MOVIE REVIEW ANALYSIS

Final Project Report

Abstract

The aim of the project is to investigate with different deep learning approaches to predict the sentiment of movie reviews from the sentiments of the reviewers.

University of Connecticut

Team 7 - *Raghavendra, Roshan, Suvir, Surya, Yanfu*

OPIM 5671 – Data Mining and Business Intelligence

Contents

I. Executive Summary

Movie reviews provide decisive evidence about the likability of the movie and stay as the major deciding factor behind our choice to spend time and money to watch the movie in cinemas. Reviews are often multifaced and differ according to the review's individual taste. Our project aims on simplifying the exhausting process of going through heaps of reviews by generating quick visual summaries by analyzing and understanding sentiments of movie reviews and reviewers.

In this project, we generate a swift sketch of the terms(words) and their relationships with each other by exploring various aspects of the review textual data. First, we clean and parse the movie review data. Then we utilize this parsed data to perform vectorization for text mining. The vector word data in the form of words and documents is then clustered into topics/tags by Latent Semantic Analysis(LSA) and Latent Dirichlet Allocation(LDA). This text cluster topic element nodes can be utilized to identify the genre of the movie and identify crucial plot keywords that can help in determining the likability of the movie. Finally, the document vector data is used for generating the overall sentiment of each review and determine if the review gives good or bad opinion about the movie.

The aim of this report is to document our learnings about the various techniques we tested out to optimize the process of vectoring textual data, text mining, modeling for sentiment analysis and text cluster visualization. We also discuss the challenges faced, intriguing insights, recommendations, and future scope for this project. Concepts like Natural Language Processing and Deep Learning for text understanding were also studied and explored as a part of this project.

II. Business Problem

With over half-a-million fiction feature-length or theatrical-cinema films in the world, the number of reviews for movies world-wide is colossal. Every new theatrical release movie these days get at least a thousand reviews spilled across websites, blogs, social media, and mass-media. While most movie goers settle on their choice of movie based on reading reviews, the deciding process gets tedious with such vast review data. These reviews are largely in the form of online reviews and such reviews can be gathered and analyzed to create concise abstracts of the movie based on public opinion. Reviews and analysis on these reviews even help the movie makers themselves make better movies.

III. Background

The project is carried out in Python programming language with the intension of exploring new technology with methods and techniques learnt in class. The dataset used for performing text mining and sentiment analysis was obtained from Kaggle containing reviews of 50,000 movie reviews from the online movie review website – Internet Movie Database (IMDb).

1. Overview of Data

The whole dataset of IMDb movie reviews consists of over 100,000 movie reviews and is specially selected for sentiment analysis. The target prediction variable / sentiment of reviews is binary, meaning IMDb sentiment scores of 5 or less gets a rating of 0 and those that is greater than 6 is rated 1. To maintain variance in data, no individual movie has more than 30 reviews in the whole dataset. The dataset is conveniently split into training and testing sets with each comprising of 25,000 reviews paired with a random unique ID for identification. There is also an unlabeled training data of 50,000 reviews that does not have a rating label. This set has been retained for more advanced analysis and can be utilized for future scope.

Name	Description
id	Unique ID of each review
sentiment	Sentiment of the review; 1 for positive reviews and 0 for negative reviews
review	Text of the review

2. Technical Overview

We researched a variety of techniques as a part of this project and have attempted to briefly describe the concepts behind them, along with how the IMDb data is utilized to perform text mining and analysis using the said techniques.

i. Word2Vec

Word2Vec takes a text corpus as input and produces word vectors as output. Tomáš Mikolov's Word2vec algorithm uses a large amount of text to create high-dimensional (50-300 dimensional) representations of words capturing relationships between words unaided by external annotations, capturing many linguistic regularities. The algorithm learns vector representation of words from training

test data and constructs a vocabulary. For instance, [king] - [man] + [woman] ~= [queen]. There are two main learning algorithms in word2vec – continuous bag-of-words and continuous skip-gram.

a. Continuous skip-gram

Word2vec uses a single hidden layer, fully connected neural network. The input layer is set to have as many neurons as there are words in the vocabulary for training. If the vocabulary for learning word vectors consists of V words and N to be the dimension of word vectors, the input to hidden layer connections can be represented by matrix WI of size VxN with each row representing a vocabulary word. In same way, the connections from hidden layer to output layer can be described by matrix WO of size NxV. In this case, each column of WO matrix represents a word from the given vocabulary. The input to the network is encoded using "1-out of -V" representation meaning that only one input line is set to one and rest of the input lines are set to zero.[1]

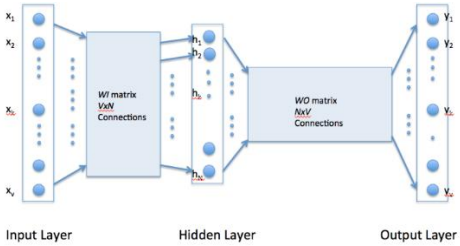


Fig: Representation of Neural Network working in continuous skip-gram

b. Continuous bag-of-words

In the continuous bag of words(CBOW) model, the context is represented by multiple words for a given number of target words. For instance, using words "police" and "thief" as context words for the target word "arrested". The neural network architecture is modified to replicate the input to hidden layer connection C times (number of context words) and adding a divide by C operation in the hidden layer neurons.[1]

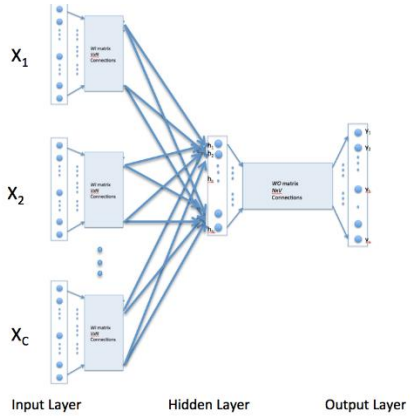


Fig: Representation of Neural Network working in continuous bag-of-wordss

ii. Latent Semantic Analysis

Latent Semantic Analysis is a Natural Language Processing technique of analyzing relationships between a set of documents (sentences/paragraph chunks) and the terms they contain by generating a set of context topics related to that documents and terms. This algorithm works based on the assumption that words whose meanings are closer (synonyms) will occur in similar contexts/pieces of text. A matrix of word counts per document is constructed with words in rows and columns for paragraphs. Singular Value decomposition is used to reduce the number of rows while preserving the similarity of columns. Words are then compared by dot product or other calculations between two vectors formed by any two rows. Values represent similarity with 1 or close to 1 meaning very similar words and vice versa.

iii. Latent Dirichlet Allocation

Latent Dirichlet Allocation is a statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar and is used for *topic modeling*. If observations are words collected into documents, it suggests that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. A topic generates various words, such as “action”, “marvel”, and “comic”, which can be classified and interpreted by the viewer as "SUPERHERO_related".

iv. K-means text clustering

A predefined number of clusters have to be setup for k mean clustering. K means helps us to understand the hidden structure of data segmentation. In K means each observation is assigned to predefined cluster to minimize the within cluster sum of square. In next step, mean of mean of cluster observation is calculated and used for new cluster centroid. Later , the observation are reassigned to clusters and centroids recalculate and this process keep in loop until algorithm reach its convergence.

IV. Project Methodology

We carried out a 6-step approach to our project, starting with exploration of the textual data followed by Vectorization, Clustering, Modeling, Model Selection and LDA (Topic Modeling).

1. Data Preprocessing

The Movie review data is humongous with a lot of value but a lot of noise. In order to achieve better results in modeling for Text Mining and Sentiment Analysis, the data was preprocessed to reduce noisy values through the following steps, included with Python code snippets.

i. Removing Stop Words

Stop words are words that occur very frequently in the common language like *the, is, at, which, who, on, at* etc., that no significant insight can be obtained. These words are filtered out before performing natural language processing to reduce data bulk and increase result speeds.

```
from nltk.corpus import stopwords
# ...
filtered_words = [word for word in word_list if word not in
stopwords.words('english')]
```

ii. Splitting Attached Words

Text data from social media and online platforms commonly contain hashtag words which has to be split to obtain meaningful words. Such hashtag words are meaningful words without spacing and would be generally readable to common eye. However, machine learning algorithms consider such words as a single word, rendering them meaningless. Ex: #InceptionMovieisAwesome, #GreatMovieToWatch. This was achieved using an experimental function in python – *HashTagSplitter*.

```
split_hashtag_to_words_all_possibilities("edgeofentertainment")
[['edge', 'of', 'entertainment']]
```

iii. Standardizing words

Words that are not in proper format - words often misspelled to convey stressing points or slang should be processed into proper format abiding grammar and spelling rules.

```
data = ''.join(''.join(s)[:2] for _, s in itertools.groupby(data))
```

2.Vectorization

Natural language processing is based on learning capability of computer about English language which involve hierarchical, word ambiguities and sparse nature of sentence words etc. Machine work in tandem as support mechanism to help identify and summarize information.

Here our idea, is to provide our machine a dictionary, we feed a large amount of words to our algorithm word2vec, with the aim to provide learning of English context. Algorithm learn from these words and try to predict other relevant words in paragraph or sentence. Internally, these words are represented in form of vectors (high dimensional). Each axis of vector in multi-dimensional space represent characteristic and magnitude represents the closeness of that property to that word. So, we can say that more similar words are more close than other dissimilar words in vector space. We utilized numpy and pandas to generate vectorization using the tokenizer function for words present in the document.

```
vectorizer = CountVectorizer(analyzer = "word",   \
                             tokenizer = None,    \
                             preprocessor = None, \
                             stop_words = None,   \
                             max_features = 5000)
```

3. Clustering

For our second modelling approach which is based on unsupervised learning, we also utilized clustering technique. Clustering algorithm group set of documents into clusters. The algorithms' goal is to create clusters that are coherent internally, but clearly different from each other. In other words, documents within a cluster should be as similar as possible; and documents in one cluster should be as dissimilar as possible from documents in other clusters.

Clustering was one of the most important part of our project, and with on doubt, it also determined whether the text analytics could work well. For this, we used the *sklearn* cluster packages. Given the data size, we decided to cluster the feature of data using K-means. We set the K value first and count the time for the process of extracting centroids. To be honest, it was tough for us to choose a good k value. It was suggested by expert that 5 words in each cluster is good. But in our case, since we had so many data, we could reach a higher amount. Therefore, we decided to use 10 as our K, instead of 5. At that stage, we took look at what cluster we had, and we were excited to find the result that was what we expected. Then, we create a function that converts reviews into bags-of- centroids and loop reviews using the function, making them different centroids.

```
In [19]: from sklearn.cluster import KMeans
    ...: import time
    ...:
    ...: start = time.time() # Start time
    ...:
    ...: # Set "k" (num_clusters) to be 1/5th of the vocabulary size, or an
    ...: # average of 5 words per cluster
    ...: word_vectors = model.wv.syn0
    ...: num_clusters = word_vectors.shape[0] / 5
    ...:
    ...: # Initalize a k-means object and use it to extract centroids
    ...: kmeans_clustering = KMeans( n_clusters = num_clusters )
    ...: idx = kmeans_clustering.fit_predict( word_vectors )
    ...:
    ...: # Get the end time and print how long the process took
    ...: end = time.time()
    ...: elapsed = end - start
    ...: print "Time taken for K Means clustering: ", elapsed, "seconds."
    ...:
Time taken for K Means clustering:  537.146999836 seconds.
```

```
Cluster 0
[u'assignment']

Cluster 1
[u'clutter', u'bangkok', u'manor', u'formerly', u'brighton', u'es', u'dublin',
u'hilton']

Cluster 2
[u'cody', u'hyde', u'gandalf', u'moses', u'earl', u'jones', u'bean', u'cain',
u'vinnie']

luster 3
u'distortion']

luster 4
u'wandering', u'walking', u'aimlessly', u'continuously', u'wander', u'riding',
'circles', u'prancing', u'stumbling', u'roaming', u'driving']

luster 5
u'occupied', u'conquered', u'ruled', u'devastated', u'sponsored', u'hijacked',
'destroyed', u'invaded', u'flown', u'ravaged']

luster 6
u'smashed', u'sliced', u'blasted', u'bashed', u'gutted']

luster 7
u'kelly', u'penelope', u'sharon', u'rhonda', u'blaine', u'parker', u'trey',
'eleanor', u'marian']

luster 8
u'simultaneously', u'wildly', u'decidedly']

luster 9
u'rewarded', u'enthralled', u'satisfied', u'engrossed', u'impressed']
```

Fig: Generated 9 different cluster in k-means algorithm

4. Modeling

We have utilized two different approaches first one is utilization of bag of words algorithm involving continuous vectorization of words which based on frequency of presence of set of words while the other is based on word2vec algorithm. Bag of words learn to use the document as vocabulary, then using the frequency of words. Word2vec is deep learning algorithm based on creation of distributed word vectors. Word2vec algorithm have a edge over other deep learning or unsupervised learning techniques like neural nets in terms of training model period especially when we have large amount of data. Also, word2vec doesn't require any kind of labelling to create representation. Word2vec is implemented using the genism package, for this project. For both approaches, we decided to utilize random forest model for fitting based of generated vectors.

5. Model Selection

Both approaches are quite strong and produce quite close prediction. Based on the accuracy and misclassification rate, bag of words marginally outperformed word2vec by 1%. This may be due to the not utilizing the weight while generating vectors for word2vec.

Below are the Graphs/plots generated using models based on both approach:

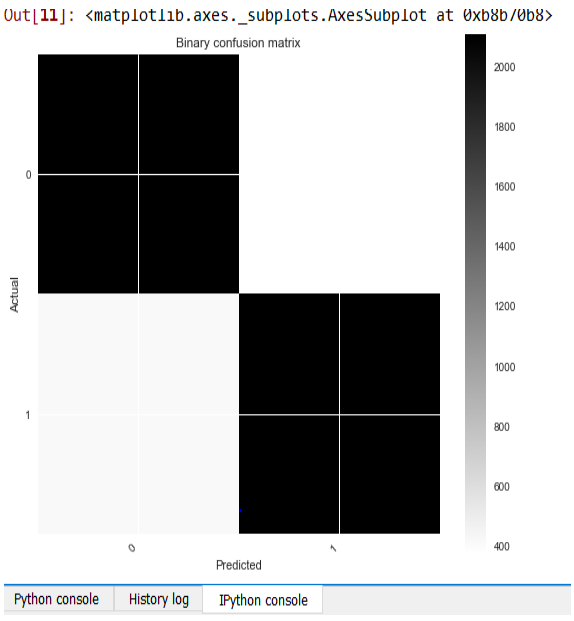


Fig: Confusion Matrix – Bag of words

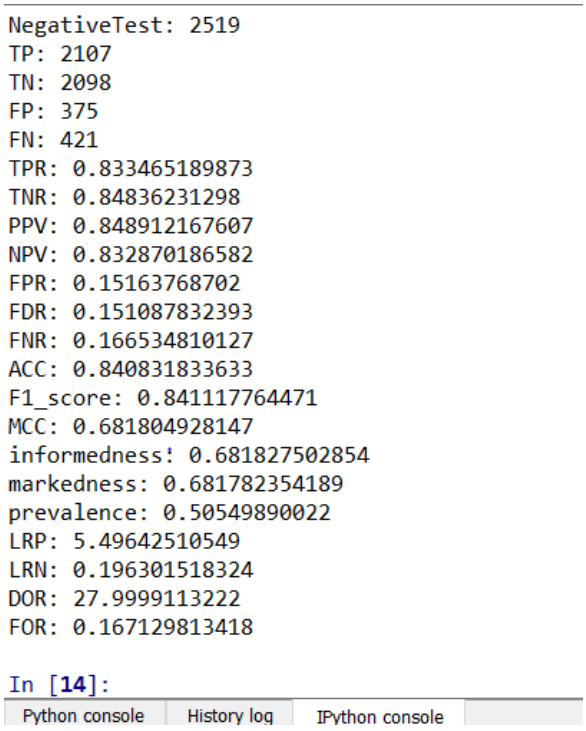


Fig: Fit Statistics – Bag of Words

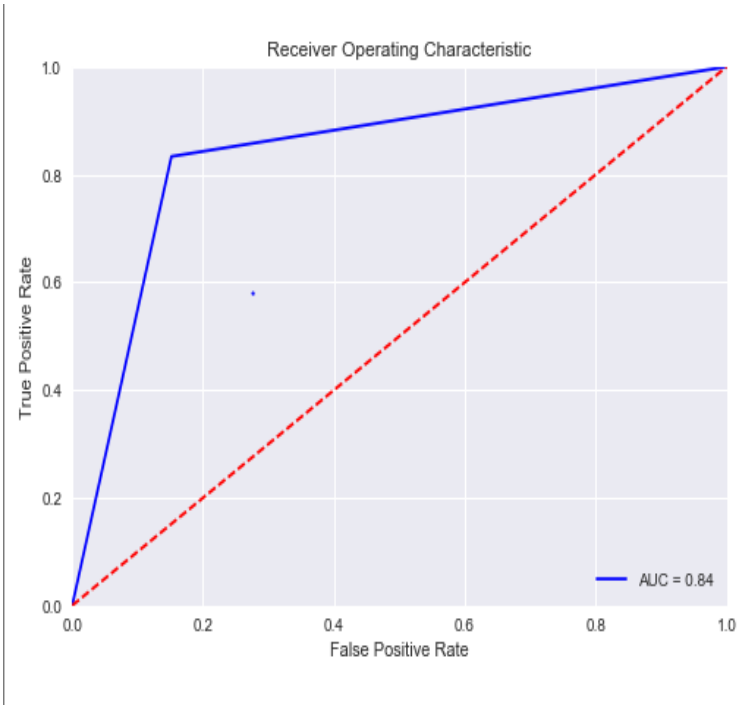


Fig: AUC Curve – Bag of Words.

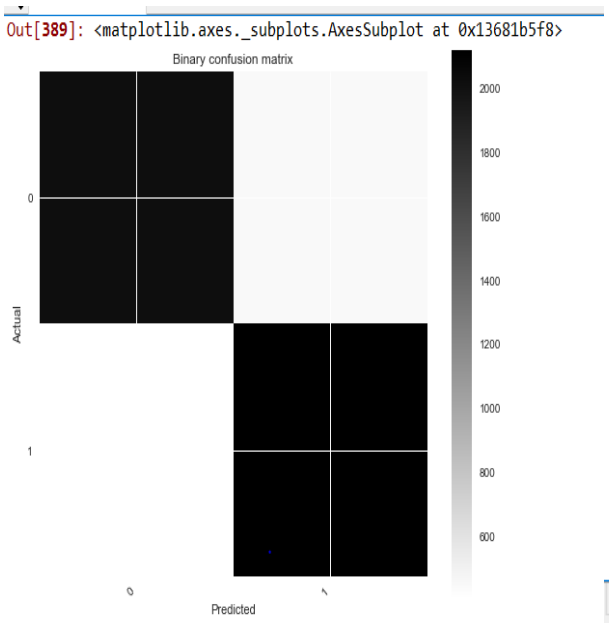


Fig: Confusion Matrix – Word2vec

```
population: 5001
P: 2528
N: 2473
PositiveTest: 2572
NegativeTest: 2429
TP: 2118
TN: 2019
FP: 454
FN: 410
TPR: 0.837816455696
TNR: 0.816417306915
PPV: 0.823483670295
NPV: 0.831206257719
FPR: 0.183582693085
FDR: 0.176516329705
FNR: 0.162183544304
ACC: 0.827234553089
F1_score: 0.830588235294
MCC: 0.654461805569
informedness: 0.654233762611
markedness: 0.654689928015
prevalence: 0.50549890022
LRP: 4.56370064964
```

Python console | History log | IPython console

Fig: Fit statistics – Word2vec

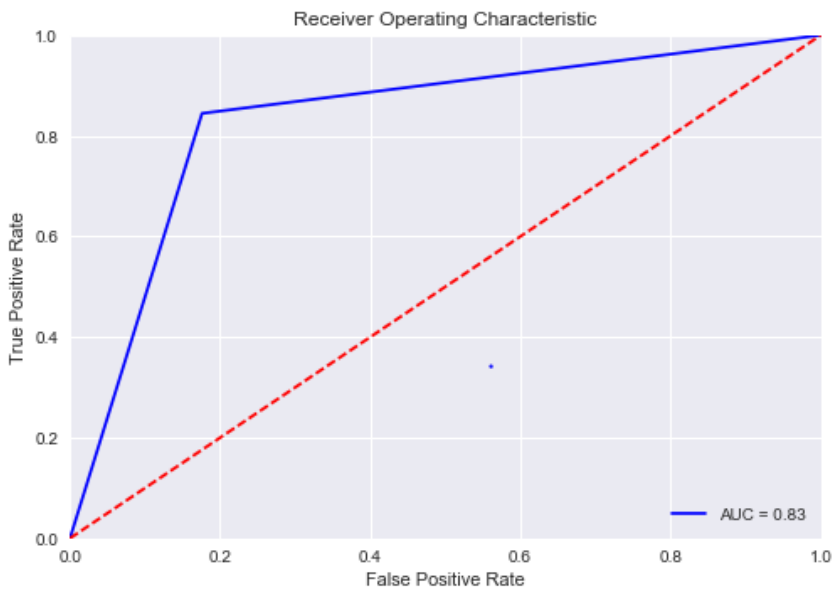
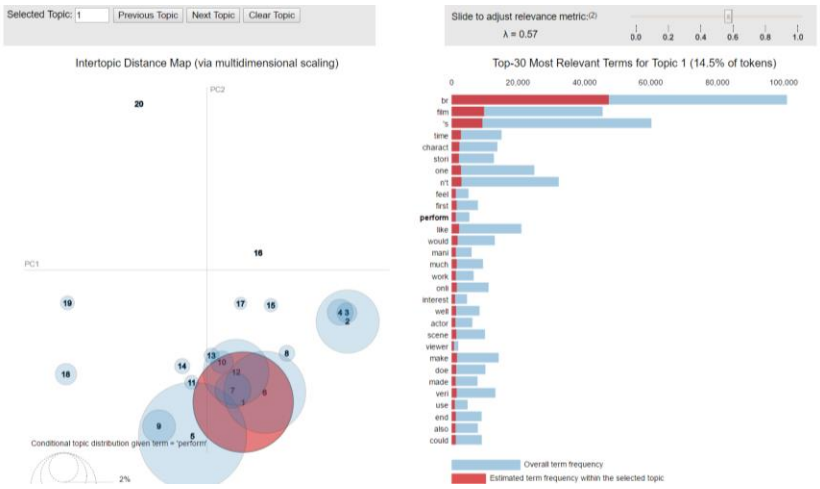


Fig: AUC Curve – word2vec

6. LDA

We also utilized LDA algorithm / technique to get better understanding of relationship among words. Latent Dirichlet Allocation is a statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. Below are the some of our findings visualized to emphasize the importance of graphs.



u'0.037*"br" + 0.028*"film" + 0.023*"\'s" + 0.007*"stori" + 0.007*"one" + 0.007*"charact" + 0.005*"veri" + 0.004*"make" + 0.004*"time" + 0.004*"life"'),

V. Challenges

We faced lot of challenges, while working on this project, particularly in preprocessing and the vectorization of the words using the natural language processing. Data have lot of ambiguity like of presence of backslash, html, extra spacing, stop words etc. All these wording issues and ambiguity need to be removed to improve prediction ability of a model in other words to improve the learning ability of algorithm. Another challenge is to decide which n-gram approach we should use, we can use bi gram or tri gram approach as well, it may improve learning ability of model. But at same time, it will include lot of complexity as well as require high computational.

VI. Future Scope

This project provides lot of insight about the natural language processing and its importance in text mining. There is lot of scope to learn and extend this project in coming future. Several enhancements can be done like currently word2vec algorithm even through word2vec is quick in comparison of bag of words algorithm in terms of training model / running efficiency itis performing slightly lower than the bag of words algorithm. This can be improved, if we utilize weights on vectors using the techniques like “tf-idf” weights, this will surely improve prediction ability of word2vec as weighting allow us to decide importance of word in the document.

Similarly, we currently generating binary outcome, positive or negative review as our training data have outcome in binary format. We can utilize our approaches or model for rating movies on scale of 1-10 or 1-5, if we have rating information about the movie based on the review.

VII. Conclusion

Bag of words and word2vec both proved to be important algorithm of natural language processing for sentiment analysis. Although, bag of word algorithm slightly more accurate than word2vec, word2vec algorithm holds lot of potential due to its unsupervised learning ability. Due to unsupervised learning ability and less processing time, word2vec algorithm can be used for vast variety of text mining analysis.

The major advantage of using text mining for data analysis is that it helps businesses to create marketing and campaigning strategy to not only capture potential customer but also to mitigate customer churn by analyzing the customer sentiments using deep learning techniques. Also, these techniques will reduce lot of manual effort in categorizing / segmenting the product like in our case movies recommendation to customers.

Hence, the focus of this project is to help business to devise effective marketing strategy by analyzing and predicting the movies popularity using the text mining.

VIII. References

https://iksinc.wordpress.com/tag/continuous-bag-of-words-cbow/

https://code.google.com/archive/p/word2vec/

https://www.kaggle.com/c/word2vec-nlp-tutorial