International Institute of Information Technology

SMAI PROJECT REPORT

for the course

Statistical Methods in AI -CSE471

Predict mood of the song from lyrics

Report Prepared By:
Manojit Chakraborty (2018202022)
Sayan Ghosh (2018202002)
Shubham Das (2018202004)
Siddesh Sawant (2018201031)
Group ID : 38

Contents

Chapter 1

Introduction

1.1 Project ID and Title

- **Project ID :** 44
- **Project Title :** Predict mood of the song from lyrics

1.2 Project Github Link

https://github.com/manojit32/MusicMoodPrediction

1.3 Team Members

- **Manojit Chakraborty** - M.Tech CSIS - 2018201032
- **Sayan Ghosh** - M.Tech CSIS - 2018202002
- **Shubham Das** - M.Tech CSIS - 2018202004
- **Siddhesh Sawant** - M.Tech CSE - 2018201031

1.4 Main Goal of the Project

Develop a Mood Classification System from song lyrics as well by combining a wide range of semantic and stylistic features extracted from textual lyrics.

1.5 Problem definition

The primary focus of the project is to develop a model that analyses a song by its lyrics and to detect the mood in the song. A few of such work have been done on English songs, but we are to explore it by also putting Hindi songs to it. The methodology used by far is the translation of songs in other languages to English and then analyze, because a lot of annotated datasets are available in English.

1.6 Results of the project

The project aims at developing a system to predict songs' mood and clustering them which can be used in a Recommendation System based on users' listening habit and their choice of genres. The project is also based on predicting mood of Bollywood songs by translating their lyrics to Hindi and using approaches of NLP, Polarity based Classification by combining a wide range of semantic and stylistic features extracted from the lyrics.

1.7 Team members and task assignment

- **Manojit Chakraborty [30%]**
  - Studying project related research papers and literature.
  - Creation of training data and test data.
  - Data preprocessing and feature engineering using NLP.
  - Project Report and Presentation preparation
- **Sayan Ghosh [25%]**
  - Studying project related research papers and literature.
  - Model Implementation.
  - Measuring performances of the model.
  - Improving the model by changing certain parameters.
- **Shubham Das [25%]**
  - Understanding and interpreting the implementations already out there.
  - Preparation of training and testing dataset.
  - Reporting the results and inferences.
  - Project Report and presentation Preparation.

- **Siddhesh Sawant [20%]**
  - Understanding and interpreting the implementations already out there.
  - Exploring places from where dataset to be created.
  - Model exploration and selection.
  - Project Presentation Preparation.

1.8 Project Milestones and Timelines

- **Studying project related research papers and literature.**
- **Understanding and interpreting the implementations already out there. [*27th March, 2019*]**
- **Exploring places from where dataset to be created.**
- **Data preprocessing and feature selection.[*10th April, 2019*]**
- **Model exploration and selection.**
- **Model Implementation**
- **Measuring performance of the model. [*17th April, 2019*]**
- **Improving the model by changing certain parameters.**
- **Reporting the results and inferences. [*25th April, 2019*]**
- **Project Report, Presentation Preparation.[*28th April, 2019*]**

1.9 Online Resources

- http://www.aclweb.org/anthology/W15-5939
- https://github.com/rasbt/musicmood
- https://arxiv.org/abs/1611.00138
- https://www.slideshare.net/SebastianRaschka/musicmood-20140912

- https://github.com/hyades910739/NJU$_M usicMood$

Chapter 2

Approaches Considered

2.1 Data Preprocessing

- **Tokenization** — Convert sentences to words as tokens Removing unnecessary punctuation, tags
- **Removing stop words** — Frequent words such as "the", "is", etc. that do not have specific semantic meaning and contribution to understanding the flow of text in documents or lyrics
- **Stemming** — Words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.
- **Lemmatization** — Another approach to remove inflection by determining the part of speech and utilizing detailed database of the language.

Libraries Used :

- NLTK — The Natural Language ToolKit is one of the best-known and most-used NLP libraries, useful for all sorts of tasks from tokenization, stemming, tagging, parsing, and beyond.

- BeautifulSoup — Library for extracting data from HTML and XML documents
- PyLyrics — PyLyrics is a python module to get Lyrics of songs from lyrics.wikia.com
- GoogleTrans — Google Translator to Convert Hindi to English while keeping the context of each line of lyrics

2.2 Feature Engineering

2.2.1 Bag Of Words Model (BOW)

We make the list of unique words in the text corpus called vocabulary. Then we can represent each sentence or document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary. Another representation can be count the number of times each word appears in a document.

- **Counter Vectorizer Model** : It calculates the Term Frequency of each terms in the document with the features as all the unique terms present in the document.It forms a dense matrix of shape No. of documents * No of unique terms in all the documents.
- **Term Frequency - Inverse Document Frequency (TF-IDF) Model** : TF-IDF helps to capture the significance of a term across the documents, and thus, more a term appear in other documents lesser is it's significance. Term Frequency (TF) = (Number of times term t

appears in a document)/(Number of terms in the document) Inverse Document Frequency (IDF) = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a frequent word is likely to be low. Thus having the effect of highlighting words that are distinct. We calculate TF-IDF value of a term as = TF * IDF

- **Uni-gram Model** : Here we do not keep the context of the words.It implies that the word can appear anywhere in the document has same weightage and we do not keep track of the words that appear before or after
- **N-gram Model** : n-gram model considers the features as 'n' contiguous words.Hence bi-gram would imply 2 continuous words(or pairs) and similarly for tri-gram etc.It is done by setting the n-gram range as (a,b) .

2.2.2 Word Embedding Model

One of the major disadvantages of using BOW is that it discards word order thereby ignoring the context and in turn meaning of words in the document.It is a representation of text where words that have the same meaning have a similar representation. In other words it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together.

- **Word2Vec** : Word2vec takes as its input a large corpus of text and produces a vector space with each

unique word being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec is very famous at capturing meaning and demonstrating it on tasks like calculating analogy questions of the form a is to b as c is to ?.

2.2.3 Machine Learning Models

2.2.4 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method,[2] which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

2.2.5 Multinomial Naive bayes

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.Naive Bayes is a simple technique for

constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

2.2.6 Logistic Regression

In statistics, the logistic model (or logit model) is a widely used statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value).

2.2.7 Ensemble Method - Bagging

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

2.2.8 Boosting

Boosting is a machine learning ensemble meta-algorithm for primarily reducing bias, and also variance[1] in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones.[2] Boosting is based on the question posed by Kearns and Valiant"Can a set of weak learners create a single strong learner?" A weak learner is defined to be a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

2.2.9 Support Vector Machines

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new

examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

Chapter 3

Our Implementation

3.1 Challenges

- Training Dataset Preparation (Using English Song Lyrics)
- Test Dataset Preparation (Using Hindi Song Lyrics translated to english)
- Data Preprocessing
- Feature Engineering
- Machine Learning Model Preparation
- Model Selection and Prediction

3.1.1 Training Dataset Preparation

The dataset that we start with is a 10000 Song subset of the **Million Song Dataset**. Now we do the following :

- Store the dataset into a Pandas Dataframe with features **File Name, Artist Name, Song Name**

- Using the Artist Name and Song Title, We are fetching lyrics for all the songs using **PyLyrics** package, which uses `LyricWikia.com` API to get lyrics for songs

- We are creating our model using english lyrics. So all the song lyrics containing any other language are removed from the Dataset.

- Now we will use `Last.FM` API to extract Tags for the remaining 3000 songs in our dataset. Tags can be based on Genre, Mood, Artist Type etc. For getting the song tags we request to "http://ws.audioscrobbler.com/2.0/" as endpoint with the parameters as follows :

  - method = track.getTopTags
  - api.getKeys = 0f6916aff634cb3e768baa9d5ee89341
  - artist = artists fetched from our csv file
  - tracks = tracks

In the paper **Lyric Text Mining in Music Mood Classification**, `Hu et.al`, Last.FM tags are being grouped into 18 categories according to different human moods. We have taken 10 groups from it and distributed them into our Mood Categories - **Happy, Sad, Angry, Relax.**

- **Happy Tags** : cheerful, cheer up, festive, jolly, jovial, merry, cheer, cheering, cheery, get happy, rejoice, songs that are cheerful, sunny,happy, happiness, happy songs, happy music, glad, mood: happy, upbeat, gleeful, high spirits, zest, enthusiastic, buoyancy, elation,

mood: upbeat,excitement, exciting, exhilarating, thrill, ardor, stimulating, thrilling, titillating

- **Sad Tags** : sad, sadness, unhappy, melancholic, melancholy, feeling sad, mood: sad - slightly, sad song, depressed, blue, dark, depressive, dreary, gloom, darkness, depress, depression, depressing, gloomy, anger, angry, choleric, fury, outraged, rage, angry music,grief, heartbreak, mournful, sorrow, sorry, doleful, heartache, heartbreaking, heartsick, lachrymose, mourning, plaintive, regret, sorrowful

- **Angry Tags** : anger, angry, choleric, fury, outraged, rage, angry music,aggression, aggressive, angst, anxiety, anxious, jumpy, nervous, angsty,pessimism, cynical, pessimistic, weltschmerz, cynical/sarcastic

- **Relaxed Tags** : calm, comfort, quiet, serene, mellow, chill out, calm down, calming, chillout, comforting, content, cool down, mellow music, mellow rock, peace of mind, quietness, relaxation, serenity, solace, soothe, soothing, still, tranquil, tranquility, tranquility,brooding, contemplative, meditative, reflective, broody, pensive, pondering, wistful,desire, hope, hopeful

By correlating the Tags that we found from `Last.FM` and the tag groups generated by us, we are creating the Class Labels for Moods in our Dataset. From the paper **Multimodal Music Mood Classification by Fusion of Audio and Lyrics**, `Hao et.al` , we get to know about 777 other songs, already categorized into Happy, Sad, Angry, Relaxed. We append this dataset with our previous dataset for our final training dataset.

3.1.2 Test Dataset Preparation

- We manually collected over 250 Hindi song lyrics and stored into dataframe
- Using **Google Translate API**, we auto-translated them into English lyrics and manually labelled them for performance checking.
- As our test dataset is labelled manually and training dataset is auto-labelled from the tags of Last.FM and Hao's paper, we add a fraction of translated-to-english lyrics to our training dataset to reduce the bias of the testing dataset.

3.1.3 Data Preprocessing

We have lyrics in our data. To Preprocess the lyrics column we did following preprocessing steps on both training and test datasets.

- **Tokenization** : Taking a text or set of text and breaking it up into its individual words

- **Stop-word Removal** : Stop words such as *the, a, an, in* are removed from lyrics
- **Punctuation Removal** : Removes punctuation from lyrics
- **Stemming** : Reducing inflected (or sometimes derived) words to their word stem, base or root form
- **Lemmatization** : Process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form

3.1.4 Feature Engineering

We created training, validation and testing datasets for these 3 models using natural language processing.

- CountVectorizer
- TfIdfVectorizer
- NGram Vector Model

3.1.5 Model Selection

The following classifiers are used for model preparation and testing :

- Random Forest
- Multinomial Naive Bayes
- Logistic Regression

- Ensemble - Bagging
- Ensemble - Boosting
- Support Vector Machines

Final Training Dataset : It contains 1482 english song lyrics preprocessed and feature-engineered using the previous steps

Final Validation Dataset : It contains fraction of training dataset containing english song lyrics preprocessed and feature-engineered using the previous steps

Final Testing Dataset : It contains 243 Hindi Bollywood translated-to-english song lyrics preprocessed and feature-engineered using the same previous steps.

Chapter 4

Results and Discussion

4.1 Validation Dataset Results

The validation dataset, which was created from a fraction of the training dataset of english song lyrics, saw an accuracy of atmost 53% for the TFIDF model using Logistic Regression Model as the machine learning model. Here is the confusion matrix -

4.2 Test Dataset Results

Here are the accuracy scores on our test datasets for each 3 NLP models

Classifier	CountVect	TfidfVect	NGramVect
Naive Bayes	47.2	40.5	**54.1**
Random Forest	57.2	**58.7**	52.5
Logistic Regression	**55.8**	56.2	46.3
Boosting	46	42.5	33.1
Bagging	55.3	53.5	51.7
SVM	55.7	60	55

Table 4.1: Accuracy table

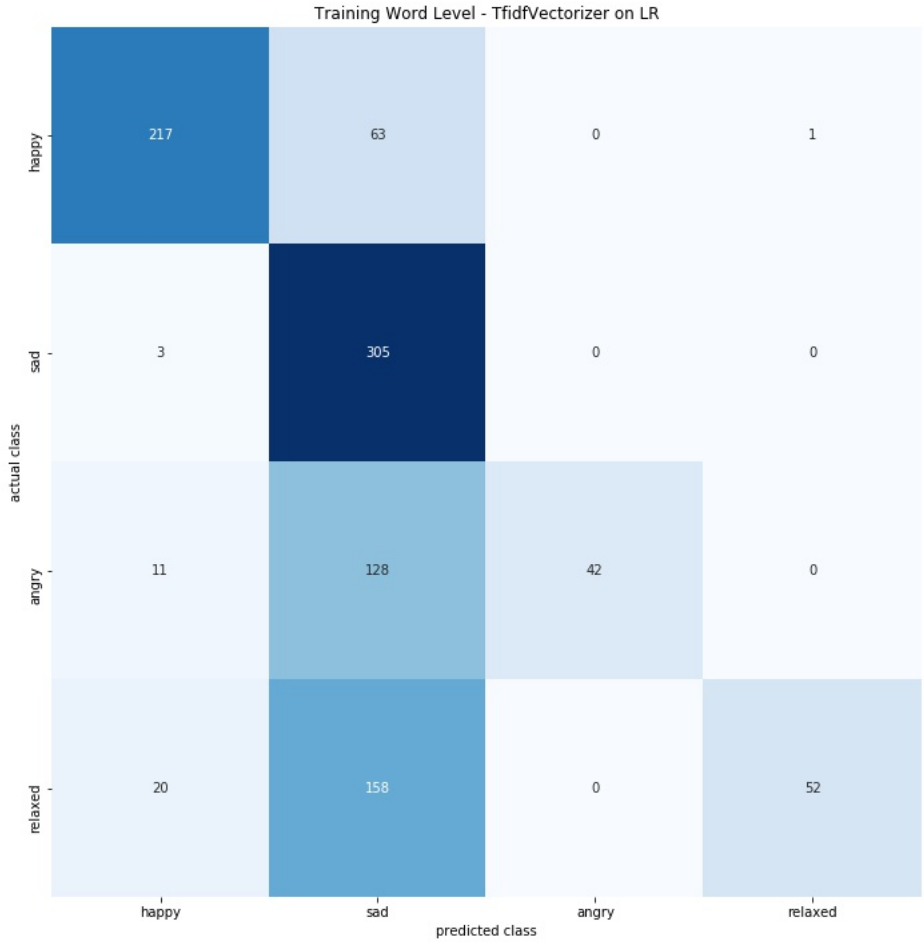

Figure 4.1: Validation Data Confusion Matrix

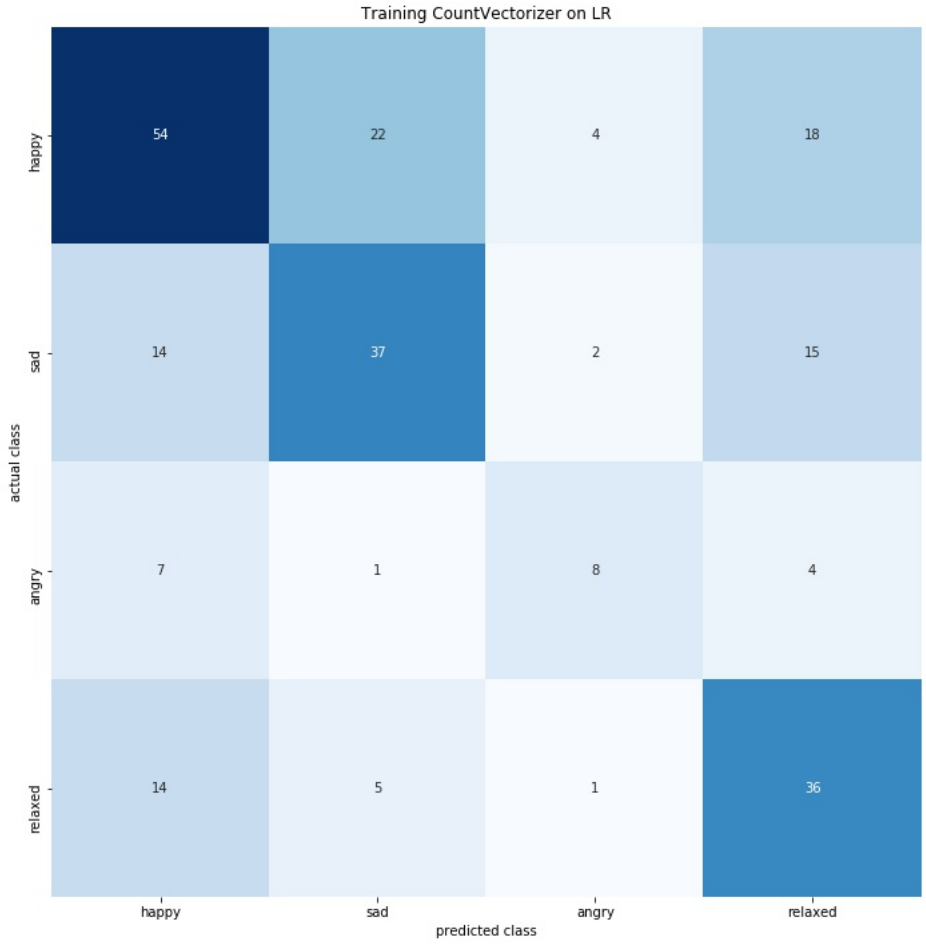

Figure 4.2: Logistic Reg on CountVectorizer

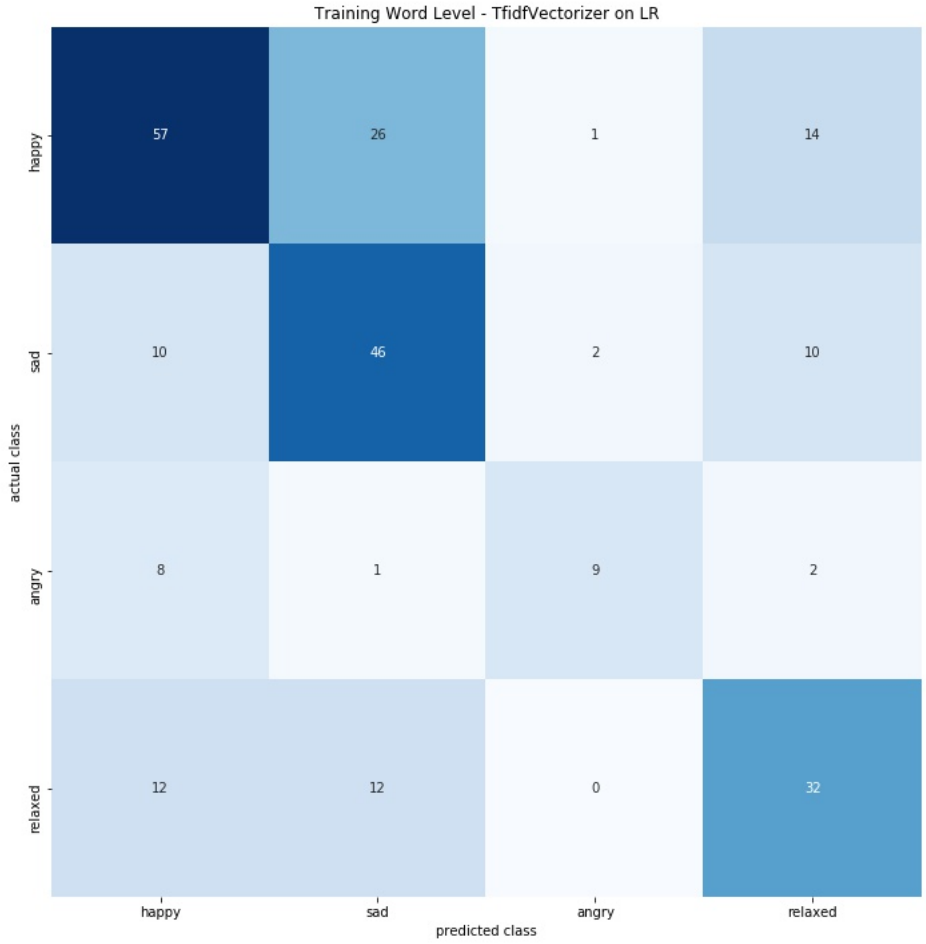

Figure 4.3: Random Forest on TfidfVectorizer

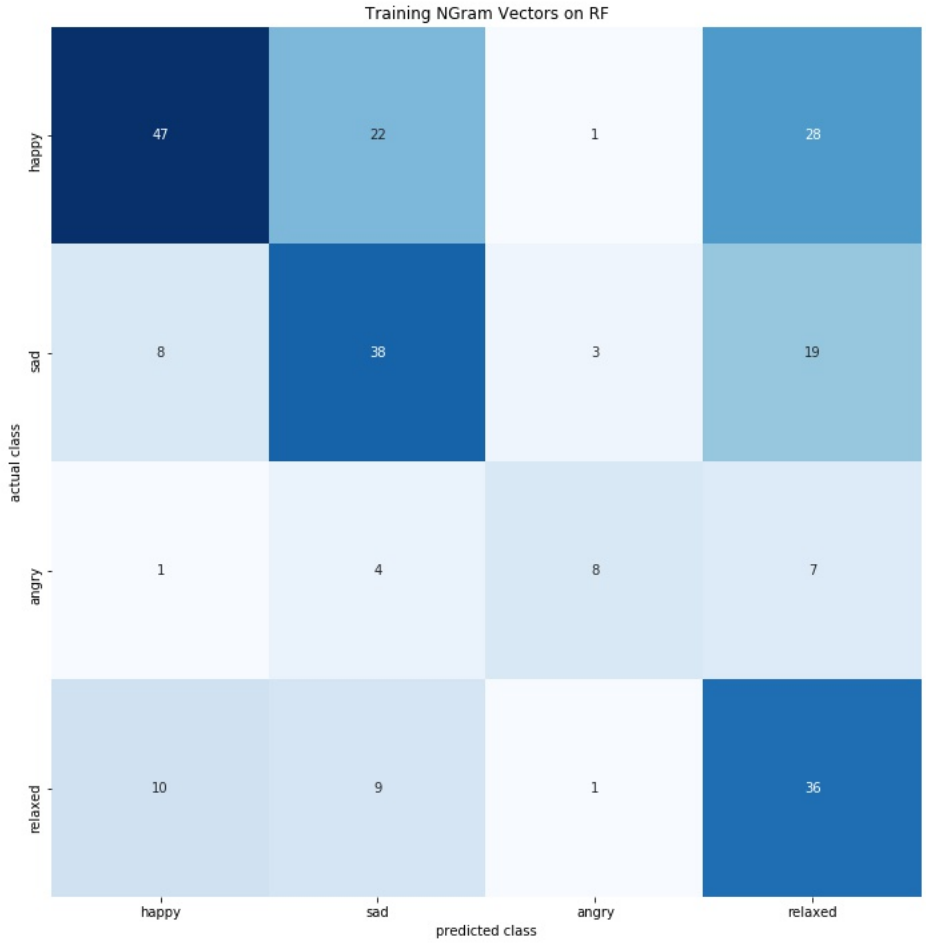

Figure 4.4: Random Forest-Bagging on NGram Vectorizer

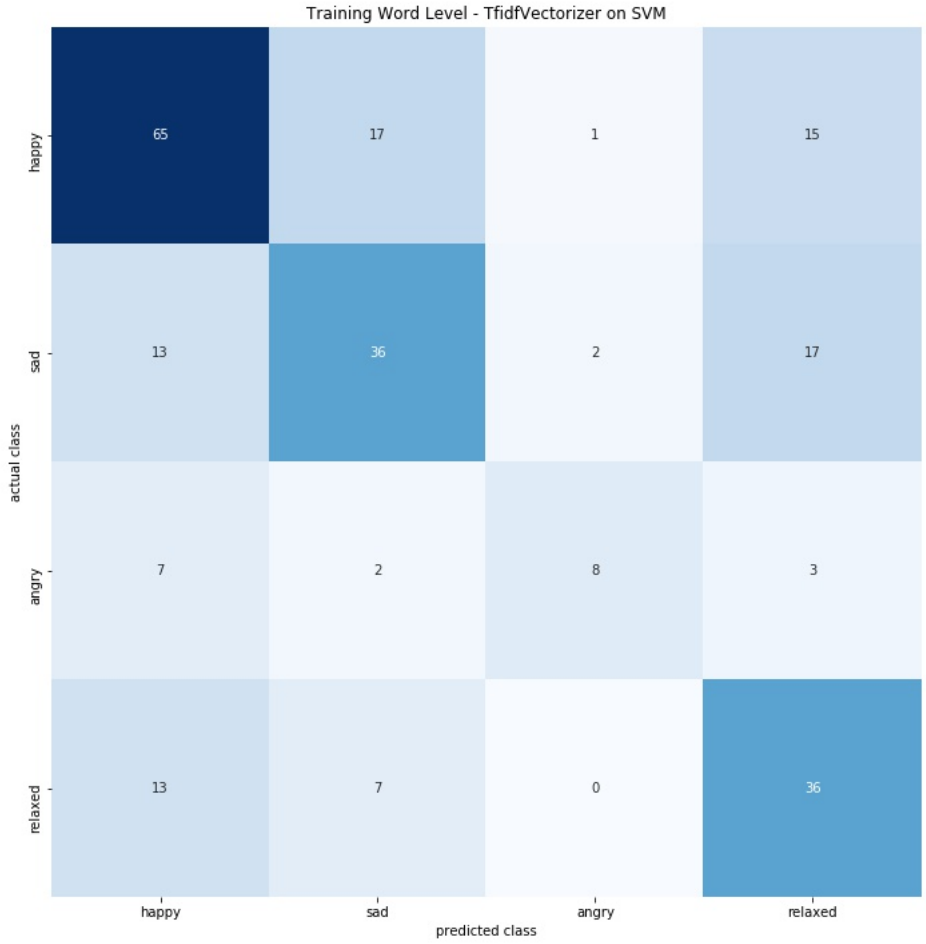

Figure 4.5: SVM on TFIDF Vectorizer

4.2.1 Best Classifier for our project

SVM using One-Vs-Rest Multilabel Classification Technique.
During Validation with English lyrics validation dataset,along with a highest accuracy of 56%, we got the following -

- **Precision** : 0.83
- **Recall** : 0.56
- **F1-Score** : 0.60

During testing with Bollywood Hindi Songs translated-to-English lyrics dataset, we got an accuracy of 60% and the following -

- **Precision** : 0.65
- **Recall** : 0.65
- **F1-Score** : 0.65

4.3 Discussion

4.3.1 Success

Many works have been done on Mood Prediction on English Songs using only two moods - Happy and Sad, where people and researchers found accuracy, precision, recall, F1-score over 60% in average. Our work is primarily the first one in which we trained our classifiers using English Song Lyrics from Million Song dataset and LyricWikia.com, and tested it against Bollywood

Song Mood Classification, all done using 4 Mood Categories - Happy, Sad, Angry, Relaxed. There is a lot of unstructured data and biases in the test dataset, yet we found an accuracy score of $\tilde{6}0\%$, with the precision, recall, F1-score more than 60%.

4.3.2 Failure

Precision, Recall Scores could be much higher if our trained classifier can properly distinguish between Happy and Relaxed moods. Moreover, there isn't any API from which we could collect hindi song lyrics at a bulk amount, from where we can generate tags automatically just like we did for english songs. There isn't many angry mood songs in bollywood, if there are some, most of them contain many sadistic words, which led our classifier to predict them as Sad. So, lack of technologies to generate nd prepare the test dataset is a big issue in this project. Also, for most of the songs, we could not use Google Translate API to translate english from Hindi, because Google blocks IP address after a certain number of requests from an end-user.

4.4 Future Directions

- We trained our model using Word2Vec along with CNN with pre-trained genism library.However, we obtained approx. 40% accuracy due to which we decided not to go along with it due to small size of training data and might need to change the parameters to obtain better results.

- Also, we can further add a Web based Search and Recommender Engine where a user can obtain his songs according to his choice of mood using our model.

- Using Selenium and BeautifulSoup4 library we can automate the action of retrieving and parsing the data and then passing it through Google translator API to the get the English lyrics for generating our test dataset.

Chapter 5

Bibliography

---

[1] Xue, H., Xue, L., Su, F. Multimodal Music Mood Classification by Fusion of Audio and Lyrics. MMM 2015, Part II, LNCS 8936, pp. 26–37, 2015.

[2] X. Hu, J. S. Downie, A.Ehmann. Lyric Text Mining in Music Mood Classification. ISMIR 2009, Oct. 2009.

[3] Sebastian Raschka. MusicMood: Predicting the mood of music from song lyrics using machine learning. arXiv:1611.00138v1 [cs.LG] 1 Nov 2016

[4] Cecilia Ovesdotter Alm. Emotions from text: machine learning for text-based emotion prediction. (HLT/EMNLP), pages 579–586, Vancouver, October 2005.

[5] Patra, Das, BandyoPadhyay. Automatic Music Mood Classification of Hindi Songs. IJCNLP 2013, pages 24–28, Nagoya, Japan, October 14, 2013.