Term Project Report

Sentiment analysis, or opinion mining, is an important natural language processing task that attempts to extract the evident opinion, attitude, emotion of a given text which is useful in a wide range of applications beyond computational linguistics, including in business, communications, politics, and advertising (Liu, 2015). Sentiment analysis gained popularity in the early 2000s, as the Internet started to take the form we know it to be today, and today is one of the most active areas in NLP research (Zhang et al., 2015). Sentiment analysis can be performed on various levels – word/term level, sentence level, document level, and aspect level – and attempts to assign a polarity – positive, negative, or neutral – to the given input (Hussein, 2018). The primary challenge for sentiment analysis is extracting and combining useful features often from naturalistic contexts that may inform machine learning models text; common methods include negation, dependency parsing, pointwise mutual information (PMI), using sentiment and subjectivity lexicons, and more recently, deep learning methods and word embeddings have become the state-of-the-art (Zhang et al., 2017; Feldman, 2013). This term project explores the implementation of various feature/feature sets for a binary sentiment classification task on a movie reviews dataset and a restaurant reviews dataset.

Methods

Datasets

The datasets I used to train and evaluate my baseline and improved systems were a movie review dataset, consisting of 2000 movie reviews labeled as positive or negative (Pang et al., 2002), and a Yelp restaurant reviews dataset, consisting of 10391 restaurant reviews from the Champaign-Urbana region. The restaurant reviews dataset annotates reviews on a 5 star rating

system, and I collapsed all reviews with 3 stars and less as negative and all reviews with 4 and 5 stars as positive in order to implement a binary sentiment analyzer.

Data Preprocessing [Question 1]

I decided to keep my data preprocessing relatively simple, and only implemented sentiment-aware tokenization using Christopher Potts' HappyFunTokenizer, which also reduces all alphabetic characters to lowercase. This function also allows me to implement negation, which I used as one of my features. I chose not to remove any stop words or implementation lemmatization/stemming as I did not want to remove any potentially informative sentiment information.

Baseline System [Question 1]

For my baseline system, I used HappyFunTokenizer to label phrases that contain negation For example, the sentence "This is not a good movie" is annotated as "This is not_NEG a_NEG good_NEG movie_NEG". Negation is a popular feature in sentiment analysis that provides important information (Wiegand et al., 2010). A model that sees the raw sentence may label it as positive due to the word "good"; the "_NEG" tag inverts the polarity of that sentence. I then used these tokens to construct a TF-IDF matrix.

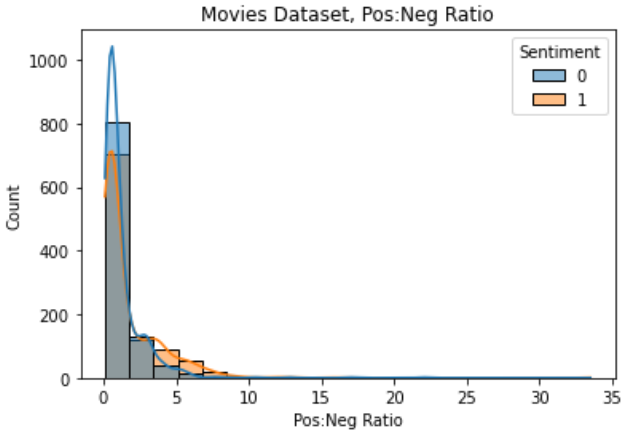
Improved Systems & Additional Features/Feature Sets [Question 3]

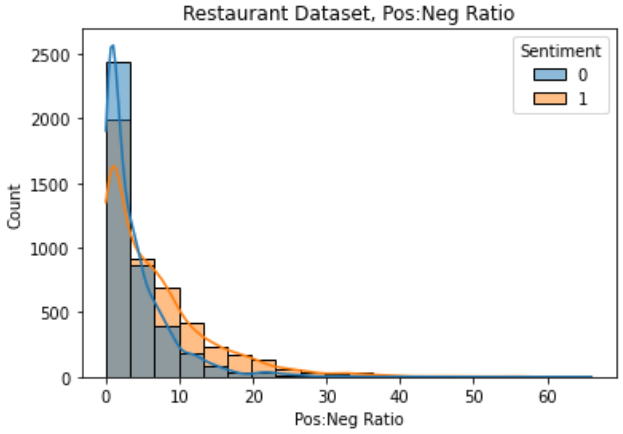
For my improved systems, I added 3 more features to my baseline system: the ratio of positive to negative words in a review, review length, and the number of noun phrases. In order to extract positive and negative words from a review, I employed 3 sentiment lexicons: the NRC Emotion Lexicon, NRC Emotion Affect Intensity Lexicon, and the Opinion Lexicon (Mohamed & Turney, 2013; Hu & Liu, 2004). After combining these 3 lexicons, I had a resulting sentiment lexicon dataset of 18,706 words with a labeled positive or negative polarity. To identify noun

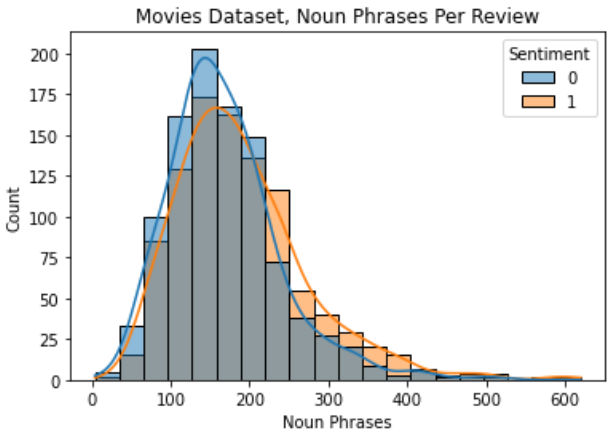
phrases, I employed SpaCy's dependency parser, which extracts noun phrases in a given sentence. I chose to use the raw values for these 3 values, rather than scaling them using sklearn's MinMaxScaler or StandardScaler as I found that using the raw values improved model performance. Histograms of these features for both datasets are shown in Figures 1-6.

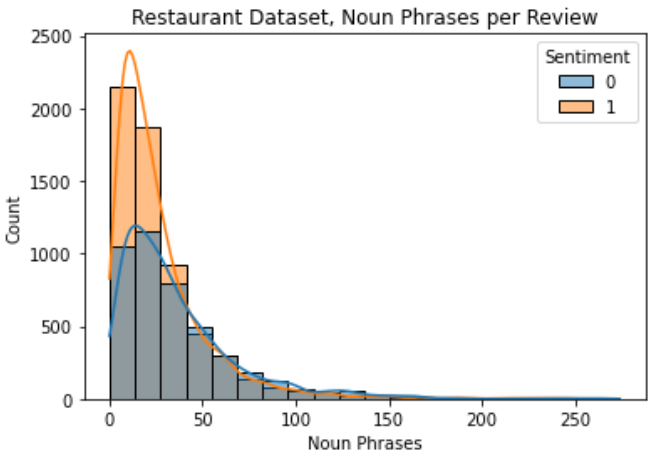
Models [Question 2]

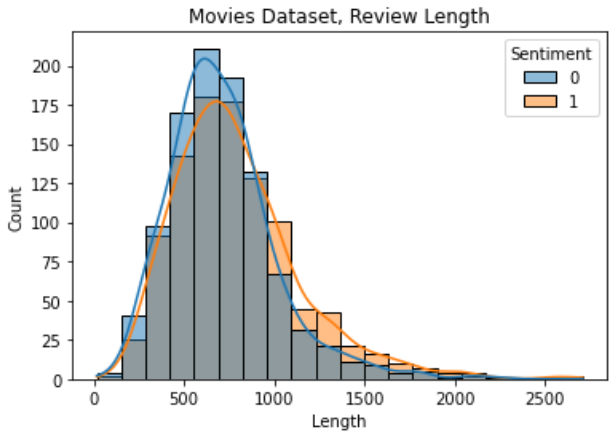
I used Naive Bayes, Decision Tree, Linear SVM, and Random Forest classifiers from sklearn to train my data on. I chose these models partially because these models were mentioned in sentiment analysis papers I reviewed (particularly Naive Bayes and SVM) and partially because, after exploring sklearn's available linear classifiers, I found their algorithms interesting.

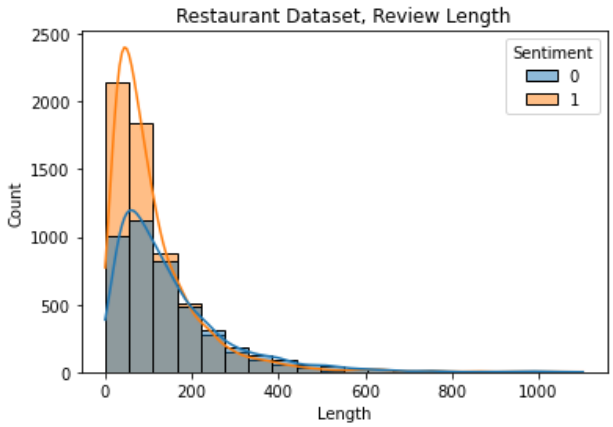












Figures 1-6. Histograms of feature distributions for both datasets. 0 refers to negative sentiment and 1 refers to positive sentiment.

Feature Engineering & Model Evaluation [Question 4]

I obtained F1, precision, recall and accuracy metrics for my baseline and improved systems. For feature engineering, I decided to implement the incremental approach, as suggested in the assignment description.

		Baseline (Negation)	Negation + Pos:Neg Ratio	Negation + Pos:Neg Ratio + Noun Phrases	All features
Naive Bayes	F1 Precision Recall Accuracy	0.686 0.687 0.684 0.687	0.686 0.687 0.684 0.687	**0.750** 0.750 0.750 0.750	**0.752** 0.757 0.755 0.753
Decision Tree	F1 Precision Recall Accuracy	0.638 0.638 0.637 0.638	0.635 0.637 0.635 0.636	0.692 0.692 0.693 0.692	0.692 0.696 0.695 0.692
Linear SVM	F1 Precision Recall Accuracy	0.683 0.685 0.684 0.684	0.669 0.674 0.671 0.669	0.605 0.695 0.646 0.639	0.563 0.543 0.630 0.62
Random Forest	F1 Precision Recall Accuracy	**0.695** 0.687 0.696 0.695	**0.689** 0.695 0.691 0.69	0.643 0.713 0.675 0.668	0.618 0.605 0.669 0.661

Table 1. Performance metrics for Movie Reviews Dataset. All features include negation (TF-IDF), Pos:neg ratio, Noun phrases, and review length. The highest F1 measure for each feature system/combination is bolded.

		Baseline (Negation)	Negation + Pos:Neg Ratio	Negation + Pos:Neg Ratio + Noun Phrases	All features
Naive Bayes	F1	0.730	0.747	0.758	0.794
	Precision	0.732	0.747	0.756	0.793
	Recall	0.727	0.745	0.756	0.792
	Accuracy	0.732	0.748	0.758	0.794
Decision Tree	F1	0.755	0.747	0.756	0.776
	Precision	0.756	0.746	0.755	0.775
	Recall	0.754	0.746	0.755	0.775
	Accuracy	0.756	0.747	0.756	0.776
Linear SVM	F1	0.784	0.770	0.776	0.789
	Precision	0.785	0.769	0.774	0.790
	Recall	0.782	0.769	0.775	0.788
	Accuracy	0.784	0.770	0.776	0.790
Random Forest	F1	**0.800**	**0.792**	**0.794**	**0.804**
	Precision	0.801	0.791	0.793	0.805
	Recall	0.798	0.790	0.793	0.802
	Accuracy	0.800	0.792	0.794	0.804

Table 2. Performance metrics for Restaurant Reviews Dataset. All features include negation (TF-IDF), Pos:neg ratio, Noun phrases, and review length. The highest F1 measure for each feature system/combination is bolded.

Results

Performance metrics for baseline and improved systems for both datasets are presented in Table 1 and Table 2. For the movie reviews dataset, the Naive Bayes classifier performed the best under the improved system with 3 features (F1 = 0.750) and under the improved system with all features included (F1 = 0.752). Additionally, when features were added incrementally, the Naive Bayes and Decision Tree classifiers increased performance while the Linear SVM and Random Forest classifiers' performance actually decreased.

For the restaurant reviews dataset, the Random Forest classifier performed the best under the baseline and all improved systems, with the highest performance under all 4 features (F1 = 0.804). Additionally, the other models did not seem to have reduced performance from baseline to improved systems, unlike with the movie reviews dataset. *[Question 2, 4]*

Discussion & Conclusions

Overall, the models seemed to perform better with the restaurant reviews dataset. This may potentially be due to the fact that for my chosen additional features, the positive and negative reviews presented much more distinct distributions. Additionally, I had more samples for the restaurant reviews dataset; the restaurant reviews dataset contained 8781 samples, and the movie reviews dataset contained 2000 samples. Interestingly, the movie reviews dataset has a larger total number of feature columns (14559 total features) compared to the restaurant dataset (28004 total features) due to a larger TF-IDF matrix. I am not entirely satisfied with my models' performance; perhaps the restaurant dataset would have benefitted from a smaller TF-IDF matrix, and in a future iteration of this project I could consider removing stop words or lemmatization/stemming.

In terms of feature combinations, both datasets had the highest performance under the improved system that contained all features, albeit for different classifiers. This suggests that my chosen features indeed do provide the models important information about a given reviews' sentiment. In future iterations, I would like to explore other features, for example using an LDA model to obtain topic words that I can perhaps construct word embeddings from.

This project allowed me to explore just how text classification models are implemented, which is something I had not done before. I was able to explore different feature extraction methods and investigate different data exploration/science techniques as well.

References

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, *56*(4), 82-89.

Hussein, D. M. E. D. M. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*, *30*(4), 330-338.

Liu, B. (2015). Sentiment analysis: Mining opinions, sentiments, and emotions. The Cambridge University Press.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

Wiegand, M., Balahur, A., Roth, B., Klakow, D., & Montoyo, A. (2010, July). A survey on the role of negation in sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing* (pp. 60-68).

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1253.