Franklin Thomas
frathom@iu.edu

Abstract

In this paper, I revisit the problem of document classification based on overall sentiment, e.g., classifying the document as a positive/negative review(Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002)). Using the movie reviews dataset, I will be trying to replicate the results of the 'Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002)' paper also try improving upon the results using feature engineering, feature extraction models, and LSTM (a recurrent neural network architecture).

1 Introduction

Sentiment Analysis is being used widely today. Major corporations use it to gauge public opinion about their products. This can be done by going through tweets about the products or by looking at the online reviews for the product on eCommerce websites. Online surveys about their products also help in gathering data. It is not only positive sentiment/reviews that help a brand, but the negative sentiment also helps the companies identify mistakes in marketing/quality of the product. Sentiment analysis can also be used by political parties to understand opinions about a party candidate, bills passed, etc.

Recognizing the semantic importance of words or phrases is a challenging task in itself, but in many cases, the overarching sentiment of a text is not the same as decontextualized snippets(Tony et al., 2004). There may be many positive words present in a negative review document, and conversely, a lot of negative words present in a positive review document; hence classifying an entire document as either positive or negative can be a daunting task.

In this paper, I will be performing sentiment analysis on the movie reviews data sourced from the Internet Movie Database(IMDB) archive of the rec.arts.movies.reviews newsgroup. Several models will be used for the task, and their performances will be compared. The SVM(Support Vector Machine) model, which obtained the highest accuracy in the Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002) paper, Adaboost Classifier, Gradient Boosting Classifier, and the LSTM(Long Short Term Memory), which are a type of recurrent neural network and have become immensely popular over the years with various applications in fields like speech recognition, grammar learning, text generation, etc.

The Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002) paper mentions that some feature engineering steps like stop word removal weren't performed before the sentiment analysis task. In addition to building the sentiment analysis models, these missing feature engineering steps would also be done on the movie review documents to see if they impact the models' performance. The unigram vectors to be fed into the sentiment analysis models were generated using a bag of words model. My SVM model(with bag of words feature extraction) performed almost as good as the SVM model in Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002), achieving a repeated average three-fold cross-validation accuracy of 82.257%.

2 Related Work

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002) is the most important paper to be discussed. The authors have tried solving the document classification problem considering the entire sentiment of the document. The training data used was the movie reviews dataset consisting of 2000 movie review documents(1000 positive and 1000 negative reviews). Three machine learning methods, namely Naïve Bayes, Maximum Entropy Classifier, and Support Vector Machines, were used for this task. They achieved the highest three-fold cross-validation accuracy on SVM with unigram features. Interestingly, these machine learning methods perform worse on sentiment classification than on topic-based categorization, hinting that sentiment classification is a more challenging task. The authors have mentioned in the paper that no stemming or stop word list was used for training the models. Feature engineering steps like lemmatization, stop word removal, stemming, etc., may further improve the performance of the models.

Another essential and closely related work is Wang, Xin, et al. (2015), where Twitter sentiment prediction was made using LSTM recurrent network. The LSTM model performed better than many feature engineering techniques on public noisy labeled data achieving results comparable to the then best data-driven technique. The model was able to distinguish special function words(negation and transition). It also augmented the dissimilarities between words having opposite sentiments. A dataset consisting of 1.6 million tweets(800000 positive tweets and 800000 negative tweets) was used for the model. 20000 negative and 20000 positive tweets were set aside as validation data for early stopping. The remaining 1.56 million tweets were used for training the model. For testing the model, 177 negative tweets and 182 positive tweets were used. The LSTM network with trainable lookup-table gave the highest accuracy achieving accuracies as high as 87.2%.

Agarwal, Apoorv, et al. (2011) performed sentiment analysis on Twitter data. Two sets of classifiers were built, a binary classifier for positive/negative tweet classification and a multiclass classifier(3 classes) for positive/neutral/negative tweet classification. Unigram, feature-based, and tree kernel models were built for these tasks. Out of these three models, it was observed that the tree kernel models not only outperformed the other two models but also performed almost as good as the best feature models. The dataset used consists of 11,875 manually annotated tweets, which were collected by archiving a real-time stream. The authors conclude that sentiment analysis on tweets is not so different from other forms/types of sentiment analysis.

3 Data

For the sentiment analysis task, I'll be using the same data as the one used in Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan(2002), which is the movie reviews dataset sourced from the Internet Movie Database(IMDB) archive of the rec.arts.movies.reviews newsgroup. The dataset consists of 2000 movie review documents(1000 positive and 1000 negative reviews). The average word count of the movie review documents is about 747 words. It was based on the star rating of the reviews that they were assigned to a positive, neutral or negative category(I have assigned 0 to negative review documents and 1 to positive review documents). Like the original work focus will be on distinguishing between the positive and the negative reviews. Out of the 2000 movie review documents, 1400 will be used for training the Machine Learning models, and the rest will be used for testing the performance of the models.

4 Methods

Supervised Machine Learning algorithms were used for the sentiment analysis task. I experimented with the SVM(support vector machine), Adaboost classifier, Gradient Boosting classifier, and LSTM models. Feature extraction for SVM, Adaboost Classifier, and Gradient Boosting Classifier was done using the bag of words model and the TF-IDF model, whereas GloVe(Global Vectors for word representation)100-dimensional pre-trained word vectors were used for LSTM.

4.1 Machine Learning Models

The SVM(Support Vector Machines) works by maximizing the margin, which is the distance between the hyperplane separating the classes(decision boundary) and the points closest to this hyperplane. The hyperplane, which maximizes the margin, is referred to as the optimal hyperplane.

Adaboost Classifiers use the concept of combining several poorly performing classifiers to form a strong classifier that gives good results. The wrongly classified observations/records in the dataset are assigned a high weight so that these records have a higher probability for classification in the next iteration. Each classifier also gets a weight in order of their classification accuracy(the most accurate classifier gets the highest weight). For classification, a majority vote across all the classifiers is taken to decide the observation class(positive or negative). The number of estimators used in the algorithm for this project was 1500.

Gradient Boosting Classifiers combine the Adaboosting method and weighted minimization. The objective of the algorithm is to minimize the

loss(the difference between the actual and predicted value). Unlike Adaboosting, where the weights are adjusted with new learners' addition, the gradient boosting algorithm retains the same weights(for previous learners) as new learners are introduced. The number of estimators used in the algorithm for this project was 1500.

Long Short-Term Memory(LSTM) is a type of recurrent neural network capable of learning order dependence in sequence prediction problems. LSTMs use mechanisms called gates to regulate information flow. The LSTM architecture consists of the forget gate, the input gate, the cell state, and the output gate. Gates help the LSTM model retain essential data in the sequence and discard the irrelevant information.

4.2 Feature Extraction Models

In the Bag words(BOW) model, vectors of fixed lengths are made by counting how many times each word occurs in the document. It's called a "bag of words" as the model is not concerned about the order and structure of the words but only about the words' presence in the document.

TF-IDF(Term Frequency – Inverse Document Frequency) works by finding the frequency of words in the document as well as finding the inverse document frequency(IDF), which measures the informativeness of a word. IDF is very low for commonly occurring words, and it's very low for rare words. The IDF multiplied with the term frequency gives us the TF-IDF score.

GloVe(Global Vectors for Word Representation) is an unsupervised learning algorithm the result of which is a vector representation of words. For this project, I used pre-trained word vectors of 100 dimensions for the LSTM model.

4.3 Tools Used

Python libraries like NumPy, pandas, scikit-learn, NLTK, re, Keras, string, and collections were used. The models were built using scikit-learn and Keras, whereas re, NLTK, string libraries were used for cleaning and preprocessing the data.

5 Evaluation

Since the dataset is balanced(1000 positive review documents and 1000 negative review documents) accuracy was used as the evaluation metric. The dataset was split into 1400-600(1400 for training and cross-validation, and 600 for testing). The average repeated(repeated seven times for consistent results) three-fold cross-validation accuracies for the models have been reported in figure 1 and figure 2. Unigram features were used for all the models since the highest accuracy model in Pang et al.,(2002) used a unigram features' model. The number of unigram features for all models(except LSTM) was fixed to 16165(same as Pang et al.,(2002)). The baseline model is the SVM model with no preprocessing and with a bag of words feature extraction, which had the highest accuracy in Pang et al.(2002).

6 Discussion and Conclusion

	Feature Extraction Model	SVM	Adaboost Classifier	Gradient Boosting Classifier
1.	BOW with no preprocessing	82.257	82.128	**82.428**
2.	BOW with preprocessing	**82.429**	81.564	81.957
3.	TF-IDF with no preprocessing	**84.143**	82.222	81.889
4.	TF-IDF with preprocessing(lemmatization included)	**83.893**	81.357	80.957

Figure 1: Repeated three-fold cross-validation accuracies(in %) of different Machine Learning Models for different feature extraction models

Feature Extraction Model	LSTM
GloVe Vectors	67.02

Figure 2 : Repeated three-fold cross-validation accuracy(in %) of LSTM Model for GloVe vectors

The repeated average three-fold cross-validation accuracies of the models have been shown in figures 1 and 2. The SVM model with BOW(Bag of Words) feature extraction and no

preprocessing achieved an accuracy of 82.257%, which is in line with the results obtained in Pang et al.,(2002)(SVM model with BOW model obtained an average three-fold cross-validation accuracy of 82.9%). In this category of models(BOW with no preprocessing), the Gradient Boosting classifier with 1500 estimators(other parameters set to default values) performed the best.

With preprocessing steps(stop words and punctuation removal) on the BOW feature extraction model, it was observed that the performance of the SVM model improved. In contrast, the accuracies of the other two models dipped.

Since only the BOW(Bag of words) model was used in Pang et al.,(2002), I wanted to check if other feature extraction models would be better suited for the sentiment analysis problem. The TF-IDF feature extraction model, paired with SVM, gave the highest accuracy among all models, 84.143%. The other two machine learning models also performed well in this setting.

Removing stopwords wouldn't impact the performance of a TF-IDF feature extraction model since it anyway gives a very low weightage to commonly occurring words. I observed that removing punctuation and stopwords gave the same results as the previous setting(TF-IDF with no preprocessing). Hence I also added lemmatization to the preprocessing and observed that the accuracies of all the models decreased, but the SVM model still gave good results(83.893%).

The LSTM model, paired with pretrained GloVe vectors, didn't perform that well, achieving a repeated average cross-validation accuracy of 67.02%. I suspected the model was overfitting, so I tried techniques such as early stopping and dropout addition to layers but observed that it didn't improve the model performance by much. It can be that the dataset is not huge enough to be trained on a deep learning model may be having more reviews in the dataset would help improve the LSTM results. Adding more layers to the LSTM could help as well.

Another interesting feature extraction model that can be tried as part of future work is the Word2Vec model. It can be obtained using two methods, Skip Gram and Common Bag of Words(CBOW). Tweaking the parameters of the machine learning models like Gradient Boosting classifiers could also give better results. Using a grid search, we can find the best combination of parameters like learning rate, the number of estimators, max-depth, etc., which can give us better results. Finally, we can also try BERT(Bidirectional Encoder Representations from Transformers) for sentiment classification, which applies bidirectional training to language modeling.

7 Acknowledgments

I wish to thank Prof Allen Riddell for his constant guidance, feedback, and support. This project was part of the Social Media Mining(ILS-Z639) course at Indiana University, Bloomington.

References

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques." *arXiv preprint cs/0205070* (2002).

Wang, Xin, et al. "Predicting polarities of tweets by composing word embeddings with long short-term memory." *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015.

Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." *Proceedings of the workshop on language in social media (LSM 2011)*. 2011.

Mullen, Tony, and Nigel Collier. "Sentiment analysis using support vector machines with diverse information sources." *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004.

https://www.brandwatch.com/blog/understanding-sentiment-analysis

https://en.wikipedia.org/wiki/Long_short-term_memory

https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

https://www.kaggle.com/rtatman/glove-global-vectors-for-word-representation

Franklin Thomas
frathom@iu.edu

https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

https://www.datacamp.com/community/tutorials/adaboost-classifier-python

https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/

https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

https://nlp.stanford.edu/projects/glove/

https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270