Twitter Sentiment

Austin Wilson

SJSU

austin.wilson@sjsu.edu

Paul Nguyen

SJSU

paul.t.nguyen@sjsu.edu

**ABSTRACT**

The ability to understand public opinion is important to any product. As such, it is necessary to find a way to retrieve the sentiment of those opinions as a foundation. We approached this problem by using Natural Language Processing, Neural Networks and Logistic Regression to obtain sentiment from text data.

# INTRODUCTION

The problem that we are trying to solve is a classical artificial intelligence problem. What is the sentiment of some text value? This topic has been heavily researched and we were able to take advantage of python libraries to train our own custom models. We build a pipeline using both the original text data and pickle objects. This was deployed as a flask application on the cloud. Of the several models that we trained on the same data we decided to deploy the LSTM mode because of its high accuracy and ability to distinguish context.

# RELATED WORK

A project that also uses the same Kaggle dataset that we chose does a similar project. In that project, the author performed a Bag Of Words analysis in order to find the overall sentiment of a particular topic. Unlike the author’s approach where the bag of words tries to understand what features make up positive or negative sentiment, our LSTM/RNN approach distinguished the differences between the positive and negative samples.

# DATA

The dataset used for this project is a generic [Twitter Sentiment Analysis](https://www.kaggle.com/jp797498e/twitter-entity-sentiment-analysis) Dataset from Kaggle. The data itself consists of 4 attributes: Tweet ID, Entity (tweet topic), Sentiment, and Tweet Content. The sentiment column also consists of 4 possible values: Positive, Negative, Neutral, and irrelevant (to the topic in Entity.) Irrelevant sentiments in this context are considered neutral. We received the data as a CSV file and applied multiple transformations upon the tweet context to boost relevancy of the words within the text. Some of the methods include replacing “rt” at the beginning of the text since it indicates that the text is a retweeted quote, lemmatizing so that similar words are reduced to their base forms, and removing stop words that provide no weight to either positive or negative sentiments. Speaking of sentiments, since we aimed to classify our tweets to either positive or negative, we filtered out the sentiments column to match those criteria.

# METHODS

SAMPLE: Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Conference’10*, Month 1–2, 2010, City, State, Country.

Copyright 2010 ACM 1-58113-000-0/00/0010 …$15.00.

DOI: http://dx.doi.org/10.1145/12345.67890

In identifying appropriate approaches to classification of tweets, we decided to go with a couple of neural networks and logistic regression methods in applying Natural Language Processing. All models used for training and deployment are implementations of Natural Language Processing as we are dealing with text data and aim to understand context.

## PREPROCESSING

For the preprocessing we used sklearn and NLTK. We took two approaches. We first used the Natural Language Tool Kit to remove stopwords, lemmatize and tokenize the data. We also saved that as a pickle object for later use. We also used a tf-idf vectorizer from sklearn to preprocess the data for our other neural network models.

## LSTM

We used the tokenized data from the first preprocessing method to train a LSTM model.

## Neural Networks

We also used the tf-idf data to train two neural networks and compare their performance. The first neural network had just one layer with 25 fully connected nodes and relu activation function. The output is 2 layers with softmax activation function for classification. The second neural network has 4 layers with 25,50,25,10 fully connected nodes and all relu activation functions. We have also implemented a logistic regression model.

Table . LSTM Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **Graphics** | **Top** | **In-between** | **Bottom** |
| Tables | End | Last | First |
| Figures | Good | Similar | Very well |

# EXPERIMENTS AND RESULTS

We used a holdout test set to gain some insight into the accuracy of the models we trained. The results are as follows

## LSTM

We found mixed results with the LSTM model. When we first tried it, the model performed well but when we retrained it, we found the f1-score was much lower at .26.

LSTM Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **class** | **precision** | **recall** | **F1-score** |
| negative | .6 | .16 | .26 |
| positive | .49 | .88 | .63 |

## Neural Network 1

With our simple one layer neural network we had good results. The f1-score was .9 and the classification report is shown below

NN1 Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **class** | **precision** | **recall** | **F1-score** |
| negative | .92 | .88 | .9 |
| positive | .87 | .91 | .89 |

## Neural Network 2

Our second neural network performed very similarly to the first one. This was not too surprising considering they were very similar.

NN2 Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **class** | **precision** | **recall** | **F1-score** |
| negative | .92 | .9 | .91 |
| positive | .89 | .91 | .9 |

# CONCLUSION

Overall, we found that the neural networks were the most accurate and decided to move that into production. Our pipeline is designed and implemented into a flask app. The application receives text from a user in the form of form data which is then passed to the flask application. The vectorizer is pre-loaded into the application and transforms the user text into a tf-idf matrix. The model is also preloaded into the flask application. The model then classifies the text and displays the class to the user.

# REFERENCES

1. Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= <http://doi.acm.org/10.1145/161468.16147>.
2. Ding, W. and Marchionini, G. 1997. *A Study on Video Browsing Strategies*. Technical Report. University of Maryland at College Park.
3. Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 01 - 06, 2000). CHI '00. ACM, New York, NY, 526-531. DOI= <http://doi.acm.org/10.1145/332040.332491>.
4. Tavel, P. 2007. *Modeling and Simulation Design*. AK Peters Ltd., Natick, MA.
5. Sannella, M. J. 1994. *Constraint Satisfaction and Debugging for Interactive User Interfaces*. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
6. Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* 3 (Mar. 2003), 1289-1305.
7. Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology* (Vancouver, Canada, November 02 - 05, 2003). UIST '03. ACM, New York, NY, 1-10. DOI= <http://doi.acm.org/10.1145/964696.964697>.
8. Yu, Y. T. and Lau, M. F. 2006. A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions. *J. Syst. Softw.* 79, 5 (May. 2006), 577-590. DOI= <http://dx.doi.org/10.1016/j.jss.2005.05.030>.
9. Spector, A. Z. 1989. Achieving application requirements. In *Distributed Systems*, S. Mullender, Ed. ACM Press Frontier Series. ACM, New York, NY, 19-33. DOI= <http://doi.acm.org/10.1145/90417.90738>.