Sentiment Analysis on Movie Reviews

Liuchang Li
liuchangli@ucla.edu

Di Ma
scarlettmd@ucla.edu

Kaan Ege Ozgun
kaanege@ucla.edu

Ziye Xing
xingziye@ucla.edu

Abstract

Sentiment analysis is a fundamental yet a critical problem in natural language processing. It involves multiple important theories and common techniques that are widely used in text mining. With the development of machine learning methods, there are more emerging approaches that could be applied in sentiment analysis to improve the judgment accuracy. In this project, we propose to investigate several state of art techniques, assess their performance, comparing them with the baseline methods.

1 Introduction

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee [2]. Based on their research, we now want to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive which are assigned numeric value of 1-5. In this project, we want to implement different kinds of classification models (including but not limited to SVM, Nave Bayes Classifier, LSTM RNN, Random forest and SGD classifiers) as well as compare the prediction accuracy of each model to have a better understanding of each models performance.

To be more specific, the definition of our problem is to analyze the movie reviews sentiment from the Rotten Tomatoes dataset. Given a dataset that reviews are labeled with tags which indicates attitude of this review. The labels range from 0 to 4, whose sentiment changes from negative to positive. Our goal is to create a model based on that dataset to predict any movie reviews sentiment and classify them with the closest labels to the groundtruth.

2 Methods

2.1 Naive Bayes

Naive Bayes Classifier is one of the commonest probabilistic models applied to text mining and sentiment analysis in earlier days. It takes a simplified yet powerful assumption that each word has independent probability given the class. Thus, the Naive Bayes Classifier does not consider the position of words and take the bag-of-words assumption. With these two assumptions, we can simply estimate by counting the words and its likelihood in each class together with this class prior probability. In more details, in this movie review sentiment analysis task, the model just need to know how many times a word appears in the reviews given that the reviews are all belongs to one class, and what the probability of this class is tagged for a review.

Due to its simplicity and efficiency, we decide to use Naive Bayes Classifier as one of our baselines model to evaluate. However, for the same reason, Naive Bayes Classifier has very limited performance in various tasks because of these strong assumptions. We will also introduce more approaches and models that has better performance in sentiment analysis task.

2.2 Random Forest

One other approach is using random forest to do the sentiments analysis. Random forest is considered to be a powerful machine learning algorithm due to its accuracy and robust. The reason for that is, Random Forests brings the concept of combining multiple decision trees. While dealing with the single tree classifier there may be the problem of noise or outliers which may possibly affect the result of the overall classification method, whereas

Random Forest is a type of classifier which is very much robust to noise and outliers because of randomness it provides.

In order to get a better performance, we are not satisfied with the basic random forest model, we still want to tune the hyper parameters of random forest. it deals with multiple number of hyperparameters which are: Number of Trees to construct for the Decision Forest, Number of features to select at random, Depth of each trees.

All these hyperparameters are required to be set manually which will be time consuming and does not guarantee that it will give good results for the parameter that we have set manually. Each of the hyperparameters have their own importance and influence towards the output prediction. First hyperparameter is Number of Trees in the forest, increase in number of trees linearly increase accuracy of the model. Larger the size forest better the accuracy, but the accuracy will not be changed at certain level when even there is an increase in number of trees. Number of features also plays a very important role in classification. Random forest does not work on all the features but instead of that there are two values of features which are very famous in the literature [13] [14]and they probably may provide good accuracy results compared to other values of features, but it is worth trying random forest with other values for selecting features at random. Depth of tree is also a very critical hyperparameter in random forest, if smaller value is been selected for Depth then model will suffer from under fitting.

Random forest classifier with hyperparameters values for number of trees 400, number of features at random 3, depth value was set to unlimited and that provided classification accuracy of 0.63. Though different values of hyperparameters are tried manually for each iteration and based on the accuracy returned in that iteration hyperparameters values will be updated for the next iteration. However we have mainly focused on two hyperparameter that is number of trees and number of features. In which increase in number of trees linearly increases accuracy up to certain values and after that there will not be any drastic change in accuracy results.

2.3 Support Vector Machine

Support Vector Machine has been showing that it works well for many tasks in classification problems. Before the Neural Network's recent success in Machine Learning, SVM was considered for long as the standard and powerful model to solve the Machine Learning problem, and many practices and applications have been developed since its promising results for many tasks. More techniques were introduced to improve its performance. Its more advanced techniques such as soft margin classification and non-linear kernel mapping made it successful in many non-linear separable and high dimension data space. Language model happens to be one of those kinds of problem we are trying to solve. It will be a reasonable choice for this task to solve by SVM as its one method to be comparing to others.

In this project, we are dealing with a multi-class classification problem that is categorizing each phrase of a movie comment or review by its sentiment representation, from negative to positive, as 1 towards 5. Most of the implementations of SVM use one-against-one approach to classify a instance, which means 10 classifiers will be trained for 5 tags classification. Adopting this method also avoid heavy imbalanced problems that could be an issue if one-against-rest strategy is used. For simplistic reason and easy comparison to other models, we use bag-of-word model for language and feature model for SVM, other word embedding methods and feature selections could potentially have better result for SVM model. We will train and evaluate its result by using cross-validation process.

2.4 LSTM-RNN

Recurrent neural networks (RNNs) plays a vital role in natural language processing. It takes an input as a sequence of symbols, where at each time step a simple RNN unit is applied to a single symbol, as well as to the networks output from the previous time step. But it also has some kinds of shortcomings including its inability to deal with long-term dependencies. So the LSTM model was developed as an improvement over the standard RNN and it is the major model that we adopted today since its performance has been verified.

Starting from the text input, we did some preprocessing work to the input and successfully tokenized them into word vectors. Then, in the model construction part, the entire model contained an embedding layer with 128 dimensions to project each tokenized word id into a vector space, fol-

Figure 1: Illustration of the LSTM model.

lowed by s dropout layer which helps to resolve overfitting issue by randomly removing similar data. Then we added a LSTM layer with a batch size of 128 followed with the Dense layer, which was a standard and linear neural net layer that reshapes the vector into a fixed dimension (in our case it is the length of unique label size). Last but not least, an Activation layer was added to tell the network how to judge when a weight for a particular node has created a good fit.

Due to the limitation of the datasets, there is no better way to test the performance of the model since no properly labeled test dataset was available (the test dataset we obtained is only containing the phrase and phrase id). So, we first tried to use the Twitter Sentiment Analysis Dataset from ThinkNook which contains 1,578,627 classified tweets, each row marked as 1 for positive sentiment and 0 for negative sentiment. Due to the large size of the dataset (over 120MB), a split was conducted to the datasets into 4 parts and only use of it. The evaluation is based on the percentage of positive label (4 and 5) and negative label (1 and 2). However, the neutral label 3 cannot be attributed to neither category and therefore this evaluation result is no proper.

So we adopted the K-fold cross validation to judge the accuracy of the model constructed. We used a simple 2 layers neural network model as the baseline and implemented the LSTM and Bidirectional LSTM and adjusting the parameters based on accuracy. The unidirectional LSTM only preserves information of the past since the only input it can access is the past. But using bidirectional will also run inputs backwards so that we can preserve information from the future and past using the two hidden states to combine the preserved information. The performance of LSTM witnessed a huge improvement over the normal NN model. And the bidirectional LSTM improves on the LSTM model with a 3 percent increase.

To better improve the accuracy, we adopted some techniques in the preprocessing step to help clean the text. For instance, we used the stemmer tool from NLTK to remove morphological affixes from words and retain only the word stem. We also introduced the stop-word corpus from NLTK to filter out useless data. When converting the word to vector id, we set the normalize word id length to be $\sigma + 2 * \mu$, where μ is the arithmetic mean and is the standard deviation since that if a data distribution is approximately normal then about 68 percent of the data values are within one standard deviation of the and about 95 percent are within two standard deviations. In this way, when we pad the tokenized id to a fixed length we could avoid losing any important information.

3 TRAINING DETAILS

3.1 Feature Extractions

Features are extracted from the dataset before classification. In this part, we used two different approaches. Both approaches are based on Bag of Words model. As it proves its success on Topic Classification, we wanted to test this model for Sentiment Analysis. In Bag of Words model, first we created a vocabulary. The vocabulary consists of tokens obtained from preprocessing of phrases from training and testing dataset.

Preprocessing of phrases is done as follows:

- All words(tokens) are converted to lower case
- Punctuation marks are removed from the phrase.
- The phrase is splitted into tokens(words) from whitespace characters.
- Tokens consists of only numbers are removed.
- Stop words are filtered with the help of stop word dictionary of NLTK software library.
- Each token is replaced with its word stem with the help of `SnowballStemmer` library of NLTK.

After preprocessing step, vocabulary is built from these tokens. After this point, we followed two different approaches.

In the first approach, for each phrase in the training dataset, we create a vector of size of vocabulary which counts the token count of current phrase for all the tokens in the vocabulary.

Then these vectors are used as the classical Bag of Words features.

The second approach makes use of polarity scores of words obtained from publicly available SentiWordNet lexical resource. This resource has a large vocabulary of words with their meaning, and positive and negative polarity scores. Main challenges of using this corpus are that considerable amount of data has 0 positive and negative polarity scores, and the words that are in the main dataset may not be included in this corpus. For these reasons, we followed a method where word counts in the Bag of Words features are multiplied by a score if they have nonzero polarity scores in SentiWordNet corpus. This score can be formulated as

$$score = exp(|posscore - negscore|) \quad (1)$$

4 Evaluation

In this section, we will present the experiment result and show the comparison between some of them to evaluate the performance of each model. First, before we construct and train our models, let us first look through the data we are going to use in the training process. The dataset provided for this competition has been split into train and test set which is preserved for the purpose of benchmarking. Each review comment of movie review has been parsed into phrases by Stanford parser. There are around 8500 sentences while there are more than 156,000 phrases. According to our experiment, the length of those phrases are mostly below 50 words, where 10 is the mode for the length of phrase that is at least 4500 phrases are length of 10.

In terms of label and balance issue, we collected and counted the class label in the training dataset. Roughly speaking, the sentiment labels are fitting follow the normal distribution. Most of the reviews are labeled by 3, which is neutral comment. More precisely, more than 51% reviews are neutral sentiment. This can also be considered as the average of the normal distribution. There are around 30,000 Label 2 and label 4 reviews, and even less label 1 and label 5 comments. There are less than 10,000 reviews are labeled as absolute negative or positive.

Table 1 shows the accuracy we have achieved for all the different models we tested. Comparing to other baseline methods, we could clearly observe that the LSTM-RNN shows the dramatically better result than others.

Model	**Best Accuracy**
Naive Bayes	57.63%
Random Forest	62%
SVM	59.89%
polarized RNN	62.88%
LSTM-RNN	63.07%

Table 1: Best Accuracy for all Models

5 Conclusion

In conclusion, for this movie review sentiment analysis task, we have tried to commit different Machine Learning methods to solve this sentiment analysis challenge and compare their results. For the approaches we have used, including Naive Bayes, Random Forest, SVM, and RNN with LSTM, we found that the RNN has shown the most promising result according to our configuration. Due to the limitation of computing resource we have, we believe that with more training data and longer training epoch, the RNN model will converge to an optimal point that potentially produce a better result.