Towards partial fulfilment for Undergraduate Degree level Program

Bachelor of Technology in Computer Engineering

A Project Preliminary Evaluation Report on:

Efficient Word2Vec Vectors for Sentiment Analysis to Improve Commercial Movie Success

Prepared by:

Admission No.	Student Name
U13CO108	Yash Rajeshkumar Parikh
U13CO058	Abhinivesh Palusa
U13CO075	Kasthuri Shravankumar

Class : B.TECH. IV (Computer Engineering) 8th Semester

Year : 2016 - 2017

Guided by : Dr. Rupa Mehta

Dr. Dipti Rana

DEPARTMENT OF COMPUTER ENGINEERING
SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY,
SURAT - 395 -007(GUJARAT, INDIA)

Student Declaration

This is to certify that the work described in this project report has been actually carried out and implemented by our project team consisting of

Sr.	*Admission No.*	*Student Name*
1	U13CO108	YASH RAJESHKUMAR PARIKH
2	U13CO075	KASTHURI SHRAVAN KUMAR
3	U13CO058	PALUSA ABHINIVESH GOUD

Neither the source code therein, nor the content of the project report have been copied or downloaded from any other source. We understand that our result grades would be revoked if later it is found to be so.

Signature of the
Students:

Sr.	*Student Name*	*Signature of the Student*
1	YASH RAJESHKUMAR PARIKH	
2	KASTHURI SHRAVAN KUMAR	
3	PALUSA ABHINIVESH GOUD	

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

CERTIFICATE

This is to certify that the project report entitled **Efficient Word2Vec vectors for Sentiment Analysis to improve commercial movie success** is prepared and presented by

Sr.	Admission No.	Student Name
1	U13CO108	YASH RAJESHKUMAR PARIKH
2	U13CO058	PALUSA ABHINIVESH GOUD
3	U13CO075	KASTHURI SHRAVAN KUMAR

of **B. Tech (Computer Engineering)** and their work is satisfactory.

GUIDE **JURY(s)** **HOD**

(i) Dr. Rupa Mehta **COED**

(ii) Dr. Dipti Rana

Abstract

IMDB releases open source data useful in various applications. In this project, this data is used to predict gross for a movie given its various attributes available to the producer just after completing the shooting of the movie. Along with this, the sentiments of the people are analyzed using the reviews they give for the movie. This is done using a modified approach applied on top of Word2Vec vectors used in sentiment analysis, which gives comparable and sometimes even better results than Doc2Vec with very less time complexity. This will in turn aid the producer to improve good marketing strategies in very less time, which will increase the profits and as a result the total gross of the movie. Also the distributors still be benefited as they will know which movies can earn them huge returns.

Table of Contents

List of Figures

List of Tables

List of Acronyms

IMDB	Internet Movie Database
CART	Classification and Regression Tree
SVM	Support Vector Machine
LDA	Linear Discriminant Analysis
LSA	Latent semantic analysis
Tf-idf	Term Frequency and Inverse Document Frequency
CBOW	Continuous Bag of Words
INR	Indian rupee
ML	Machine Learning

Chapter 1 Introduction

Movies have become a multi-billion dollar, rather a trillion dollar industry today. Profits and losses on each Friday, happen in hundreds of thousands of dollars. They might be just a source of entertainment for us, but for many, it is the source of earning a livelihood. There are many people involved in the economics of it, as pecuniary effects of it affect many. Gone are the days when people just used to see a poster and go to a movie; today, people go through the trailer and then the reviews on that. Also with a lot of movies coming in every week, producers have to make sure that their movies reach to the audience by marketing their movies in different ways.

Marketing strategies can be planned taking into consideration various factors including people reactions on the trailer and the movie poster, events, etc. Now this can be captured using reviews given by users on various social platforms including IMDB, YouTube, twitter, etc. These reviews if properly analysed can greatly aid the producers to plan their marketing strategies. Along with this, it can even help the distributors to predict how good a movie can be expected to do. This is where sentiment analysis comes into picture.

In this project, firstly gross is predicted by taking the attributes that the producers have, just after finishing the shooting of the movie, which can help them to plan the marketing strategies accordingly and then by analysing the reviews which could be done at regular intervals that could help the producers to improve their marketing strategies in order to improve their gross box-office collections. This will even help distributors to decide how many prints to buy for each movie to increase their personal profits.

Model initially uses machine leaning algorithms to predict whether the movie will earn profits and if it does, whether the profits are in sufficient quantity or not in comparison to the budget as each producer calculates economic movie success in multiples of movie-making budget. A subset of the data available as open source from IMDB and predict the relative gross on the basis of attributes like actor's and director's popularity, release year, genres, budget, etc. is used. Different classifiers like random forest classifier, gradient boosting, decision tree, etc. are tested and the best one is employed for the model.

Reviews available as open source from IMDB classified into positive and negative reviews for training and testing are analysed after the model predicts the gross profit. There are *12500* positive reviews each for testing and training, and same way for negative reviews. Also, there are *25000* unclassified reviews which can be used when required [1]. Sentiment analysis is performed using the techniques of *tf-idf* [2], *word2vec* [3] and *doc2vec* [4]. A new way is given for improving the sentiment classification accuracy for word vectors obtained by word2vec utilizing the expected rating given by Potts in 2011 [5]. 100 dimensions are used for each of the feature vectors in word2vec as well as 100 for the feature vectors in doc2vec.

The rest of the report is organized as follows. In section 2, various classifiers used for modelling gross as well as review analysis are discussed. Even sentiment analysis and the techniques already present like *tf-idf, word2vec and doc2vec* are discussed. Along with this, even expected rating as calculated by Potts [5] is discussed. In section 3, the implementation of the gross prediction is discussed followed by the implementation of sentiment analysis on the user reviews in section 4. In the same section, details of our new approach to improve the classification accuracy on word2vec vectors are given. In section 4, the results obtained by our technique are discussed and compared it with other techniques as well. Along with this, even the results obtained by testing on Pang and Lee's dataset [6] are discussed. The conclusion and possible future work are mentioned in section 5.

Chapter 2 Literature Review

In this chapter previous work done relating to the topic is discussed. Various classifiers used are discussed followed by vector space models.

2.1 Classifiers

A classifier is an algorithm used to categorize or divide the complete dataset into different classes. In our project, a new attribute called Norm_Gross is created from two existing attributes Gross and Budget and this Norm_Gross is the ratio of Gross profit to Budget. For the purpose of classifying this new attribute, classification algorithms are used. In the second phase of the project, i.e. in sentiment analysis, classifiers are used to categorize movie reviews as either positive or negative. The various classifiers used in [4], [6] are implemented in our project to compare the results. Following classifiers are used to prediction of gross and sentiment analysis.

2.1.1 Logistic Regression

In general, regression algorithms predict the value of a numeric attribute. But logistic regression, is a classifier which separates each class with a linear boundary. This linear boundary is known as linear discriminant and this discriminant might be a straight line or a plane if there are more than two dimensions. The linear discriminant is obtained using a gradient based optimization algorithm [7]. The final class of a data point is predicted based on quantification of probability of the data point and this is done using a logistic function. Hence the name, *logistic regression* to the classifier.

2.1.2 Decision Trees

Decision tree is used in classification problems. It works for both categorical and continuous input and output variables. In this algorithm, the population or sample (dataset) is split into two or more homogeneous sets (sub-populations) based on most significant splitter in input variables. Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes. The algorithm selection is also based on type of target variables. The commonly used algorithms are Gini Index, Chi-Square, Information Gain and Reduction in Variance. The algorithm used here is Gini Index. Gini index says, if two items are selected from a population at random, then they must be of same class and probability is 1 if population is pure. It works with categorical target variable "Success" and "Failure". It performs only binary splits. Higher the value of Gini higher the homogeneity. CART (Classification and Regression Tree) uses Gini method to create binary splits [8].

2.1.3 Random Forest

In Random Forest, multiple trees are grown as opposed to a single tree in CART model. To classify a new object based on attributes, each tree gives a classification and this implies the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest) [8].

2.1.4 Gradient Boosting

The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. To convert weak learner to strong learner, we'll combine the prediction of each weak learner using methods like average, weighted average, considering prediction has higher vote. To find weak rule, base learning (ML) algorithms with a different distribution is applied. Each

time base learning algorithm is applied, it generates a new weak prediction rule. This is an iterative process. After many iterations, the boosting algorithm combines these weak rules into a single strong prediction rule. Finally, it combines the outputs from weak learner and creates a strong learner which eventually improves the prediction power of the model. Boosting [8] pays higher focus on examples which are misclassified or have higher errors by preceding weak rules. The two most commonly used Boosting algorithms are Gradient Boosting (GBM) and XGboost. Gradient Boosting builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

2.1.5 Neural Networks

Neural Networks, given a set of input features and input class labels, can learn a non-linear function approximate for either regression or classification. The main difference between logistic regression and neural networks is that in the former only a linear discriminant is present between input and output layers whereas in the latter one, there can be more than one non-linear functions (layers) called hidden layers between input and output layers. The output layer receives the values from the last hidden layer and transforms them into output values. Neural Networks train the model using Backpropagation with minimizing loss function. To be more precise, it trains using some form of gradient descent and the gradients are calculated using Backpropagation [9].

2.1.6 Support Vector Machines

Support Vector Machines, given training data, each point in data belonging to one of the categories, builds a model that assigns new data to one category or the other. 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. This gap might be a linear classification or a non-linear one based on the kernel tricks used. New data or the testing

data are then mapped into that same space and predicted to belong to a class based on which side of the gap they fall.

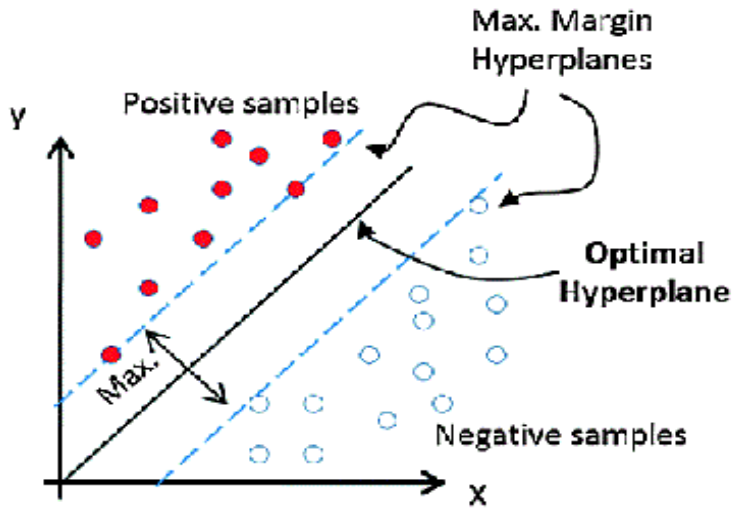

Figure 1 - Optimal Hyperplane in SVM [6]

The above figure is an example to visualize maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors [10].

2.1.7 Naïve Bayes' Classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1 \dots x_n) = \frac{P(y) * P(x1,\dots,xn \mid y)}{P(x1,\dots,xn)}$$

Bayes theorem provides a way of calculating posterior probability P (c|x) from P(c), P(x) and P (x|c). Look at the equation following:

$$P(c \mid x) = \frac{P(x \mid c)\, P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) * P(x_2 \mid c) *\ldots.. * P(x_n \mid c) * P(c)$$

- P (c|x) is the posterior probability of class (target) given predictor (attribute).
- P (c) is the prior probability of class.
- P (x|c) is the likelihood which is the probability of predictor given class.
- P (x) is the prior probability of predictor.

Multinomial Naïve Bayes' model estimates the conditional probability of a particular word (term or token) given a class as the relative frequency of term t in documents belonging to class c. Thus this variation takes into account the number of occurrences of term t in training documents from class c, including multiple occurrences [10].

2.1.8 Stochastic Gradient Descent

Stochastic Gradient Descent is an incremental Gradient Descent algorithm, which is a stochastic approximation (recursive update rules) of the gradient descent optimization method inorder to minimize cost function, or trying to find maxima or minima for each iteration, written as a sum of differential functions. In simple words, generally, computing the cost and gradient descent for the training data can be too slow on a single machine if the training data is very huge to fit in main memory. Batch optimization methods (Gradient Descent is also known as Batch Gradient Descent), don't give an easy way to incorporate new data in an online setting. Stochastic Gradient Descent addresses these two issues and one more use of this method is in neural network setting, which is accompanied by high cost of running backpropagation, Stochastic Gradient Descent can overcome this high cost [10].

2.2 - Sentiment analysis

Sentiment analysis is a field that has been worked upon a lot in the recent times. Various applications of the same have been delineated. This report focuses on one such application. With the penetration of social networks increasing to a great extent in day-to-day life, movie reviews which formerly were considered only after the movie is released, can now be utilized before the release by the producer to improve his marketing strategies. While the traditional techniques include bag of words and bag-of-n-words. Whereas the recent techniques include *LDA, LSA, word2vec and doc2vec*, along with their variations. The accuracies and running times of word2vec and doc2vec are significantly better than LDA or LSA, at least for our application. So, these are majorly focused. This report intends to simulate a model adhering to our requirements, which can aid the producer [6].

2.3 Vector Space Models

Vector space models are used to map words to vectors. In simple terms, they are used to quantify vectors and convert them into a form understandable by the computers and as a result, capture sentiment information. Here the different vector space models including tf-idf, Word2Vec and Doc2Vec algorithms are discussed that are used in sentiment analysis.

2.3.1 Term Frequency and Inverse Document Frequency (tf-idf)

Term Frequency is a basic feature vector of words (vocabulary or feature names) over the entire corpus considered. This frequency, in most cases is the raw count of a word in a document and in special cases, where the documents are longer, this frequency is divided by the maximum raw count of any word in the respective document. These counts are stored in a matrix format where the documents are considered to be rows and each unique word is considered to be feature or a column in the matrix. As a result, each unique word will be having a score or a weight with respect to each document over the corpus considered.

Document Frequency is the ratio of number of documents a word is present in to the total number of documents in the corpus. Inverse Document Frequency is the reciprocal of the above obtained ratio. Logarithmic function is applied to the final obtained ratio and this is multiplied to the term frequency score. The importance of Inverse Document Frequency is that it filters out the most commonly occurring words in the documents as these words have no significant role as they are occurring in almost all the documents [2].

2.3.2 Word2Vec

This algorithm brings in the notion of context of a word and learns the vector representations of these words from the context whereas the *tf-idf* model is a kind of collection of words (bag of words) using only frequencies to obtain vector representations of words. The unique words of the corpus are initially represented using one-hot encoded vectors. For example, if there are *V* unique words in the vocabulary of the entire corpus containing text files or documents, then each word is represented by a vector of size *V* using only one 1 and the remaining V-1 elements of the vector to be zeroes. $[0\ 0\ 0\ 1\ 0\ 0\ 0\ \ldots\ 0\ 0\ 0\ 0]_{(1xV)}$ is an example of one-hot representation of a word containing *V* unique words. These one-hot vectors are given as input to a neural network in order to learn the context of the words from the corpus available and this learning is done using one the two methods – Continuous Bag of Words model (CBOW) and Skip-Gram model [3].

The size of the hidden layer in the neural network decides the dimension size of each word embedding or word vector representation. For example, if the size of the hidden layer is *N*, then the number of dimensions or the vector size of each word would reduce to *N*. This hidden layer, in the *CBOW* model, learns the vector representations of the word in such a way that, given a context, a word will be predicted whereas in *Skip-Gram* model, given a word, the context of that word will be predicted.

For example, consider a sentence, ‘The past has no power over the present moment ‘. In this sentence ([the, has], past), ([past, no], has), ([has, power], no) … are the examples of (context, target) pairs of window size 1 in *CBOW* model. In the same sentence if the target and set of

context pairs is reversed, then (past, the), (past, has), (has, past), (has, no), (no, has), (no, power) … are the examples of (word, context) pairs or (input, output) pairs in the *Skip-Gram* model.

2.3.3 Doc2Vec

Doc2Vec overcomes the problem faced by bag of words and bag-of-n-words models by capturing the context along with the word order. Instead of modelling words as vectors, entire sentences and even paragraphs as a whole are represented as vectors which in turn make it a distributed memory model. But one limitation of Doc2Vec is that it requires more memory than Word2Vec. Also, the corpus with which the model is trained makes a big difference. But the highest accuracies are obtained using Doc2Vec, sometimes even giving a relative improvement of 30% than the conventional bag-of-words model [4].

2.4 Expected Rating

Potts in 2011 gave quantitative weights to the words based on the corpora available from IMDB and the Experience Project. This was basically intended to know how negative some words can be when taken into consideration the entire context. The expected rating given to the words in the end can be used even to distinguish the positive and negative words as the negative ones are given negative weights. This expected rating given by him can be used to improve the accuracy of traditional sentiment analysis algorithms [5].

Chapter 3 Model preparation for predicting gross profit

This section focuses on the preparation of the model before the release of the movie. This aids the producer to predict the movie business in terms of gross amount just after producer has completed the making of the movie based on the attributes available with him. This helps producer to design proper marketing and promotion strategies that will help improve the business of the model which producer can judge time to time from the incremental model discussed later.

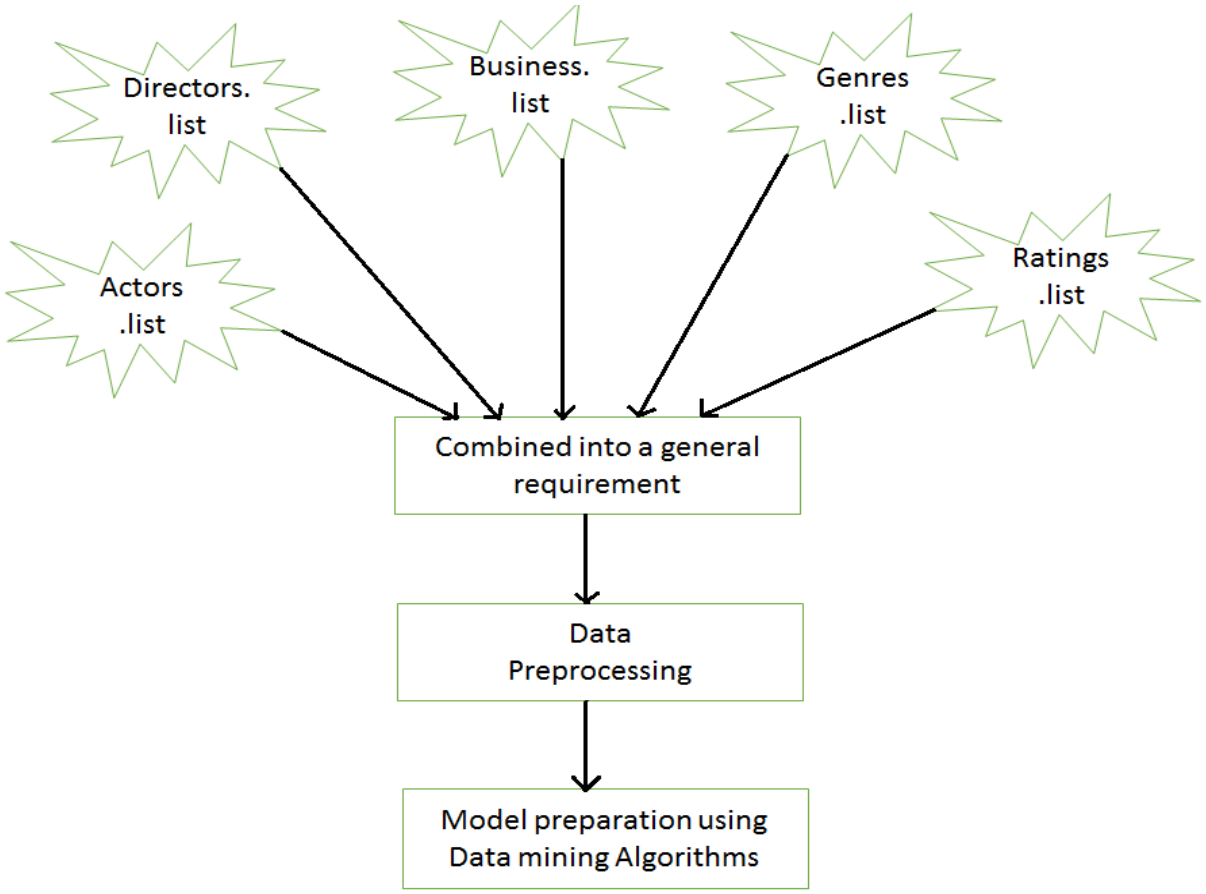

Figure 2 – "Gross profit prediction" model workflow

3.1 Data Collection

The data used is collected primarily from the open source repositories pointed out by imdb.com. All the files are available as X.list files which are pre-processed and converted in the required form. This data is a small subset of the entire data which is made open-source for educational purposes. For this particular iteration, a subset (around 5000 instances) is obtained of the big data that is split up in various X.list files. The data uses the following attributes for model preparation:

- Movie_Name
- Actor_1_Name
- Actor_1_fb_likes
- Actor_2_Name
- Actor_2_fb_likes
- Actor_3_Name
- Actor_3_fb_likes
- Director_Name
- Director_fb_likes
- Producer_Name
- Genre
- Budget
- Release_Year
- Movie_fb_likes
- **Gross***

* Gross attribute is utilized as a training element for model.

All the above data is converted in the form of the mentioned attributes in a .csv file.

3.2 Data Preprocessing

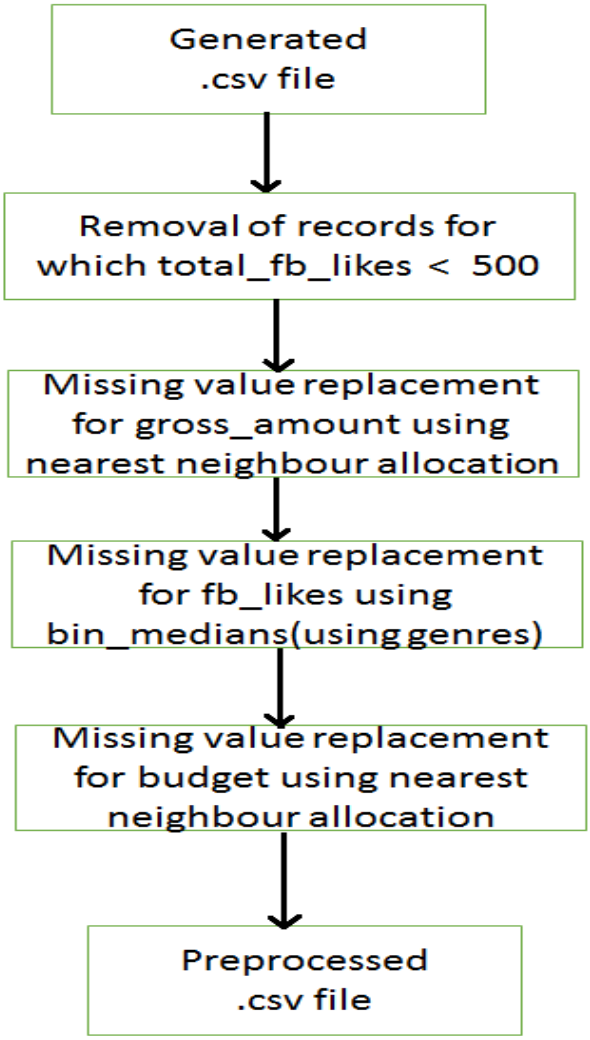

Figure 3 – Preprocessing Workflow

Figure 3 is the preprocessing workflow including all the stages that is iterated to obtain the final preprocessed dataset for further application of machine learning algorithms.

Two more attributes from the existing attributes to aid us in the model preparation.

- Actors_fb_likes = Actor_1_fb_likes + Actor_2_fb_likes + Actor_3_fb_likes
- Tot_fb_likes = Actors_fb_likes + **0.9***Director_fb_likes

Here, after trying various multiplication factors, a multiplying factor of 0.9 is used for the director_fb_likes which gives the best results. This is based on the fact that the impact of the director's popularity on the movie is just 90% as compared to that of actors' popularity who are involved in the movie.

This derived attributes helps in the missing value replacement as well as in elimination of the potential outliers. As the Facebook likes are approximately in lakhs for most actors, directors and movies; the cases need to be handled where they are null or labelled as zero. Now, Tot_fb_likes attribute is calculated based on the formula provided above and eliminate all those tuples where this count falls following 500. This will help us to eliminate the outliers.

Next, Gross attribute is considered. There are tuples where that attribute is null. So, here, nearest neighbour allocation technique is employed for missing value replacement of the attribute. kNN can be employed but based on the normal approach considering only fb_likes values of the actors, near same accuracy is obtained which reduces the time complexity to a great deal. So the same is employed for replacing the missing values for the Gross attribute. Here median is used in place of mean or any similar algebraic measures because they would have a big impact due to the extreme scenarios i.e. due to the extremely high grossing movies or the movies which have incurred huge losses and have very small amount as gross. The algorithm is as following:

(i) If for given tuple gross=0 or gross=NULL then
   a. Look for genre of the movie and obtain all the movies in the dataset corresponding to that genre
   b. Sort the obtained tuples according to the Actors_fb_likes
   c. Assign the median of all the Gross attributes of these tuples

Next, Director_fb_likes having null values are considered. Bin-medians [8] can be used for replacing the null values. The reason that bin medians are used in place of bin-means is that

mean would be hugely impacted by the extreme values, i.e. the tremendously popular directors. The algorithm for the same is as following:

(i) If for the given tuple director_fb_likes=0 or director_fb_likes=NULL
   a. Obtain all the tuples with the same genre of the current movie tuple. If the number of such tuples>0 and at least one tuple don't have director_fb_likes missing then
      i. Sort the tuples according to director_fb_likes
      ii. Prepare bins of appropriate size
      iii. Replace the missing value with the appropriate bin-median
   b. Else
      i. Obtain the tuples with Actors_fb_likes within the range of 10,00,000 from the current tuple's Actor_fb_likes
      ii. Prepare bins of appropriate size
      iii. Replace the missing value with appropriate bin-median

Similarly, bin-medians [8] is used for the preprocessing of the Movie_fb_likes attribute. The algorithm for the same is given following:

(i) If for the given tuple Movie_fb_likes=0 or Movie_fb_likes=NULL
   a. Obtain all the tuples with the same genre of the current movie tuple. If the number of such tuples>0 and at least one tuple don't have director_fb_likes missing then
      i. Sort the tuples according to Movie_fb_likes
      ii. Prepare bins of appropriate size
      iii. Replace the missing value with the appropriate bin-median
   b. Else
      i. Obtain the tuples with Actors_fb_likes within the range of 10,00,000 from the current tuple's Actor_fb_likes
      ii. Prepare bins of appropriate size
      iii. Replace the missing value with appropriate bin-median

Next, budget attribute with missing values are considered. This case is much similar to the case where gross attribute was missing and can be handled in a similar way employing modified version of missing value replacement using nearest neighbour value allocation. Here also

median is employed in place of mean for reasons similar to the gross attribute's case. The algorithm for the same is given following:

(i) If for given tuple budget=0 or budget=NULL then
   a. Search for genre of the movie and obtain all the movies in the dataset corresponding to that genre
   b. Sort the obtained tuples according to the Actors_fb_likes
   c. Assign the median of all the budget attributes of these tuples

Also, there are some missing values for the Release_year as well. This case is handled by replacing all the missing values with a global constant of 2010. As based on all the attributes, film's year cannot be judged and even if it is tried to obtain, the time complexity increases a lot giving away so much time for an attribute that won't have such a huge impact on the model is not efficient.

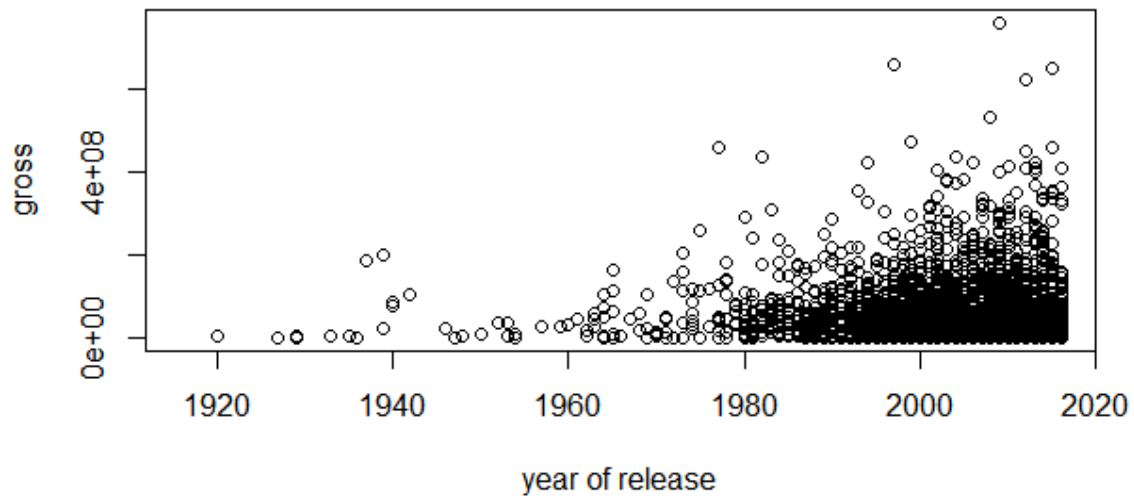

Figure 4 - Plot of gross -> year of release

As evident from the plot, most of the entries occur between 2000 and 2020. That is the reason the global constant for missing value replacement of year of release attribute is chosen to be 2010.

3.2.1 Target Attribute

Every producer wishes to predict the profit incurs and as a result take measures to increase it. Now, the term profit is subjective for different producers. One having a very small budget film may be satisfied by a profit of 10 lakh INR. But, producers of big banner movies may not be even contented with 10 crore INR. But, one thing that the distributers as well as producers have their eye on is the ratio of Gross attribute to that of Budget as everyone wishes to recover what they spent and earn profit in its multiples.

Thus, using the above logic, a new attribute "Norm_Gross" is created that would act as our target attribute to be predicted, i.e. the ratio of "Gross" attribute to that of "Budget" attribute. This would as a result be an attribute with continuous distribution. But, it can be modified to be distributed in different classes to estimate commercial success at the very beginning.

The following logic is taken into consideration to split the "Norm_Gross" attribute's range in two ways, which can be inferred from the distribution of it in the dataset as shown in the figure:

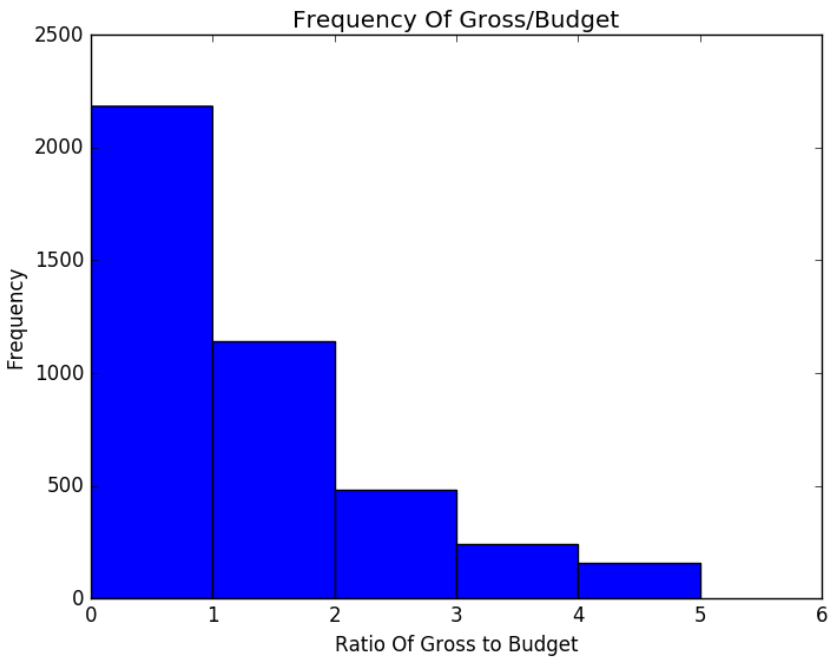

Figure 5 – Plot of Frequency vs Norm_Gross (continuous)

(1) Considering 4 classes

- Class 0 : Range [0,0.5)
- Class 1 : Range [0.5,1)
- Class 2 : Range [1,5)
- Class 3 : Range [5,∞)

(2) Considering 3 classes

Later after calculating (visualizing) the frequencies of each class, we even take the case for considering three classes:

- Class 0 : Range [0,1)
- Class 1 : Range [1,5)
- Class 2 : Range [1,∞)

3.2.2 Outliers

Outliers tend to distort the model by overfitting. Now, in Stanford dataset there are movies which have earned tremendous successes. But such unmatched commercial success cannot be predicted beforehand. Thus, such records need to be removed to improve the accuracy of the model used.

If the value of Norm_Gross is greater than 5, this value corresponds to an extraordinary or overwhelming success which cannot be accounted for previously. This may happen due to various dynamic reasons that cannot be predicted beforehand. Hence, these are outliers and are to be removed before applying modelling to the dataset. Thus, class 4 can be omitted from way number 1 of classifying and class 3 from way number 2 of classification. But just to ensure that the removal of outliers yields better results, the accuracy of a model is tested on it. The accuracy improved by 4% by removing the outliers.

3.2.3 Genre Splitting

According to Stanford dataset, there are 25 total genres which includes genres like Action, Adventure, Crime, Drama, Musical, Thriller, etc. Now, as these values are in string format, they cannot be incorporated in the model directly. Also, a movie may have multiple genres at the same time. Thus, a way has to be figured out to incorporate genres as one of the attributes at the same time converting it into a numeric attribute.

A proposal is made to achieve this by splitting the “Genres” attribute into 25 different attributes with the name of the attributes as the genre-name itself. It would be Boolean indicating whether the given movie belongs to that genre or not. But there is a problem associated with this approach. The number of attributes increases a lot i.e. 25 new attributes and the resulting matrix is a sparse matrix. Also, there are some related genres like “Musical” and “Music” which are combined into one common attribute “Musical”.

First, genres are not used at all as an input to the Decision tree model. The accuracy is recorded to be 51.5%. Now, all the genres are split according to the above logic and obtain an accuracy of 55.6%. Now, as the number of attributes have significantly increased along with the processing time required, the number of attributes must be optimized. It is found that in the list of genres, there are genres like “Film-Noir”, “Documentary”, “War”, etc. which either have only some records corresponding to them or have very little impact on the commercial movie performance as the audience doesn’t look out for these genres in any movie in common. Thus, an optimized list of genres is obtained which would affect the movie’s commercial success and even reduce the processing required for the model along with improving the accuracy.

Different combinations of genres are taken and found that the best accuracy along with the least processing power was for a list of 8 genres which comprised of "Action", "Animation", "Comedy", "Crime", "Horror", "Romance", "Sci-Fi", and "Thriller". This yielded us an accuracy of 58% when tested with decision tree model. Thus, required optimized list of genres is obtained that will be tested against various models.

3.3 Modelling

The first algorithm that was implemented is Logistic Regression using the 'One vs Rest' method with all the attributes mentioned previously, followed by Gaussian Naïve Bayes' Algorithm, Decision trees, Random Forest classifier, Gradient Boosting, Artificial Neural Networks, and Support Vector Machine. The accuracies obtained are shown in the table following.

Table 1 – Gross prediction accuracies by various classifiers

	Three Classes [0-0.5, 0.5-1, 1+]	**Two Classes [0 – 1, 1 +]**
Decision Tree	60.62%	76.37%
Random Forest	**67.72%**	**78.74%**
Gradient Boosting	66.14%	77.95%
Gaussian Naïve Bayes	50.39%	62.99%
Logistic Regression	56.69%	59.05%
Support Vector Machine (kernel = 'linear')	53.54%	62.20%

As evident from the table, random forest classifier gives the highest accuracy, both for 3 classes as well as 2 classes. Also, it is easy to implement random forest classifier in distributed systems environment which will be useful when data is very large to analyze. Thus, the model obtained by it is considered along with its accuracy. But other classifiers may yield better result for other dataset.

Chapter 4 Review Model

Now, the sentiment analysis part of our technique is focused that will aid the producers to improve their marketing strategies catering to the user requirements. Also, it helps the distributors to decide which movie-prints in what quantity can yield them higher profits. The workflow block diagram is as shown below:

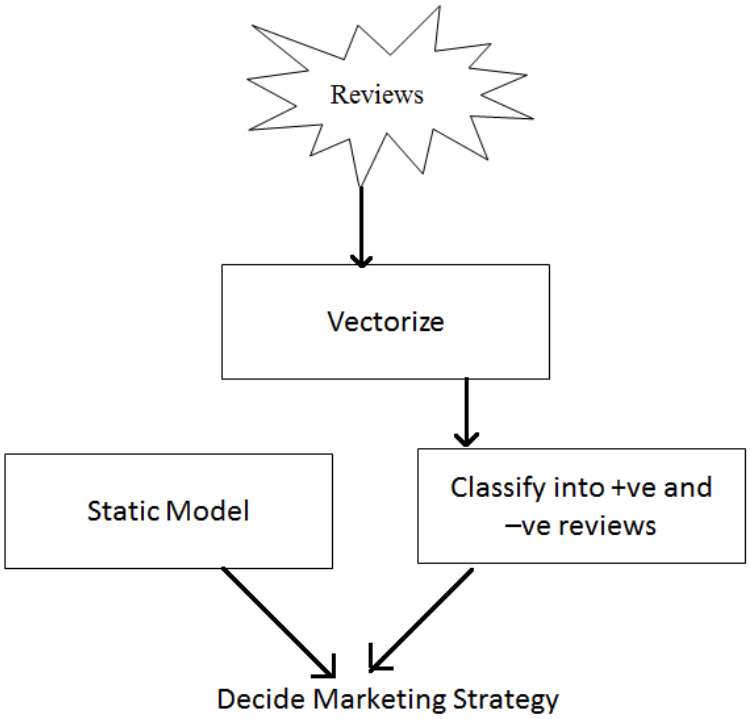

Figure 6 – Review Model preparation block diagram

4.1 Data Collection and Preparation

The review corpus is taken from Large Movie Review Dataset, available from Stanford.edu [1]. It contains 25000 labelled polar reviews for training and 25000 for testing. Along with this, there is also unlabeled data available for use. This data is already preprocessed removing extra spaces and other delimiters. Emoticons are not considered in the model for now but they may be included in future work.

4.2 Vectorization

Here, various techniques used for vectorization, including tf-idf, word2vec and doc2vec are discussed.

4.2.1 tf-idf

In our movie review dataset, as there are *25000* review documents for training, *25000* rows are obtained in the co-occurrence matrix and the unique words in the training corpus will act as features. After removing most occurring and least occurring words (features) from the dataset, around *18000* unique words are left. So, the dimensional size would be approximately *18000* for each row (document). For example, most frequently occurring words like *a, the, …* occur in almost all the documents and this would imply that these word's document frequency would be equal to one. After applying logarithmic function over the obtained inverse document frequency, the score or weight of these words would be resulting in zero thereby significantly reducing the term frequency score and as these words are not obtaining their threshold values of *tf-idf* scores, these words are being eliminated.

4.2.2 Word2vec with Mean Embedding Vectorizer

In the previously mentioned vector space model, the dimensional size obtained is as high as *18000.* So, the dimensional size is to be significantly reduced.

- *Word2vec* model overcomes this as the number of dimensions is dependent on the size of the hidden layer in the neural network. For Stanford dataset, a value of *100* is defined to be the dimensional size of each word. So, each word would be a vector of size *1*x*100*. This implies that the dimensionality is greatly reduced when compared to *tf-idf*.
- Similar to the above vector space model, each document is represented as a row which would result in *25000* rows. But, in a document, mean of all word embedding are calculated which results in *100* columns or dimensions for each row or document. So, in effect training a word2vec vector space model takes much less time when compared to *tf-idf* vector space model.

4.2.3 Doc2Vec

Now, Stanford dataset is tested with Doc2Vec model. Then different classifiers are applied to the trained paragraph vectors.

- The paragraph vectors are trained using 100 dimensions of the feature vectors.
- Different variants of Doc2Vec are used i.e. the distributed bag-of-words (PV-DBOW), averaging of the word vectors and concatenation of the word vectors. Doc2Vec is implemented using the scikit-learn library in Python.
- The vocabulary is built first using the preprocessed files of positive and negative reviews in Stanford dataset. The built paragraph vectors can be stored for further use.
- The vectors are built using all training data, testing data and unsupervised data as well.
- Now these vectors are input to various classifiers and the accuracies obtained are shown in the table.

4.3 Proposed approach

Traditional model of tf and tf-idf don't capture information about the context. Thus they are in essence incapable of predicting any novice review even if it uses all the words from the trained model. This becomes even difficult when the review is written in a sarcastic tone. Also, word2vec algorithm can predict words pertaining to the particular topic. But for the sentiment classification, words in the same topic may be adhering to different emotions, i.e. both positive and negative. Also word2vec algorithm models vectors of words "good" and "great" nearby each other. But they have different intensities. This can be captured by incorporating the expected rating by Potts [5].

The classification accuracy can be improved if the word vectors can in some way embed these word weights in them, along with the negative words away from the positive ones. Now this is taken care of by Potts as negative words have negative sign i.e. they are represented as negative integers. This can be embedded by multiplying the word vectors with their corresponding weights. Also, the mod weights are included that will just capture how important the word is in the context. First, this technique is applied to tf-idf vectors and then to word2vec vectors.

This is implemented for both the variants of word2vec i.e. continuous bag of words and skip-n-gram technique. The technique is delineated out in steps as follows:

- Obtain word vectors using Word2Vec and store them.
- Multiply each of the word vectors with the corresponding weights as obtained by Potts [5]. This multiplication is done in two ways:
    - First, just multiply the mod weights.
    - Second, multiply weights along with their polarity signs.
- Apply classification algorithms for sentiment classification using the modified vectors.

The accuracies obtained by using this technique shows great results with word2vec, giving accuracies comparable to and sometimes even better than Doc2Vec for Stanford dataset. Now word2vec takes less amount of space to store the vectors and the time taken by classifiers is also considerably low. The running times are shown in the table 3 for reference. Also, the accuracies obtained by different classifiers are shown in the table 2, for both the variants of word2vec and even for tf-idf. Results obtained before and after the inclusion of expected rating are included for comparison. The accuracies have shown considerable improvement as shown.

Table 2 – Accuracies for review analysis by various classifiers on different vectorization techniques

	tf	**tf-idf**	**w2v (cbow)**	**w2v-mod (cbow)**	**w2v-pol (cbow)**	**w2v (sg)**	**w2v-mod (sg)**	**w2v-pol (sg)**	**d2v**	**d2v (bow)**
RFC	78.05%	77.62%	74.18%	83.08%	84.98%	78.05%	84.75%	86.33%	73.34%	71.81%
DT	72.27%	70.94%	67.09%	77.1%	78.78%	71.16%	79.02%	80.78%	66.22%	63.76%
GBC	72.43%	72.24%	72.80%	81.97%	84.38%	76.48%	83.7%	86.60%	72.74%	69.90%
LR	85.39%	88.09%	85.29%	88.09%	88.55%	86.43%	83.53%	87.48%	86.24%	88.62%
SGD	83.73%	87.8%	80.18%	87.98%	88.58%	85.94%	86.91%	87.34%	79.44%	86.46%
MNB	82.94%	84.02%	_	_	_	_	_	_	_	_
LSVM	82.80%	86.16%	85.34%	88.1%	88.68%	87.28%	88.74%	***89.01%***	86.22%	88.708%

Table 3 – Running times comparison of various vectorization techniques for sentimentpolarity determination

	tf	tf-idf	W2v (cbow)	w2v-mod (cbow)	w2v-pol (cbow)	w2v(sg)	w2v-mod(sg)	w2v-pol(sg)	d2v	D2v (cbow)
Vector -ization time	6.423s	9.957s	171.92s	171.92 + 90s	171.92 + 90s	891.71s	891.71s + 63s	891.71s + 63s	1837.6s	5224.2s
RFC	7.857s	10.237s	3.583s	3.921s	2.899s	3.827s	2.813s	3.415s	4.377s	5.896s
DT	33.72s	49.02s	4.780s	5.784s	6.324s	5.724s	5.421s	6.454s	6.606s	7.675s
GBC	0.979s	3.126s	2.212s	1.599s	1.578s	1.474s	1.425s	1.595s	2.263s	2.368s
LR	5.886s	2.078s	1.627s	0.849s	1.170s	0.834s	0.629s	0.459s	1.101s	2.356s
SGD	1.665s	0.142s	0.143s	0.118s	0.086s	0.090s	0.136s	0.105s	0.082s	0.190s
MNB	0.307s	0.108s	-	-	-	-	-	-	-	-
LSVM	5.160s	0.628s	-	6.087s	4.514s	1.040s	0.682s	0.461s	16.330s	13.157s

In the above comparison table, it can be observed that training the vectors of words obtained using *tf* and *tf-idf* take significantly large time to fit in the classifiers when compared to both *word2vec* and *doc2vec*. This is because the dimensions or the features of each row of the vectors obtained using the first two vector space models are approximately *18000* whereas the features obtained using *word2vec* and *doc2vec* are 100. This greatly reduces the fitting time into classifiers for both *word2vec* and *doc2vec*. If these two vector space models are considered, then the time taken to generate vectors for *doc2vec* is significantly greater compared to that of *word2vec* and *word2vec* with *polarity* models.

Also Doc2Vec vector representations need paragraph vectors along with the word vectors where as word2vec vectors are just word vectors. Thus, the space complexity is definitely reduced by using word2vec vectors in place of Doc2vec vectors. Also, for predicting a new review's analysis, Doc2vec needs to generate a paragraph vector unique to that review and then analyze its sentiment in contrast to word2vec vectors that just need vector representations of words and analyzes the review by concatenating them. This even reduces the time required for sentiment analysis. Thus word2vec is better than doc2vec in terms of space and time complexity.

Our approach built on top of word2vec vectors gives two major advantages:

- It outperforms doc2vec in some cases and gives comparable accuracies in others.
- Running time is significantly low as compared to Doc2Vec, as we don't have to create a new paragraph vector to predict each of the new reviews, only word vectors are required for analysis.

For comparison, the same technique is used to classify reviews as positive and negative on Pang and Lee's dataset [11]. This dataset consists 2000 reviews in all, with 1000 positive reviews and 1000 negative reviews. This data consists of unprocessed, unlabeled html files from the IMDb archive of the reviews. The ratings were determined from the star-ratings explicitly given by the users on IMDB while mentioning their reviews. For a 5-star rating system, reviews with 3.5 stars and above are considered positive reviews and 2 stars and below are considered negative reviews.

The word vectors generated using Stanford dataset of training, test and unsupervised data are used here. Both the training and test data from Stanford dataset are used for training the classifiers here. There even, the accuracies obtained are comparable with Doc2Vec. The accuracies obtained for some of the classifiers are shown in the table for comparison.

Table 4 – Accuracies comparison of Doc2Vec and our modification applied to Word2vec

	Doc2Vec	**Our approach on Word2vec (LSVM classifier)**
Stanford dataset	88.708	**89.01**
Pang Lee Dataset	86.75	**87.4**

Thus, after predicting the reviews polarity, the producers can figure out whether the movie is reaching out to the people in a positive way or not. Accordingly, producer can change his marketing strategy to improve the perception of people towards the movie and thus can expect better returns on the movie. This can be done in multiple phases before the release of the movie,

so that at least producer can be assured that the movie will get a proper opening. Along with this, the distributor can analyze those reviews after predicting the initial gross and order more or less number of prints of that movie based on the people's eagerness and in turn, increase his profits too.

Chapter 5 Conclusion and Future work

The results for predicting the gross for the IMDB data are presented here. The best possible classifier i.e. random forest classifier, is used. Then sentiment analysis on the user reviews is done and sentiment polarity classification is obtained using various classifiers, along with the accuracies provided for each of them, and their running times. Along with this, our approach is discussed and accuracies obtained using that approach were comparable to Doc2Vec approach. Our technique is also tested on Pang and Lee's dataset and the accuracies are presented. Better accuracy with less time complexity and less space complexity was obtained. It is tentatively concluded that producer can thus change his marketing strategies according to the user reviews and thus increase profits for the movie and as a result the gross.

In future work, reviews from all social media possible, like twitter, YouTube, etc. can be included, not just IMDB. Also, the model can be extended to distributed systems to give the results in real-time.

Acknowledgement

We would like to thank Dr. Rupa Mehta (Head of the Department, Computer Engineering Department, SVNIT), and Dr. Dipti Rana, project guides, for their constant supervision, guidance and comments that helped us shape up this report.

We would also like to thank all the faculty members of Computer Engineering department, SVNIT who have always done their fair share in solving our queries, making us aware of new developments and competitions, exposing us to various technologies and encouraging reading of research papers. It is due to their constant efforts in our education that we became capable of taking up such and interesting topic.

References

[1] Maas, Andrew L. et al. “Learning Word Vectors for Sentiment Analysis.” ACL (2011).

[2] Rajaraman, A.; Ullman, J. D. Mining of Massive Datasets. Cambridge University Press, 2011, pp. 1–17.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop, 2013.

[4] Quoc Le, Tomas Mikolov; Distributed Representations of Sentences and Documents, Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2):1188-1196, 2014.

[5] Potts, Christopher. “On the negativity of negation.” (2011).

[6] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up? Sentiment classification using machine learning techniques, Proceedings of EMNLP, pp. 79--86, 2002.

[7] Malouf, Robert (2002). A Comparison of Algorithms for Maximum Entropy Parameter Estimation. Sixth Conf. on Natural Language Learning (CoNLL). pp. 49–55.

[8] Han J, Kamber M, Pei J (2006) Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems), 2nd ed.. Morgan Kaufmann, San Francisco, CA, USA.

[9] Vidushi Sharma, Sachin Rai, Anurag Dev, a Comprehensive Study of Artificial Neural Networks, Volume 2, Issue 10, October 2012.

[10] Andrew Ng, Machine Learning Yearning, Draft Version 0.5, 2016.

[11] Pang, B., Lee, Lillian, a Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In: Proc. of the 42nd ACL, pp. 271–278, 2004.