Comparison of sentiment prediction techniques on a common movie review data set.

Matthew Smith

***Abstract*—In language, sentiment is a major component of overall expression. Sentiment analysis and classification uses traditional machine learning techniques but learns to predict what the text is trying to express. In this paper we will survey, discuss, and compare various techniques for sentiment analysis. Such techniques include older methods such as the Bag of Words model and more modern techniques such as neural net based word and paragraph vector analysis. These models would be used on a common random forest classification tree. These methods were implemented and tested on the Stanford IMDB movie review dataset. This dataset expresses the star review system as a binary value, where greater than seven is a 1, and less than five is a 0. Intermediate values were thrown out. ¡results¿**

I. Introduction

Were currently living in a world saturated with textual reviews, discussions, and opinions. Sentiment is a crucial datapoint for being able to take these pieces of text and extract a useful understanding of the author's intention. While much of machine learning tends to lean towards predicting what the data represents, sentiment analysis gives an opportunity to understand emotion and opinion behind the data. Sentiment analysis is distinct from subject analysis in the inference phase of training. For both types of analysis, vocabularies are built for the model to have an understanding of surrounding words. These methods split during training for classification, as subject analysis would train using subject ground truth labels and sentiment analysis would train using sentiment labels accordingly. Sentiment analysis is primarily performed on text input taken from sources such as reviews, forums, and comment sections. For this paper, we will be using the Stanford IMDB dataset. This dataset consists of 100k individual reviews, with 50k labeled with either a positive or negative sentiment. This will allow for a large vocabulary to be available for training producing a diverse model to be used on unknown reviews. This dataset will be beneficial due to the medium to large sized paragraphs available, which will be allow for more robust models. In this paper, we will be implementing several models beneficial in sentiment analysis, including Bag of Words, Word vectors, and paragraph vectors. These models will be compared using identical classification trees in a random forest configuration.

II. Discussion on the dataset

In this section we will be discussing the IMDB dataset and its format. Discussion on the data split, the labeled, and unlabeled, train and test data obtained from a kaggle competition page.

Also discussed will be the processing of the data, such as removing any html tags, removing any unwanted punctuation, and splitting the data on whitespace to create an array from which to learn on. Additionally, discussion on what punctuation should be kept and what should be ignored. The Large Movie Review Dataset introduced by mass et al. [1],was built with sentiment analysis in mind. This dataset consists of 100k movie reviews from the Internet Movie DataBase (IMDB). Half of the reviews in the dataset were labeled using Amazons Mechanical Turk. For each sample, if a review gave greater than or equal to seven stars, the review was given a positive label. Likewise, a review with less than or equal to four, it received a negative label. This binary classification labels allow for the task of classification to be easier later on, as it avoids vague moderate reviews and their effect on the learning.

In raw form, the data was separated into 5 directories, each containing many text files of individual reviews. Test and train data were separated into separate directories, and within those, positive and negative reviews were separated as well. In order to make use of these reviews, each directory in the lowest level was concatenated into a single TSV file, with the filename as the ID, the sentiment, and the full review, each separated by tab characters.

Next, the data must be further processed and cleaned. For each review, punctuation, capital letters, and HTML tags were stripped and the clean reviews were written to a new TSV file with the same format. This process was done a second time, but this time removing stop-words from the reviews. Stop words are words which connect together sentences and often do not provide much meaning. These are removed to avoid noise in the models.

III. Survey and comparison of methods

Here we will be an overview of methods to solve the problem of sentiment analysis as well as a comparison of their strengths and weakness.

IV. Method of Test

In this section, we will be discussing the methods that will be chosen for comparison. Also to be discussed is why each was chosen.

When testing to compare these methods, it was important to ensure data was consistent. Certain implementations had taken influence from online tutorials such as [kaggle tutorial] but had datasets preprocessed from the source in varying ways. Each

implementation needed to be modified to accept the processed format defined for this experiment.

First, the Bag of Words (BoW) model was implemented. In order to construct a BoW model, a vectorizer object was defined using sklearns CountVectorizer library. The reviews of the corpus were concatenated into a list and fed into the CountVectorizer fit_transform function to create the vector model that will be used for testing.

Next, the word vector model better known as Word2Vec was implemented. Here, the training corpus was split into an array of sentences. This was then fed into the gensim librarys implementation of Word2Vec. For initial baseline tests, the parameters were given as follows; 300 features, minimum of 40 words in a sample, 4 workers, a context window of 10 words, and a frequent word downsampling of 1e-3 ¡MAKE NICE IN LATEX¿. The model is unsupervised, and therefore could be trained on both labeled and unlabeled reviews. The model was generated and saved for later use.

Finally, the paragraph vector model, Doc2Vec was implemented. Here, more work needed to be done as the gensim implementation requires very specific input. A named tuple list, which behaves in a similar manner to a dictionary, was created to hold lists of words from each review, as well as a tag for the Doc2Vec model to keep track of. This used the full training corpus, labeled and unlabeled in a similar manner to the Word2Vec implementation. Once again this model was saved for later use.

In order to compare the models and not classification methods, the same random forest tree classifier was used. This implementation was taken from a tutorial for BoW and Word2Vec by Kaggle [2]. The random forest classifier will fit a classification tree to the trained model and the corresponding training labels. The test data would then be vectorized and applied to the tree, where sentiment could be predicted. For all tests except one, the random forest used 100 estimator trees. Using this method, all three models could be used to classify in such a way that allows for comparison.

V. Results

VI. Conclusion

Acknowledgment

References

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[2] Bag of words meets bags of popcorn. [Online]. Available: https://www.kaggle.com/c/word2vec-nlp-tutorial