Machine Learning Nanodegree
Capstone Proposal

Sheng Chien

August 12 2018

1 Domain Background

Sentiment analysis (also called opinion mining) is a method of computationally identifying the underlying emotion from a piece of text. It's important in NLP field as to truly communicate with humans, computers would need to understand human emotions. It is also valuable for a business to know if their customers are satisfied.

Sentiment analysis has a long history dated back to the beginning of 20th century. Sentiment analysis is a fast growing research area in recent years. 99% of papers have been published after 2004 [7]. It shifted from analyzing online product reviews [8] to more social media texts from Twitter [3]. And many topics extend beyond by utilizing sentiment analysis such as stock prediction [6].

1.1 Motivation

One personal motivation to investigate this particular problem is because it could potentially be applied to the company I currently work for, Resonate. At Resonate, we offer real-time consumer intelligence to our customers. Our platform can show you the deep understanding of the true motivations why your customers buy your products. For example, A website might learn their customers are impulse shoppers and don't care about brand names. Or B political campaign might learn their supporters care more about education but not tax reform. With the sentiment analysis, we could further analyze the audience who gave positive feedback to learn their psychographics and what truly motivates them. And perhaps adjust the marketing strategy accordingly.

2 Problem Statement

The problem to solve is to computationally classify a polarized sentiment (positive or negative) behind the expressed words of a movie review. Our objective contains two parts. First is to map a review document d to a fixed-size vector representation of words ϕ_w using a word embedding converter function $g(d)$. The word vector is then mapped to a predicted sentiment label $\hat{y}$ by using a classifier function $f(x)$.

$$\phi_w = g(d). \tag{1}$$

$$\hat{y} = f(\phi_w). \tag{2}$$

It's noticeable that we can also improve word vector ϕ_w to achieve better prediction of the review sentiment.

3 Datasets and Inputs

The dataset chosen is Large Movie Review Dataset which was originally collected by Maas, et al. [4]. It can be downloaded from [2]. The dataset contains large amount of movie reviews from the Internet Movie Database (IMDB). Since there are unbalanced reviews among movies, no more than 30 reviews are allowed per movie. They constructed a collection of 50,000 reviews which are equally split into training and test sets. Each has 25,000 reviews. The original star rating $\{1..10\}$ are linearly mapped to $\{0, 1\}$ (bad or good). A negative review if $rating <= 4$, and a positive review if $rating >= 7$. In addition to the labeled data, an extra unlabeled data of size 50,000 are provided.

4 Solution Statement

There are two main parts of the solution proposed here. The first part is a unsupervised learning to build vector representations of words. The second part is a supervised learning to classify a sentimental movie review.

To be specific, the word2vec model published by Mikolov et al. [5] will be used to perform the unsupervised part. All the reviews in train set will be fed into word2vec model to build the word vectors. A word vector is a fixed-size low dimensional vector, typically 100 to 200. Both labeled and unlabeled train set will be included to make better model accuracy. The trained word2vec model is then used to build a document vector, or feature vector, by averaging of all vectors from each word in a review. The document vector along with the labeled sentiment become the input to the classification algorithm for training. The trained classifier will then be used to make predictions of the sentiments.

XGBoost [1], an efficient gradient boosting algorithm, is chosen for the binary classification task due to its model performance and execution speed. It's a popular algorithm has been recently dominating in Kaggle competitions.

5 Benchmark Model

One benchmark model is to compare with the model by Mass, et al. [4]. The result can be found in Table 2. The measurement for comparison is accuracy, that was used in the paper, to see which model results in more correctly predicted sentiments on the test set. In addition, logistic regression algorithm will be used as benchmark in comparison to XGBoost. The reason is there are two parts in this sentiment analysis process, where the first part is to use word2vec model to learn vector representations of words. It will be interesting to just compare these two classifier algorithms. As logistic regression algorithm is often served as the baseline, we can observe how much better (or worse) performance XGBoost could achieve.

6 Evaluation Metrics

The evaluation metric used to quantify the result is accuracy. Basically, to compare which model can predict most correct sentiments on the test set. It's defined as:

$$accuracy(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} 1 \times (\hat{y} = y) \tag{3}$$

The n is the number of test samples. The y is the truth while $\hat{y}$ is the predicted value. Since the test set is split equally among positive and negative reviews, accuracy is an appropriate measurement as the random guess would only get us 50% accuracy. The model is expected to be a lot higher than the random prediction.

7 Project Design

The workflow to approach a solution will start with data exploration. A chart with word distributions along with top N common words will be useful to get some insights of the data. It could help learn what preprocessing is appropriate and tune the word2vec model accordingly. To establish a baseline, a simple preprocessing from gensim package will be used first. More advanced preprocessing would be considered such as stop words, stemming, etc.

Next comes with the unsupervised training to build vector representations of words from the training data. The default parameters of the word2vec model will be tried first. Different hyperparameters will be tested to tune the model. Some sample word vectors will be explored to gain more insights. For instance, showing the similarity scores for a list of word pairs or similar words.

And we will train XGBoost classifier along with logistic regression classifier for comparison. Again, the default model will be first explored to get baseline accuracy. Grid search with cross validation will be conducted to find optimal performance of classification for the training data.

Last step is model evaluation. Final model accuracy will be obtained by applying the trained classifier on the test set to predict review sentiments. No more tuning will be allowed at this point. The final result will be compared to the benchmark models.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[2] Stanford AI Lab. Large movie review dataset. http://ai.stanford.edu/˜amaas/data/sentiment/, 2012.

[3] Patrick Lai. Extracting strong sentiment trends from twitter. http://nlp.stanford.edu/courses/cs224n/2011/reports/patlai.pdf, 2010.

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *ACL*, pages 142–150. The Association for Computer Linguistics, 2011.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[6] Anshul Mittal and Arpit Goel. Stock prediction using twitter sentiment analysis. http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf, 2012.

[7] Mika Viking Mäntylä, Daniel Graziotin, and Miikka Kuutila. The evolution of sentiment analysis - a review of research topics, venues, and top cited papers. *CoRR*, abs/1612.01556, 2016.

[8] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 115–124, 2005.