Movie Review Sentiment Analysis

Hanlin Zhang
School of Information and Library Science, University of North Carolina
hanlin.zhang@unc.edu

Jingyi Zhou
School of Information and Library Science, University of North Carolina
jingyz@live.unc.edu

Qishun Wang
School of Information and Library Science, University of North Carolina
qishunw@unc.edu

ABSTRACT

This paper is our report for the final project in COMP 562 at UNC-CH. In this paper, we have compared the performances of multiple statistical models and neural network model on sentiment analysis. The dataset we used is a collection of movie reviews retrieved from the Rotten Tomatoes website by Pang and Lee[2], and is also available on Kaggle [1]. The code and results are openly available at our Git-hub repository.[2]

Keywords
Sentiment Analysis, Neural Networks, LSTM-CNN

1. INTRODUCTION

Statistical learning is a new way of looking at the world surrounding us, it is new in a sense that the growing computational power has enabled advanced analytics to be performed in a relatively short period of time, such as the artificial neural networks. Machine learning has inspired generations of people in different ways, and in the future, valuable insights may come from data science, analytics and artificial intelligence. How these insights are generated, as well as conveyed and understood by people, should be thoughtfully considered and developed. In this project, we are going to develop an understanding from the Rotten Tomatoes dataset by classifying the sentiment of sentences, and we are also going to explore the differences between the linear model and the non-linear model. We hope to unveil new knowledge from the data and to communicate in ways that positively impact people and the world they live in.

2. LITERATURE REVIEW

Sentiment analysis is helpful in understanding inherent perception about a particular topic and sentiment can be positive, negative or neutral. With the increasing exposure of social media, on where people share about their lives, emotions and opinions, everyone relies more on these contents for making decision, therefore, sentiment analysis, or in other words, opinion mining becomes a popular research problem (Liu, 2012) [1]. There are different methods of calculating sentiment scores, e.g., Bing Liu, Lexicon, so as different dictionaries, e.g. Harvard General Inquirer Sentiment Dictionary and Grahams Moral Foundations Dictionary. According to Turney (2002) [3], movie reviews is the most difficult of several domain for sentiment analysis, with a reporting accuracy of 65.83%.

3. METHODS

This section summarizes the methods in our project, including the models, text processing strategy and hyper-parameters tuning methodology.

3.1 Baseline Models

We created a multi-class classification problem from our dataset. We trained four traditional classification models on the dataset as baseline classifiers, including logistic regression, linear support vector machine, Multinomial Naive Bayes, and Adaboost. It was found that logistic regression and linearSVC achieved relatively higher accuracy among these models.

3.1.1 Text Processing

In our baseline models, we used lemmatization and tokenization functions in Python nltk package to preprocess the data. And we found without other preprocessing methods such as stopword and punctuation removal, the model could have a better performance. After that, we utilized TfidfVectorizer to transform text into a meaningful vector of numbers. Term Frequency-Inverse Document Frequency and is a very common algorithm to transform text. The technique is widely used to extract features across various NLP applications.

3.1.2 Logistic Regression

We first used logistic regression on the features to provide a baseline. In this multiclass case, we set the multi_class parameter to 'ovr' to use the one-vs-rest (OvR) scheme on the training algorithm. And to use a coordinate descent algorithm that solves optimization problems by successively performing approximate minimization along coordinate directions or coordinate hyperplanes, we set 'liblinear' as the solver.

3.1.3 Linear Support Vector Classification

SVMs separate classes by finding a hyperplane that maximizes the margins between class support vectors. When classes are not linearly separable, kernels can be used to find the hyperplane that best separates the data in a higher di-

[1]`https://www.kaggle.com/c/movie-review-sentiment-analysis-kernels-only/kernels`
[2]`https://github.com/hanlin-zhang/COMP562-Final-Project`

mensional space. Simply put, SVMs minimize a regularized cost function with hinge loss. We used sklearns LinearSVC functions to implement this classifier. Similar to SVC with parameter kernel=linear, but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

3.1.4 Hyper-parameters Tuning

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm (Wikipedia, n.d.). During the model fitting step. we also explored the changes in performances on tuning parameters. After fitting a Linear SVC with default parameters, we performed a Grid Search with 5-fold cross-validation to find some possible combinations of parameters that produce the best overall result. We've also manually set the 'dual' parameter in $sklearn.svm.LinearSVC$ to *'False'* based on the recommendations from sklearn documentations because of the n_samples > n_features in the data set. The search space for our parameter search are, i.e. tuned_parameters equals to:

$$\bar{\{}'penalty' : ['l1','l2'],'tol' : [1e{-}3, 1e{-}4, 1e{-}5],'C' : [1, 10, 100]\}$$

3.2 Neural Network Model

3.2.1 CNN

Convolutional Neural Networks (CNN) are networks initially created for image-related tasks that can learn to capture specific features regardless of locality. For a more concrete example of that, imagine we use CNN to distinguish pictures of Cars vs. pictures of Dogs. Since CNN learn to capture features regardless of where these might be, the CNN will learn that cars have wheels, and every time it sees a wheel, regardless of where it is on the picture, that feature will activate. In our particular case, it could capture a negative phrase such as "don't like" regardless of where it happens in the movie comments.

3.2.2 LSTM

Long short-term memory (LSTM) are a type of network that has a memory that "remembers" previous data from the input and makes decisions based on that knowledge. These networks are more directly suited for written data inputs, since each word in a sentence has meaning based on the surrounding words (previous and upcoming words). In our particular case, it is possible that an LSTM could allow us to capture changing sentiment in a comment. Some comments have words with conflicting sentiments that would end-up confusing a simple Feed-Forward network. The LSTM, on the other hand, could learn that sentiments expressed towards the end of a sentence mean more than those expressed at the start. And we choose bidirectional LSTM. BiLSTM will run the inputs in two ways, one from past to future and one from future to past and what differs this approach from unidirectional is that in the LSTM that runs backwards you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

3.2.3 Pre-Trained Word Embeddings

Word Embedding is a very popular method of describing words in a dictionary by mapping them into n-dimensional fields. The actual methodology varies depending on the author. In our case we tried different pre-trained word embeddings to process the input. According to the result, we finally decided to use 300-dimensional pretrained FastText English word vectors released by Facebook[3].

3.2.4 LSTM-CNN

We tried CNN and LSTM independently, and finally decided to combine them together and use a LSTM-CNN model. Our LSTM-CNN model consists of an initial BiLSTM layer which will receive word with fasttext embeddings for each token in the comments as inputs. The intuition is that its output tokens will store information not only of the initial token, but also any previous tokens; In other words, the LSTM layer is generating a new encoding for the original input. The output of the LSTM layer is then fed into 3 convolution layers which we expect will extract local features. Finally, the convolution layers output will be pooled to a smaller dimension and ultimately outputted as one of the 5 sentiment labels. Figure 1 and Table 1 are the structure and parameter of our model.

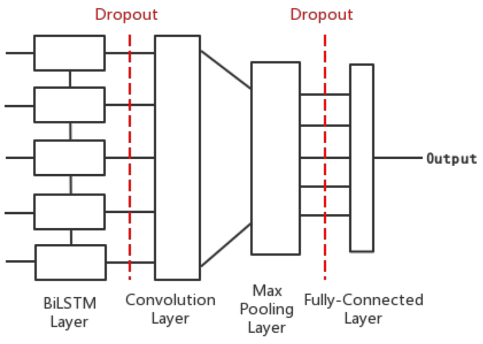

Figure 1: LSTM-CNN Model.

Table 1: Parameters of the Network

Parameter	
Word Embedding	FastText
Embedding Dimension	300
Epoch	15
LSTM Units	50
Batch Size	128, 256, 512
Kernel Size	1, 3, 5
Activation	Relu
Dropout	0.2

4. RESULTS

This section presents the results of our project, including prediction results (measured by accuracy), hyper-parameters tuning results and data visualizations.

[3]`https://www.kaggle.com/yekenot/fasttext-crawl-300d-2m`

4.1 Models Results

Table 2 displays how our models behave on the measurement of accuracy. It was found that MultinomialNB Classifier and Adaboost Classifier could achieve no higher than 60% accuracy. Logistic Regression and LinearSVC achieved about 63% accuracy.
Our neural network model achieved an accuracy of over 67.5%, which is much higher than all the other traditional models. And among all the neural network methods we tried, The LSTM-CNN structure seems to be the best logically because its initial LSTM layer seems to act as an encoder such that for every token in the input there is an output token that contains information not only of the original token, but all other previous tokens. Afterwards, the CNN layer will find local patterns using this richer representation of the original input, allowing for better accuracy.

Table 2: Accuracy of Models

Model	Accuracy
Random Guessing	0.20000
AdaBoostClassifier	0.54793
MultinomialNB	0.57554
Logistic Regression	0.61462
LinearSVC	0.62822
*LSTM/CNN	0.67588

4.2 Hyper-parameters Tuning Result

The best parameter we've found by Grid Search:

$$\{'C' : 1, 'penalty' : 'l1', 'tol' : 0.0001\}$$

Surprisingly, the best parameter combinations, in this case, are exactly the same as the default setting, i.e. there is no new information gained by doing the Grid Search and the parameters remain the same as the default. So the improvements on the accuracy score is not significant.

4.3 Data Visualization

We also visualized the confusion matrix in order to look into the black box.Figure 2 is the confusion matrix for Logistic Regression, and Figure 3 is for the Linear SVM. In confusion matrix, the diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions. In our case, although the prediction accuracy for Logistic Regression and Linear SVC are close (0.61462 vs 0.62822), but the classification results are slightly different: Logistic Regression is more confused between label 0 and label 2, but Linear SVM is much better on this. Further, we also noticed that correctly classify label 1 is the most challenging task in this project, since both Logit and SVC model failed to classify it correctly.

4.4 Futher Results

4.4.1 *Learning Rate*

We found that the optimal number of epochs to train our models was around 10. However during our search, we realized that each model had different learning rates. In other words, some models would learn and start over-fitting

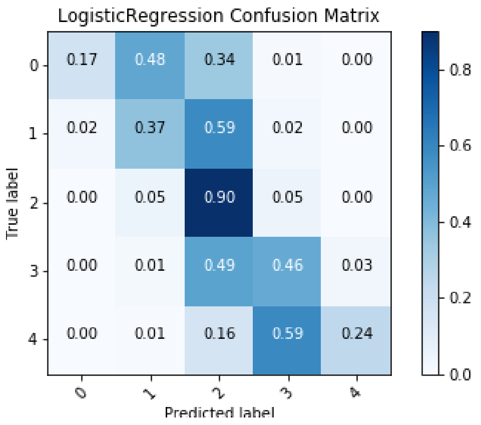

Figure 2: Logistic Regression Confusion Matrix.

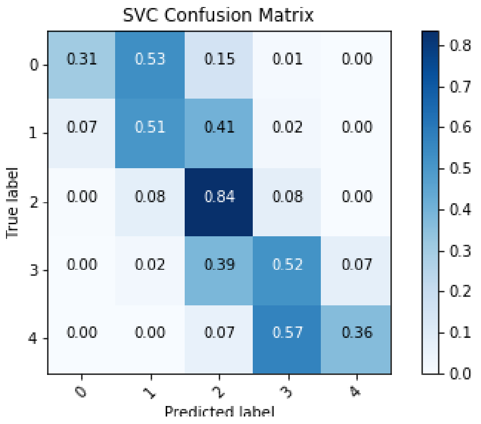

Figure 3: Linear SVM Confusion Matrix.

quicker than others. training the LSTM-CNN model with just a few epochs was enough to get accuracy above 60%. However, too many epochs would affect it substantially.

4.4.2 *Effects of Dropout*

In our original implementation, we included dropout layers to test if that would increase performance. On the LSTM-CNN layer we added the dropout layer after then CNN layer, before feeding into the fully connected layer. We found that Dropout was extremely important for these models to work properly and the best dropout rate in this case is around 0.2.

5. CONCLUSIONS

In this paper, we have looked at the Rotten Tomatoes reviews dataset and sentiment analysis. We have fitted both baseline model and neural network model in order to predict the sentiment score. Our research have demonstrated that a well-trained neural network model performs better than baseline models when it comes to sentiment analysis. But the overall advantage of neural network model over baseline models is not very significant, therefore we have proposed that future researches on the topics related to sentiment analysis and artificial neural networks can benefit form: 1) find the most optimal learning rate; 2) come up with a better drop out strategy.

6. ACKNOWLEDGEMENTS

We want to acknowledge Dr. Jorge Silva, Ric Poirson and Svetak Sundhar for this semester's lecture on the topics related to Machine Learning. Thanks to their hardworking, this semester is both fun and challenging for all of us.

7. REFERENCES

[1] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[2] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

[3] Tirath Prasad Sahu and Sanjeev Ahuja. Sentiment analysis of movie reviews: A study on feature selection and classification algorithms. *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*, pages 1–6, 2016.