Automating audience rating prediction for online movie reviews using sentiment analysis

Avish Manish Kadakia
Student ID: 1132452
Dept. of Computer Science
Lakehead University
Thunderbay, Ontario
Email: kadakiaa@lakeheadu.ca

Rahul Kishorbhai Pipaliya
Student ID: 1153308
Dept. of Computer Science
Lakehead University
Thunderbay, Ontario
Email: rpipaliy@lakeheadu.ca

***Abstract*—The 21st century has proven to be a pioneering era and a cornerstone for a plethora of innovations in the field of AI (Artificial Intelligence) and NLP(Natural Language Processing) almost regularly. In the discipline which is so highly contested and always has some cutting edge technologies to be discovered by researchers at every junction, we decided to take a few steps back and try to optimize one of the fundamental concepts of NLP sentiment analysis. When it comes to NLP, the quality of the data can perform wonders for the research in question. So we have decided to stick to the elementary but highly optimized IMDB user rating dataset. Our goal is to determine users' rating for a particular movie review. We choose this problem statement as it can prove to be a gateway for various other similar topics of research such as Twitter User Sentiment Analysis or Amazon Customer Rating Prediction. Researchers have tried to tackle the same conundrum in the past by using many well-known algorithms such as Multinomial Naive Bayes, CountVectors, Google's Word2Vec, LSA(Latent Semantic Analysis), etcetera. Our research will not focus on creating something groundbreaking but would focus on gaining a deeper understanding of previous implementations and attempting to either modify or optimize them to yield even better results. Let us now see an analysis for some of the previous approaches used to tackle this problem.**

***Index Terms*—sentimental analysis, pattern recognition, emotion classification, opinion mining**

INTRODUCTION

The web has transformed the world drastically and the rise of web 2.0 has changed the scenario significantly as people can express their thoughts and opinions digitally. For example, if somebody wants to buy a cell phone, they can access the internet to read customer reviews before their purchase. Similarly, they can read reviews about movies online to determine if a featured film would be worth their time. The web has given its users the freedom of speech using which they can write their unbiased opinions in a blog or review. We can examine these user reviews to predict the behaviour of consumers and also identify new opportunities in the market. This process is called sentiment analysis and is known as opinion mining. Experimental results indicate that training on written movie reviews is a promising alternative to exclusively using (spoken)in-domain data for building a system that analyses movie reviews.

Our work will, therefore, focus on automatically predicting ratings for online movie reviews using primarily textual data. We have chosen the IMDB user movie review database for our problem statement as it has over 100,000 labelled user movie reviews. Each review has a rating from 1 - 4 (negative) and 7 - 10(Positive). Where (1) is the worst and, (10) is the best. Sentiment analysis, particularly the automatic analysis of written reviews in terms of positive or negative valence, has been extensively studied in the last decade. It has been widely accepted by the research community that written movie reviews seem to be rather arduous to handle. Why? One of the apparent challenges in classifying textual movie reviews is that sentiment words often relate to the elements of a movie rather than the reviewers' opinion. For instance, words we would usually associate with strongly negative valence, such as "nightmare" or "terrifying", could be used in a positive review of a horror movie.

As a first step, towards more robust sentiment analysis in written movie reviews, we propose expanding on the previous implementations of these highly effective algorithms KNN, Naive Bayes and SVM(Support Vector Machine) to try and achieve better results for the same problem.

LITERATURE REVIEW

Sentiment analysis of movie reviews aims to estimate emotions of cinema audiences so that movie theatres can estimate whether to continue showing a movie or not based on audience sentiment. With the advent of modern natural language processing techniques predicting emotion is an achievable task by computers using pattern recognition algorithms. There is some work done previously by researchers in this field.

A. *Learning Word Vectors for Sentiment Analysis [1]*

This research paper is associated with the dataset that we have selected. The research presented a model which uses a mix of unsupervised and supervised techniques to learn word vectors which captures semantic term–document information as well as rich sentiment content; this helps in finding the similarity between words. To find semantic similarities among words, they derive a probabilistic model of documents which

learns word representations. Their model evaluates categorisation of document-level and sentence-level tasks in the domain of online movie reviews. For capturing semantic similarities, they build a probabilistic model of a document using a continuous mixture distribution over words arranged by a multi-dimensional random variable. For capturing semantic similarities, they created a probabilistic model of a document using a continuous mixture distribution over words arrayed by a multi-dimensional random variable.

They extended the unsupervised model to incorporate sentiment information which showed that this extended model could leverage the abundance of sentiment-labelled texts available online to yield word representations that capture both sentiments and semantic relations. To find word sentiment, the researchers calculated conditional probability and employed a logistic regression as their predictor. Finally, they used the technique of alternating maximisation that improves word representations using MAP estimates. Their model's probabilistic foundation gives a theoretically justified algorithm for word vector induction. They extended the unsupervised model to incorporate sentiment information which showed that this extended model could leverage the abundance of sentiment-labelled texts available online to yield word representations that capture both sentiments and semantic relations. Their model can be used to differentiate a wide variety of annotations, and thus is broadly applicable in the growing areas of sentiment analysis and retrieval.

B. Sentiment Analysis of Movie Review Comments [2]

This paper presents an empirical study of the efficacy of machine learning techniques in classifying text messages by semantic meaning. The authors have used movie review comments from popular social network Digg as their data set and organise text by subjectivity/objectivity and negative/positive attitude. They have proposed different approaches in extracting text features such as bag-of-words model, using huge movie reviews corpus, restricting to adjectives and adverbs, handling negations, bounding word frequencies by a threshold, and using WordNet synonyms knowledge.

Finally, the authors evaluate their effect on the accuracy of four machine learning methods - Naive Bayes, Decision Trees, Maximum-Entropy, and K-Means clustering. Their methods of sentiment analysis are based on machine learning. For extracting features from the dataset, authors have used the bag-of-words algorithm. Since the bag-of-words do not capture synonyms in the language, the writers used WordNet.To analyse the polarity of the sentence, the writers handled the negation of the sentence using [word]-NOT for feature selection of adjectives and adverbs. For classification of movie critics, they considered three supervised – Naive Bayes, Maximum Entropy and Decision Trees, and one unsupervised classification approach – K-Means clustering. All four algorithms are available in the NLTK framework. They measured the accuracy of automated classification for each corpus and each label using 10-fold cross-validation.

C. Opinion Mining and Sentiment Analysis on Online Customer Review [3]

This research paper concentrates on mining reviews from websites like Amazon, which allows users to write their views. It automatically extracts the reviews from the website. It also uses algorithms such as the Naïve Bayes classifier, Logistic Regression and SentiWordNet algorithm to classify the survey as a positive and negative review. In the end, the authors have used quality metric parameters to measure the performance of each algorithm. The following image shows the data flow of the system proposed by the authors.

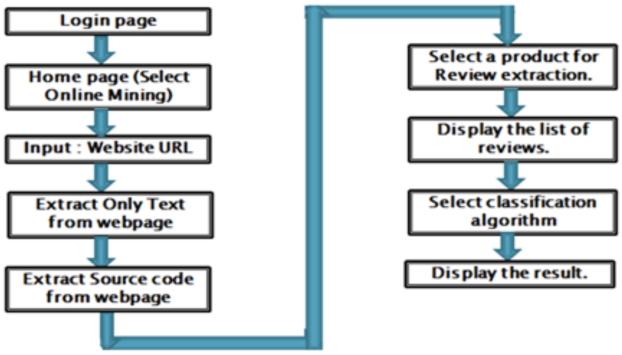

Fig. 1: Data flow of the system proposed by the authors. [3]

The performance of the classifications methods was measured using recall, precision, F-measure. According to their experiment, the Naïve Bayes classification proves to be the most efficient among three algorithms for text classification of opinion mining.

OUR APPROACH

We are planning to generate the ratings for the movies based on their reviews as presented in the selected data set. To do so, we will follow some steps mentioned below:

- We will extract features from the dataset in the form of a word vector using the feature extraction algorithm such as TF-IDF Encoding, latent semantic analysis encoding, Word2Vec embedding, GloVe.
- Next, we would classify the extracted dataset into ratings between 1 to 10 using the pattern recognition algorithms, namely, KNN, Support Vector Machine(SVM), and Multinomial Naive Bayes.
- We would test the algorithm with cross-validation and finally calculate training accuracy and testing accuracy. We are also inclined towards finding Recall, F-score, and other precision metrics.
- Finally, we would study the difference in accuracy for these algorithms for the same data set.

The comparison can be used by machine learning experts to hypothesize a reliable algorithm to predict movie ratings based on a user's reviews. Furthermore, the results may vary for different datasets for different algorithms. So our comparisons

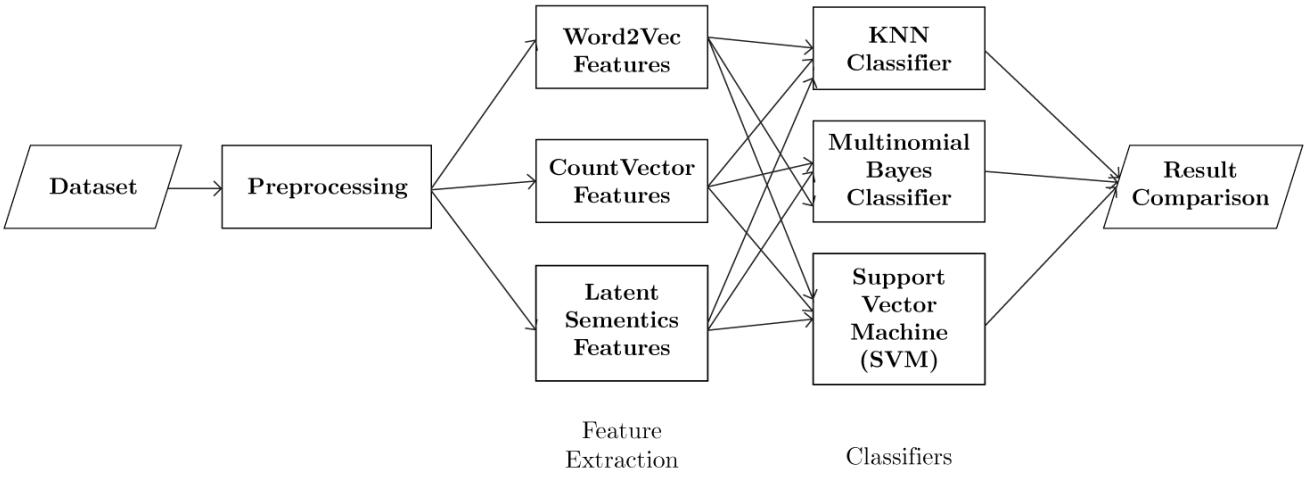

Fig. 2: System Overview Diagram

will also help researchers gain a base understanding of the algorithms for their particular use case.

METHODOLOGY AND JUSTIFICATION

For our predictions we will be following the standard practice for working with text data consisting of the following steps (Fig. 2):

- Importing Dataset
- Data Preprocessing
- Word Embeddings / Feature Extraction
- Making predictions using classifiers

As importing dataset and preprocessing are quite self-explanatory let us move to the more complicated parts of the implementation.

I. FEATURE EXTRACTION / WORD EMBEDDINGS

When it comes to developing any sort of prediction algorithm it is always best to work with numerical data as we can perform a plethora of mathematical operations and extract valuable information from them. The various techniques which are used to convert textual data to numerical data are called Feature Extraction or Word Embedding. For our use case we chose the following 3 methods:

A. Word2Vec [4] [5]

Word2Vec is a very well known technique in the NLP field for feature extraction, Introduced by Tomas Mikolov at Google in 2013. It uses a simple one-layer neural network to perform feature extraction and gives us a final representation of words in the form of a vector. It has 2 Models, Continuous Bag-of-Word(CBOW) Model, and Skip-Gram Model. CBOW model predicts the current word based on context while the Skip-Gram model predicts surrounding words given the current word.

Our approach would be to extract one hot encoded representation for all the words. We will then pass these vectors as input to a one-layer feed-forward network. The neural network will use softmax as its activation function. The network will then train and update its weights using a gradient descent algorithm. The weights at the end of the training process will be our final word vector. Using which we will be able to train our classifiers and generate predictions.

B. Latent Semantic Analysis(LSA) [6] [7]

Using this technique we will try to extract the Latent semantics i.e. hidden features that represent something essential in our dataset. It's an unsupervised technique. Once we have our counter vector we will also use various dimension reduction techniques. LSA is divided into 2 main parts, Generating Document Term matrix (Fig. 3) and Performing Singular Value Decomposition on that matrix to get 3 matrices (S, V and Σ). After performing SVD we will splice the matrix and get the first few columns and combine all matrices again so that we can easily calculate cosine similarity between words.This similarity can then be used by the classifiers to make more accurate predictions as it will also encode the semantic similarity between words.

C. CountVectors

CountVectorizer is a fairly simple algorithm that works on Term Frequency, i.e. counting the occurrences of tokens and building a sparse matrix of document X tokens. Once we have the count vectors we will then be able to apply various classifiers on the matrix to generate predictions. CountVectors is a very basic technique to convert words to vectors as it does not take into consideration any relation the words might have with each other. But even this basic method may result in a decent accuracy boost for certain cases as compared to directly working with textual data.

	brown	dog	fox	lazy	quick	red	slow	the	yellow
"the quick brown fox"	1	0	1	0	1	0	0	1	0
"the slow brown dog"	1	1	0	0	0	0	1	1	0
"the quick red fox"	0	1	0	0	1	1	0	1	0
"the lazy yellow fox"	0	0	1	1	0	0	0	1	1

Fig. 3: Document Term matrix

II. Classifiers

Our final goal is to predict user sentiment between 0-10(0 being worst and 10 being best). As this is a classification problem we will be using various multiclass classifiers in order to get the predictions. The results from the various classifiers we will be comparing are as follows:

A. *KNN (K Nearest Neighbour)*

KNN is a simple but powerful algorithm that can achieve high accuracy. The main idea in KNN is to calculate the distance between each word vector during our training process. We then compare each new word vector with these distances to obtain the K nearest neighbours and make our predictions. The only drawback of using KNN is that it may have a very high time complexity if the dataset is too large.

B. *Multinomial Naive Bayes*

$$\Pr(c|t_i) = \frac{\Pr(c)\Pr(t_i|c)}{\Pr(t_i)}, \qquad c \in C$$

Fig. 4: Bayes Theorem [8]

In Multinomial Naive Bayes, we will calculate the probabilities of each word for a given document, using the Bayes formula (Fig. 4). We can then calculate the highest probability and assign that class to a new document. We will then calculate priors Pr(c) by dividing the number of documents belonging to class c by the total number of documents and also calculate rest of the terms in formula. The approach is quite straightforward in theory but one of the issues is that it does not take into account the similarity between words. This is one of the reasons why it is referred to as a Naive approach. The predictions can still have high accuracy irrespective of this drawback. As can be seen in the research paper "Multinomial naive bayes for text categorization revisited" [8]

C. *SVM (Support Vector Machine) [9]*

SVM is a supervised learning model that has a solid theoretical foundation and performs classification more accurately than most other algorithms in many applications. Many researchers claim that this is the best text classifier and can be used for sentiment analysis to get great performance.

SVM linearly separates data and finds a hyperplane that fits best between 2 classes but when data is not linearly separable then we can convert data to a higher dimension and hope to find a hyperplane in a higher dimension, we can do this using kernel functions.

Experiments

We have used a few Python 3.0 libraries like scikit-learn(sklearn), nltk(NLP Library), gensim(for word2Vec), and numpy for our experiment.

Our initial step was loading the data and splitting it into training and testing sets.

After loading the dataset we have performed several pre-processing steps to make the later process have better results. As we are working with textual data we will first be cleaning the dataset by performing the following operations:

- Remove special characters
- Remove all single characters
- Substituting multiple spaces with a single space
- Converting to lowercase

After cleaning our data we will be performing word net lemmatization after splitting of documents so that words are converted to their proper form and we avoid creating unnecessary word vectors for the same word during the transformation phase.

At this stage, we will have 4 preprocessed arrays, 2 consisting of training reviews and their labels (Let's call it X_train, y_train) and the other 2 consisting of testing reviews and their labels (Let's call it X_test, y_test).

Our next step was to transform the training and testing reviews to some vector representation. This will enable us to perform a number of mathematical operations on the vectors to obtain meaning and relation between them. We can also pass these vectorized arrays to various classification algorithms directly. For our experiment we have to use these 3 different transformation techniques for feature extraction/word embeddings:

CountVector

First, we transform data(X_train, X_test) using the countVectorizer function so that each input will have a feature size of 5000. Then perform tfidfTransformer on the vectorized data to embed meaning for the frequency of each word.

The next step would be to Add padding along the axis of testing and training vectors to make them of the same size so that we can apply the KNN classifier easily. then we would have a sparse matrix containing lots of 0s so we convert training and testing sparse matrix to the dense matrix in order to make classification more efficient.

Word2Vec

We have created a model that was trained using gensim's word2vec model using the training review vector i.e. X_train as input and also we are using $size = 100$, which gives us 100 features. Then we transformed the training and testing reviews using this model. As the transformation process may yield features with negative values we scaled our training and testing review vectors to have only positive values between 0-1.

Latent Semantic Analysis(LSA)

For LSA transformation, we first vectorized our review vectors using TfidfVectorizer. After which we project the TF-IDF vectors onto the first 100 principal components. Next, we performed SVD on the training data and projected the results to the training review vector, and also applied the same transformation to the test review vector as well.

As performing the LSA transformation process may yield features with negative values we scaled our training and testing review vectors to have only positive values between 0-1.

The final step is to shuffle our train and test vectors (X_train, X_test) so that we can then apply various classifiers to obtain results.

We will be using the following classifiers for our experiment:

- Sklean's KNN classifier
- Sklean's MultinmialNaiveBayes multiclass classifier
- Sklean's SVM module's multiclass svc classifier

Before calculating the results for the KNN classifies we will first find the best average k by incrementing k from 1 to 20 for 5 iterations using 5-fold cross-validation and then averaging the results

RESULTS

Our results for the various different combinations of transformation techniques and classifiers have been illustrated in the tables below.

TABLE I: Results with 8 classes

	CountVector	Word2Vec	LSA
KNN(k=19)	0.174	0.3394	0.3028
MNNaiveBayes	0.176	0.3139	0.2704
SVM	0.203	**0.3970**	0.3862

TABLE II: Results with 4 classes

	CountVector	Word2Vec	LSA
KNN(k=19)	0.2786	0.6076	0.4644
MNNaiveBayes	0.3006	0.5053	0.4126
SVM	0.3068	**0.6548**	0.5626

TABLE III: Results with 2 classes

	CountVector	Word2Vec	LSA
KNN(k=19)	0.5125	0.8021	0.734
MNNaiveBayes	0.5378	0.7246	0.7968
SVM	0.5285	**0.8383**	0.8234

As we can infer from the table using a combination of Word2Vec with SVM yields us the best results giving us accuracy higher than 83% for 2 classes.

Now, let's see some of best results with full report.

TABLE IV: With 8 Classes (Best result: Word2Vec & SVM)

	Precision	Recall	F1-Score	Support
Rating 1 (1)	0.47	0.85	0.60	5022
Rating 2 (2)	1.00	0.00	0.00	2302
Rating 3 (3)	0.30	0.01	0.03	2541
Rating 4 (4)	0.27	0.37	0.31	2635
Rating 5 (7)	0.22	0.00	0.01	2307
Rating 6 (8)	0.26	0.25	0.25	2850
Rating 7 (9)	1.00	0.00	0.00	2344
Rating 8 (10)	0.42	0.79	0.55	4999
Accuracy			0.40	25000
Macro avg	0.49	0.28	0.22	25000
Weighted avg	0.47	0.40	0.30	25000

TABLE V: With 4 Classes (Best result: Word2Vec & SVM)

	Precision	Recall	F1-Score	Support
Rating 1 (1-2)	0.67	0.76	0.71	7324
Rating 2 (3-4)	0.51	0.34	0.41	5176
Rating 3 (7-8)	1.00	0.00	0.00	2307
Rating 4 (9-10)	0.68	0.89	0.77	10193
Accuracy			0.65	25000
Macro avg	0.72	0.50	0.47	25000
Weighted avg	0.67	0.65	0.61	25000

TABLE VI: With 2 Classes (Best result: Word2Vec & SVM)

	Precision	Recall	F1-Score	Support
Rating 1 (1-4)	0.84	0.83	0.84	12500
Rating 2 (7-10)	0.84	0.84	0.84	12500
Accuracy			0.84	25000
Macro avg	0.84	0.84	0.84	25000
Weighted avg	0.84	0.84	0.84	25000

CONCLUSION

As per our experimentation results, it can be clearly observed that more than the transformation techniques and classifiers used for sentiment analysis the number of classes has a much higher impact on the accuracy especially for the dataset we have used. There may be multiple reasons for such performance but as per our understanding, this is because even as humans are given a review with a rating of 10 and asked to classify it we may just as easily classify it as having an 8 or 9 rating. The main reason for this being providing a rating for a review can be quite arbitrary and is heavily influenced by an individual's opinion and cannot be predicted, as 2 people who may be given the same review may rate them differently based on their opinions. This makes it quite hard to find a determinable pattern to rate reviews for a classifier. As a result, even the algorithms that we currently use cannot guarantee high accuracy. The current methods are best at approximating at a high level if a review is positive or negative rather than predicting the exact rating for the reviews. As it is clearly evident from our results even though the same embedding techniques and classifiers were used, the accuracy we achieved for 2 classes, 4 classes, and 8 classes vary drastically.

It is also reasonable to conclude for the dataset using a combination of SVM with Word2Vec as a feature extractor provides us the best results. Followed by SVM classifier in conjunction with LSA and KNN classifier with Word2Vec respectively. We also noticed that KNN with CountVector gave pretty bad results.

FUTURE WORK

As it is quite evident the biggest issue is the decrease in accuracy as the number of classes increases. So for the future, we may want to train on an even larger dataset using machine learning models. This is because given a large enough dataset machine learning models like ANNs, CNN, etc. may be able to learn various patterns which may help them better predict the ratings for a review.

REFERENCES

[1] Maas, Andrew, et al. "Learning word vectors for sentiment analysis." Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. 2011.

[2] Yessenov, Kuat, and Saša Misailovic. "Sentiment analysis of movie review comments." Methodology 17 (2009): 1-7.

[3] K. L. S. Kumar, J. Desai and J. Majumdar, "Opinion mining and sentiment analysis on online customer review," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, 2016, pp. 1-4, doi: 10.1109/ICCIC.2016.7919584.

[4] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[5] Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).

[6] Deerwester, Scott, et al. "Indexing by latent semantic analysis." Journal of the American society for information science 41.6 (1990): 391-407.

[7] Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. "An introduction to latent semantic analysis." Discourse processes 25.2-3 (1998): 259-284.

[8] Kibriya, Ashraf M., et al. "Multinomial naive bayes for text categorization revisited." Australasian Joint Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2004.

[9] Moraes, Rodrigo, JoãO Francisco Valiati, and Wilson P. GaviãO Neto. "Document-level sentiment classification: An empirical comparison between SVM and ANN." Expert Systems with Applications 40.2 (2013): 621-633.