LIN 373 Machine Learning Toolbox for Text Analysis // Project Final Report: Sentiment Analysis on Movie Reviews

By: Tan Huey Qing & Sarang Rastogi

15 May, 2020

1. INTRODUCTION

Sentiment Analysis is the procedure to predict the inclination of people's opinions, to determine if a chunk of text is positive, negative or neutral.[1][2] Natural language processing and machine learning techniques can be combined to assign sentiment scores to the text. Sentiment analysis is a very helpful technique to gain insight on a target audience's attitude towards a certain topic. This is extremely useful for business organisations to gauge their performance and how their customers think of their competitors. Customer satisfaction can also be revealed quickly with sentiment analysis.

Here in our project, we aim to correctly classify the sentiment of movie reviews and to analyze the performance of different classifiers on this type of data. Our hypothesis is that a neural net(NN) or a support vector machine (SVM) with an embedding layer would perform the best out of our other models, naive bayes and logistic regression with cross validation. We perform sentiment analysis on a dataset containing 50,000 reviews from IMDB, half of which are classified as positive, and the other half being negative. The motivation behind this project is that we want to gain a deeper understanding of how different models perform, so that we can work towards our end goal to predict how trends in the stock market change with relation to the general sentiment of companies, which is a much more complicated topic. We attempted to do that previously, but realized we did not have enough knowledge on sentiment analysis at that point of time.

2. RELATED WORK

Our project idea is inspired from a Stock Prediction Using Twitter Sentiment Analysis project paper published in 2014 by 2 students at Stanford University [3]. The paper focuses on predicting stock market movement based on the general sentiment of a population. Following this paper, we will be implementing our sentiment analysis models. Other examples of application of sentiment analysis are twitter sentiment analysis to visualise United Airlines' PR Crisis [4] where one can analyse the negative sentiment among the public due to bad actions of the company. Thus, sentiment analysis is widely used in projects as mentioned above to gauge brand reputation or predict decisions of the public based on sentiments which is then used to make important business decisions.

3. DATASET

In this project, we used an IMDB dataset containing 50k different movie reviews to carry out binary classification. [5] The data in this dataset is highly polar and contains an equal amount of positive and negative reviews, which could have possibly impacted the performance of our models positively. The data we obtained as mentioned above was not very clean, and had to be preprocessed by us. First of all, we encoded the sentiment column of our data such that positive labels were labeled 1; negative labels were labeled 0. Then we had to make sure all words were converted to lowercase, and then stem them to bring them to their base form and reduce the inflections and variations of words. This prevented the vocabulary size from getting unnecessarily large. We also noticed that when people write reviews, they tend to use rather colloquial ways of speaking rather than following the normal rules of language or formal speech. This included unconventional "words" such as "a$$" and other fanciful swear words in addition to some leftover markdown code like "
" as shown below.

I will never get back the three hours of life this film has stolen from me.

The film is basically a psychedelic drug trip disguised as an important creative process. I'd love to know what they were on when this film was being made.

Its also the most historically-inaccurate film in existence; 2001 has come and gone without any of the events or predictions taking place.

Characters are unlikeable, design is simplistic and everything just rambles on without any sense or logic to it.

And the ending is probably the worst of it: its supposed to be thought-provoking but the only thought that entered my mind is "What the F$*K is going on?!"

I'd say for anyone looking for serious entertainment purposes, AVOID this film at all cost and choose a sci-fi movie that ISN'T stuck up its own @$$.

This prevented our models from recognizing them as words and had to be removed or

stemmed. All of this was done with a preprocessing function provided by gensim.

Upon exploring our data even more, we made an analysis on the length of the reviews and obtained a distribution as shown above.. The mean obtained was approximately 1309 words per review, which seems like a good amount.

4. METHODOLOGY

In this project, we decided to implement Naive Bayes and Logistic Regression models as our baseline, and use Neural Nets and SVM to give us better results.

- **Naive Bayes:** We decided to implement the naive bayes as our baseline classifier. As the name suggests, the classifier is known to make the assumption of the independency of words:

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

As we felt that only accuracy might not be the perfect result measure, we also plotted the roc-curve (which plots the true positive rate against the true negative rate). This model gave us an accuracy of 84.15% and of 84.99% with a 10 fold cross-validation along with an area under the curve of about 0.91.

- **Logistic Regression:** To move away from this 'naive' assumption we implemented the logistic regression model which splits feature space linearly, and typically works reasonably well even when some of the variables are correlated:

$$log\ p(x)/\ (1 - p(x)\ ) = \beta 0 + x \cdot \beta$$

On plotting the ROC curve we got an improved area under the curve of about 0.94.

- **Neural Net:** To build our neural net, we used Keras, It allowed us to experiment with neural nets rather easily as we did not have to implement every single layer by ourselves. We used the Sequential Model API, which essentially groups a linear stack of layers into a Keras model and also provides training and inference features on that model. [6] We first added a Dense layer, and gave it an output shape of 10, used the relu activation function, with the input dimensions of our training data. The reason why we only give the input dimensions in the first layer is because the other layers can then infer the shape automatically. We then added a second Dense layer with an output shape of 1 and used the sigmoid activation function before configuring the learning process with the .compile() method. To prevent overfitting we added a regularization layer in between with a drop out rate of 20%. This model gave us an accuracy of 89.69%

- **Word2Vec:** Since we were using a rather large dataset with use of unconventional language, we refrained from using pre-trained word embeddings such as GloVe as trained our own word embedding which helped us prevent overfitting. Our Neural Net model seemed to be performing only slightly better than our baseline, so we decided to add an embedding layer to our Neural Net using Gensim in order to test if performance is improved. We implemented the embedding using the Gensim Word2Vec module and decided to save the model in .txt format for convenience in terms of loading and reusing it. To check whether our word2vec model was accurate we conducted word similarity analysis of words.

- **Neural Net with Embeddings:** To do this, we build the exact same neural net model as before, but added an embedding layer and a pooling layer before the other layers. We created an embedding matrix such that the row id's correspond to the one-hots in our Tokenizer's word_index and the columns correspond to that word's embedding. This matrix is to be used in the embedding layer. After that, we added a GlobalAveragePoolingID layer which helps to minimize overfitting by reducing the total number of parameters in the model. Then, we trained the model the same way as above. This model gave us a surprisingly low accuracy of 50.54%.

- **SVM:** We decided to implement support vector machines to study the concepts of dimensionality reduction and PCA on our model and word embedding, however due to low computational power SVMs took surprisingly long to run. A basic SVM model with the default rbf kernel and a 2 fold cross-validation took a wall time of approximately 2 hours to run and gave a much lower than expected accuracy of about 72%. On trying to implement grid search in order to

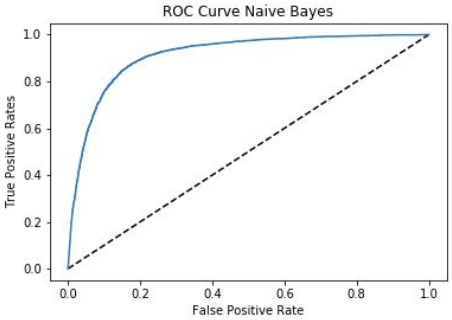
find the best parameters for our SVM model, the code kept running for about 14hrs with no output. Due to this issue we were not able to implement an advanced model.

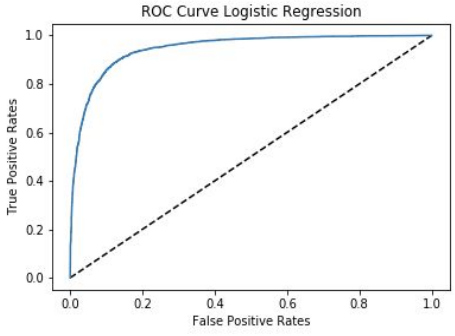
5. RESULTS

Model Used	Accuracy	Other measures	Motive
Naive Bayes	84.15% 84.99% (cross val)	ROC Curve area: 0.91	Baseline classifier
Logistic Regression	88.15% 88.34% (cross val)	ROC Curve area: 0.94	To overcome indepence of words assumption
Neural Net	89.69% 50.54% with embedding	Epoch validation accuracies: 1,6: 49.35% 2-5,7-10: 50.65%	To implement high performance classifier
Support Vector Machine	72.24%	-	To study the effects of dimensionality and PCA

As seen in the table above, our models mostly behaved as our hypothesis. However, it was shocking that our neural net model with an embedding layer performed so badly. We shall go into detail about that soon.

- **Naive Bayes and Logistic Regression:** To try out another measure of performance we implemented ROC Curves for these classifiers. Both the Naive Bayes and the logistic regression classifiers performed really well on the data giving an area under the curve of 0.91 and 0.94 respectively:

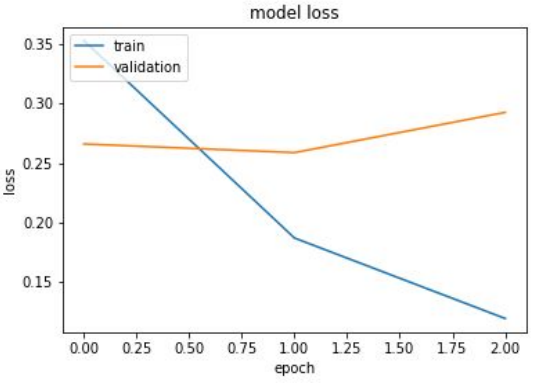


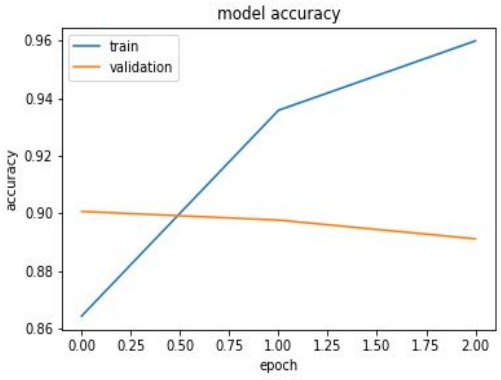


- **Word Embeddings:** Below are the results of the word similarity test performed on our 2 embedding models. One with only 100 reviews sampled at random, and the other with all 50,000 reviews we have. Clearly, we needed our entire dataset to obtain the most accurate word embedding model. The words may appear slightly weird due to stemming.

	100 reviews	50,000 reviews
'funni' top 4	work, girl, end, young	hilari, funnier, amus , hyster
'director' top 3	sea, screenplai, turn	filmmak, cinematograph, direct
'food', 'bad' top 3	not found	toilet, beer, suck
'woman', 'king'	michael, bella, pseudo	queen, princess, jerol

- **Neural Net:** As expected, our neural net model had the best performance out of all the models. However, we expected a larger jump than just 1% from logistic regression. We initially trained our model for 10 epochs, but noticed it was being severely overfitted. After some tuning, we then proceeded to train it for 3 epochs, and decided to use a batch size of 50. That seemed to be the optimal batch size before the model started to degrade, which also seemed like a reasonable amount considering how large our dataset was. This landed us with an accuracy of 89.69%. Our results were rather unusual, as it performed the best at the very first epoch, even after implementing a drop out rate of 20%.





We thought we could improve this model by adding an embedding layer, but the results were quite uncanny as we ended up with an accuracy of 50.54%. It took approximately 3 hours to run 10 epochs, and the accuracy and loss values just kept fluctuating between 2 different values for every epoch. This pattern caused us to assume increasing the number of epochs will not improve this model any further, and it was difficult to test because it took so long to run. We think this could possibly be due to an issue with our word embedding, even though our embedding model seemed rather accurate based on the results above. A dataset of 50k reviews may seem very large to our unprofessional eyes, but it could still be possible that it was not large enough to prevent overfitting.

- **SVM:** As stated in the methodology section, the only SVM model we could implement gave us an accuracy of 72.2% which is quite lower than our baseline model of naive bayes. From this and along with the issue of very large run time we inferred that our vocabulary size (93033 words) was rather large for the kernel to handle the process of dimensionality.

6. CONCLUSION

- When we initially began the project, we expected the Neural Net and SVM combined with word embeddings to be the best performing model, however the addition of word embeddings to our Neural Net model significantly decreased the accuracy thus delivering results quite different from our expectation. Moreover, our simple classifier performed way better than our expectations which could be attributed to the fact that movie reviews in our dataset were highly polar and equally distributed as positive and negative. This combination of expected and unexpected results gave the opportunity to experiment with our models in terms of parameter tuning and other measures providing us deeper insight into the working of every classifier and setting us on the path of working on more complex projects in the future.

7. WORK DISTRIBUTION

- Huey: Presentation Slides, Neural Net model, Word2Vec embedding, Report
- Sarang: Project Proposal, Naive Bayes, Logistic Regression, Support Vector Machines (SVM), Report

8. ACKNOWLEDGMENTS

We would like to thank our professor, Jessy and our teaching assistant, Laura for all their guidance and support throughout this semester, and for being so understanding and patient.

9. REFERENCES

- 1. What is Sentiment Analysis? - Definition from Techopedia [Internet]. Techopedia.com. [cited 2020May10]. Available from: https://www.techopedia.com/definition/29695/sentiment-analysis
- 2. Bitext. What is Sentiment Analysis and How Does it Work? [Internet]. What is Sentiment Analysis and How Does it Work? [cited 2020May10]. Available from: https://blog.bitext.com/what-is-sentiment-analysis-and-how-does-it-work
- 3. Mittal A, Goel A. [Internet]. [cited 2020May10]. Available from: http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf
- 4. Step-by-Step Twitter Sentiment Analysis: Visualizing United Airlines' PR Crisis [Internet]. iPullRank. 2019 [cited 2020May10]. Available from: https://ipullrank.com/step-step-twitter-sentiment-analysis-visualizing-united-airlines-pr-crisis/
- 5. N L. IMDB Dataset of 50K Movie Reviews [Internet]. Kaggle. 2019 [cited 2020May10]. Available from: https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
- 6. Team K. Keras documentation: The Sequential class [Internet]. Keras. [cited 2020May12]. Available from: https://keras.io/api/models/sequential/