(TF-IDF) in Natural Language Processing (NLP).* [Online]
Available at: https://medium.com/@sdinesh718/introduction-to-term-frequency-inverse-document-frequency-tf-idf-in-natural-language-processing-3ec503ddbdab
[Accessed 31 July 2022].
McCormick , C., 2019. *BERT Fine-Tuning Tutorial with PyTorch.* [Online]
Available at: https://mccormickml.com/2019/07/22/BERT-fine-tuning/
[Accessed 7 August 2022].
Megret, P., 2018. *Gensim Word2Vec Tutorial.* [Online]
Available at: https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial/notebook
[Accessed 3 August 2022].
Nanda, C., Dua, M. & Nanda, G., 2018. *Sentiment Analysis of Movie Reviews in Hindi Language using Machine Learning.* India, IEE, p. 4.
Ohana, B. & Tierney, B., 2009. *Sentiment classification of reviews using SentiWordNet,* Dublin: ARROW@DIT.
Ortega, R., Fonseca, A., Guti´errez, Y. & Montoyo, A., 2013. *SSA-UO: Unsupervised Sentiment Analysis in Twitter.* Atlanta, s.n., pp. 501-507.
Pang, B. & Lee, L., 2004. *A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts,* s.l.: Proceedings of the ACL.
Pant, B., 2020. *bolly_song_lyrics.* [Online]
Available at: https://github.com/bhavyapant/bolly_song_lyrics
[Accessed May 2022].
Pastukhov, D., 2022. *Indian Music Industry Analysis: Streaming, Live Industry, Bollywood, 2022 Trends, and More.* [Online]
Available at: https://soundcharts.com/blog/india-music-market-overview

Patel, A., 2021. *K-Means Clustering.* [Online]
Available at: https://imakash3011.medium.com/k-means-clustering-ef8e9258d76a
[Accessed 31 July 2022].
Rajendran, R. V., Pillai, A. S. & Daneshfar, F., 2022. *LyBERT: Multi-class classification of lyrics using Bidirectional Encoder Representations from Transformers (BERT),* s.l.: s.n.
Roul, A., 2021. *Sentiment Analysis- Lexicon Models vs Machine Learning.* [Online]
Available at: https://medium.com/nerd-for-tech/sentiment-analysis-lexicon-models-vs-machine-learning-b6e3af8fe746
Shukla, S., Khanna, P. & Agrawal, K. K., 2017. *Review on Sentiment Analysis on Music,* Dubai, UAE: International Conference on Infocom Technologies and Unmanned Systems (ICTUS'2017).
Souza, F. D. & Filho, . J. B. d. O. e. S., 2022. *BERT for Sentiment Analysis: Pre-trained and Fine-Tuned Alternatives,* Rio de Janeiro: Cornell University.
Stephan, S. D., 2019. *Sentiment-Analysis-of-songs-by-lyrics.* [Online]
Available at: https://github.com/SebinDuke/Sentiment-Analysis-of-songs-by-lyrics
[Accessed May 2022].
Suresh, A., 2020. *What is a confusion matrix?.* [Online]
Available at: https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5
[Accessed 3 August 2022].
Suresh, A., 2020. *What is a confusion matrix?.* [Online]
Available at: https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5
[Accessed 3 August 2022].
Vatsal, 2021. *Word2Vec Explained.* [Online]
Available at: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71
[Accessed 04 August 2022].
Vinithra, S., Kumar, A. M. & KP, S., 2015. *Analysis of sentiment classification for Hindi movie reviews: A comparison of different classifiers,* Coimbatore: Research Gate.
Wikipedia, n.d. *List of languages by number of native speakers in India.* [Online]
Available at:
https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers_in_India
[Accessed 5 August 2022].
Wójcik, R., 2019. *Unsupervised Sentiment Analysis.* [Online]
Available at: https://towardsdatascience.com/unsupervised-sentiment-analysis-a38bf1906483
[Accessed June 2022].

APPENDIX

Github repository link of the project

https://github.com/sanleypeter96/sentiment-analysis