Jeffrey Yan
LING 406
Term Project, EC 1 and 2

Sentiment Classifier for Review Data

Introduction

Sentiment analysis has become popular and very widely used, from businesses using it to check their brand/product's perception to researchers attempting to track changes in sentiment over time. For example, a business can use sentiment analysis to see what customers like about their product, which can influence future development of its products. A politician could try to gauge their campaign's success and find which aspects of their platform that people react positively or negatively to.

Problem definition

Sentiment analysis is the analysis of data for the purpose of identifying an opinion, or sentiment, that a person has towards something. In Computational Linguistics, given an input in text form, a program would classify the sentiment of that input, such as by putting it into a "positive", "negative", or "neutral" category. In this project, there are multiple movie reviews from a Movie Reviews dataset, and the program classifies each review as "positive" or "negative". The same program is then applied on a Yelp Restaurant Reviews dataset, with the same classification of "positive" or "negative". This task is made easier due to each review being labeled with the actual sentiment, making it easier to train a model. In the case of the Yelp dataset, the reviews are not divided into "positive" and "negative", but are between 1 and 5 stars, requiring a decision to be made on which star range can be matched to each sentiment.

Previous work

Yessenov and Misailovic (2009) was very comprehensible, classifying movie review comments by subjectivity/objectivity and a negative/positive/neutral attitude. In contrast to my project, the authors had to manually classify each comment. Also, since the data was from a discussion forum, the comments were not as long or movie-focused as the movie review dataset I used. However, the process was helpful, as the authors also mentioned marking words influenced by negation. The authors used the Naive Bayes, Maximum Entropy, Decision Trees, and K-Means clustering algorithms and found that Naive Bayes with bag-of-words and negated words returned the best results.

Angiani et al. (2016) was also very helpful in studying preprocessing techniques. One interesting operation was removing vowels that repeated more than 2 times, since people may add extra vowels for emphasis when typing online. There was also the technique of replacing all negative constructs with "not", to increase the frequency of

negation bigrams that would be inputted in a classifier model. An important technique was removing stopwords, or very commonly occurring words that might reduce the accuracy of classification.

Fang and Zhan (2015) use Amazon product reviews to categorize sentiments both at the sentence-level and the review-level. The authors removed all sentences that did not contain a positive or negative word. They used a POS tagger to help with this, since nouns and pronouns could be considered neutral, and some words convey a different sentiment depending on its part of speech. The authors then propose their own algorithm for identifying negation phrases such as “not worth”. For marking sentences with “positive” or “negative”, the authors used a bag-of-words model that counted the number of positive versus negative tokens in a sentence, and marked the sentence with whichever type occurred more. The classifiers used were Naive Bayes, Random Forest, and Support Vector Machine, with 10-fold cross-validation. The sentence-level classification had F1 scores above 0.80.

Zahoor et al. (2020) collect Facebook reviews of Pakistani restaurants to classify comments into positive and negative categories. For preprocessing, the authors removed stopwords and lemmatized words. They used bag-of-words. The classifiers used were Naive Bayes, Logistic Regression, Support Vector Machine, and Random Forest. Random Forest uses several decision trees to reduce overfitting. The authors managed an F1 score of 0.95 with this classifier.

Approach

I used a bag-of-words approach for the base. When preprocessing the data, I first ensured that the reviews were all in lowercase, then I removed all stop words from nltk.corpus minus negation. I realized that negation could have an impact on the results, so I decided to keep those words though they would normally be considered unimportant. This meant that lemmatization would be difficult, especially with negation markings such as “_NEG” that I would add in my improved model, so I decided against using it. I also removed numbers and punctuation except for underscore to clean up the data. Then I labeled each review as either “Positive” or “Negative” and saved all of it as a pandas dataframe. The result was organized into negative reviews first, then positive reviews, so I needed to randomize the order. I found that I could do this with a parameter in the sklearn KFold function, which shuffles the data before splitting it. Due to this random shuffling, I needed to also add a parameter “random_state=0” in order to get the same performance result. I used CountVectorizer() to construct the bag-of-words with word frequency.

Referencing Yessenov and Misailovic (2009), I chose Naive Bayes, Decision Trees, and Maximum Entropy. I used the Multinomial Naive Bayes classifier because the documentation stated that it is useful for discrete features such as word counts. The major advantage was the speed of this algorithm, for a similar F1 score to Maximum

Entropy. The model assumes that features are independent, which simplifies the calculation. However, Maximum Entropy was still more effective, being a more generalized version of Naive Bayes (Yessenov and Misailovic, 2009). The Decision Tree classifier performed significantly worse. I also tried using a Support Vector Machine classifier, but it took up to an hour to run each test, and performed similarly to Naive Bayes.

To try to improve the model, I first tried using tf-idf instead of simple word frequency, which I believed would help by weighting words. Inspired by Angiani et al. (2016), I also tried replacing negations with “not”. Then, after each negation, I marked all tokens up to the end of a clause with “_NEG” which I thought could help because negation can change the sentiment of the rest of the clause, and this impact would otherwise be lost with a bag-of-words approach. My final change used the provided EmoLex data to first get all words associated with an emotion. Then, I removed all sentences that did not contain one of these “emotion” words. I was inspired by Fang and Zhan (2015) for this modification.

I then tested my baseline and improved systems on a different dataset. The Champaign-Urbana Yelp restaurant reviews data was displayed differently, so I needed to change the format to be similar to my previous Movie Reviews dataset. I marked all restaurant reviews with a rating from 1 to 3 as “negative” and all reviews with a rating from 4 to 5 as “positive”. I also needed to add spaces between words and punctuation to resemble the Movie Reviews data format. Then, I was able to leave the rest of the code the same as the code for the Movie Reviews data.

Results for Movie Reviews

I used the leave-one-out-approach to get the performance for the four features I changed. All results, including precision, recall, and accuracy, are found in the Appendix. The average F1 score for each model is shown in the table:

Feature 1: All negations replaced by “not”
Feature 2: All tokens affected by negation up to end of clause marked with “_NEG”
Feature 3: All sentences without an “emotion” word are removed
Feature 4: tfidfVectorizer() instead of CountVectorizer()

	Average F1 Score for Model		
	Multinomial Naive Bayes	Decision Tree	Maximum Entropy
Baseline	81.36%	62.13%	**84.51%**

Features 1,2,3	81.46%	63.17%	83.31%
Features 1,2,4	**81.50%**	**63.40%**	83.12%
Features 1,3,4	80.85%	63.08%	83.08%
Features 2,3,4	80.95%	62.89%	82.91%
Features 1,2,3,4	80.79%	61.92%	82.47%

It appears that for Naive Bayes, it was better to either keep sentences without emotion words, or to not use tf-idf. Interestingly, while both of those combinations of features performed better than the base with two out of three classifiers, the base had a higher average F1 score than any attempted improvements for the Maximum Entropy classifier, and the highest average F1 score of any result. If I had only run the Naive Bayes classifier on the baseline system, I would not have discovered this.

Results for Yelp Reviews

I used the same leave-one-out approach for this new data:

Feature 1: All negations replaced by “not”
Feature 2: All tokens affected by negation up to end of clause marked with “_NEG”
Feature 3: All sentences without an “emotion” word are removed
Feature 4: tfidfVectorizer() instead of CountVectorizer()

	Average F1 Score for Model		
	Multinomial Naive Bayes	Decision Tree	Maximum Entropy
Baseline	**76.93%**	**72.45%**	81.15%
Features 1,2,3	75.85%	70.45%	78.16%
Features 1,2,4	67.48%	71.89%	**82.25%**
Features 1,3,4	65.61%	69.72%	79.61%
Features 2,3,4	65.41%	69.51%	80.13%
Features 1,2,3,4	65.18%	70.22%	80.09%

The Naive Bayes classifier performed significantly worse with this dataset, although Maximum Entropy performed almost as well. It appears that the same mix of features: 1,2,3 and 1,2,4 were effective in both datasets, with 1,2,4 (keeping all sentences, even ones without emotion words) having the highest overall F1 score in this case, using the Maximum Entropy classifier. The Movie Review dataset classified 3.5 stars and above as positive, and 2 stars and below as negative, leaving out the reviews for ratings in between. This might explain some of the difference in performance; since I counted 3-star reviews as negative, the language in those reviews could have affected the accuracy, since they might be more neutral. I might have gotten better performance by leaving out any 3-star reviews. The amount of data might have also affected the performance; while the Movie Reviews dataset has very long reviews, Yelp reviews are much shorter on average.

Discussion and Conclusions

I learned a lot from this project about processing data and determining which models to use. With a more powerful computer and more time, I would see how much a Support Vector Machine classifier would improve the accuracy. I was surprised by how changing the features only led to small percent improvements, but reading other papers shows that around an 80% F1 score is not too far off the mark.

There are many possibilities for improvement. One possibility is using a lemmatizer, although that was very difficult for me to execute due to one of my features being a "_NEG" marking. I could also use 10-fold rather than 5-fold cross-validation, which seems to be more common from the papers I have read. The main issue would be runtime. I could also try part-of-speech tagging and have each word with its associated part-of-speech. This could distinguish some words that may index different sentiments depending on the part-of-speech, although there would need to be a sufficient amount of data. This could help with the Yelp restaurant dataset, where there are more food-related nouns, and there may be enough information to improve the model. I would also consider using Random Forest instead of Decision Trees, since Random Forest uses averages of many decision trees, which is useful because decision trees can end up overfitting.

Works Cited

Yessenov, Kuat, and Sasa Misailovic. “Sentiment Analysis of Movie Review Comments” Massachusetts Institute of Technology, Spring 2009. http://people.csail.mit.edu/kuat/courses/6.863/report.pdf

Angiani, G., Ferrari, L., Fontanini,T., Fornacciari, P., Iotti, E., Magliani, F., &Manicardi, S. (2016). A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter. In KDWeb.

Fang, X., Zhan, J. Sentiment analysis using product review data. *Journal of Big Data* 2, 5 (2015). https://doi.org/10.1186/s40537-015-0015-2

K. Zahoor, N. Z. Bawany and S. Hamid, "Sentiment Analysis and Classification of Restaurant Reviews using Machine Learning," *2020 21st International Arab Conference on Information Technology (ACIT)*, Giza, Egypt, 2020, pp. 1-6, doi: 10.1109/ACIT50332.2020.9300098.

Appendix

Feature 1: All negations replaced by “not”
Feature 2: All tokens affected by negation up to end of clause marked with “_NEG”
Feature 3: All sentences without an “emotion” word are removed
Feature 4: tfidfVectorizer() instead of CountVectorizer()

F1, Precision, Recall, and Accuracy for Movie Review Models

Baseline	Average F1 for Naive_Bayes: 0.813587081877839 Average Precision for Naive_Bayes: 0.8135985271703335 Average Recall for Naive_Bayes: 0.813933679628291 Average Accuracy for Naive_Bayes: 0.8140000000000001 Average F1 for Decision_Tree: 0.6212577546193891 Average Precision for Decision_Tree: 0.6214052795812168 Average Recall for Decision_Tree: 0.6215279851946158 Average Accuracy for Decision_Tree: 0.6220000000000001 Average F1 for Max_Entropy: 0.8450694935554829 Average Precision for Max_Entropy: 0.8457648381576334 Average Recall for Max_Entropy: 0.8453949096594234 Average Accuracy for Max_Entropy: 0.8454999999999998
Features 1,2,3	Average F1 for Naive_Bayes: 0.8145741080277988 Average Precision for Naive_Bayes: 0.8145636769045824 Average Recall for Naive_Bayes: 0.8148905747232066 Average Accuracy for Naive_Bayes: 0.8149999999999998 Average F1 for Decision_Tree:

	0.6317246397662764 Average Precision for Decision_Tree: 0.6328556193136672 Average Recall for Decision_Tree: 0.6329320153800992 Average Accuracy for Decision_Tree: 0.632 Average F1 for Max_Entropy: 0.8330611345289345 Average Precision for Max_Entropy: 0.8331111431430152 Average Recall for Max_Entropy: 0.833206726460704 Average Accuracy for Max_Entropy: 0.8335000000000001
Features 1,2,4	Average F1 for Naive_Bayes: 0.8149593136360368 Average Precision for Naive_Bayes: 0.8184967208054917 Average Recall for Naive_Bayes: 0.817774050767123 Average Accuracy for Naive_Bayes: 0.8150000000000001 Average F1 for Decision_Tree: 0.6339963474098111 Average Precision for Decision_Tree: 0.6345692009844915 Average Recall for Decision_Tree: 0.6344426606591851 Average Accuracy for Decision_Tree: 0.635 Average F1 for Max_Entropy: 0.8311850527103697 Average Precision for Max_Entropy: 0.8314014042327186 Average Recall for Max_Entropy: 0.831787557467656 Average Accuracy for Max_Entropy: 0.8314999999999999
Features 1,3,4	Average F1 for Naive_Bayes: 0.8084699927004072 Average Precision for Naive_Bayes: 0.8122444477800649

	Average Recall for Naive_Bayes: 0.8114102777443082 Average Accuracy for Naive_Bayes: 0.8084999999999999 Average F1 for Decision_Tree: 0.6307614623286842 Average Precision for Decision_Tree: 0.6315276814359939 Average Recall for Decision_Tree: 0.6314582510294147 Average Accuracy for Decision_Tree: 0.6315 Average F1 for Max_Entropy: 0.8307746114697139 Average Precision for Max_Entropy: 0.8312807052948982 Average Recall for Max_Entropy: 0.8317346170694799 Average Accuracy for Max_Entropy: 0.8310000000000001
Features 2,3,4	Average F1 for Naive_Bayes: 0.809468316163823 Average Precision for Naive_Bayes: 0.812990770155072 Average Recall for Naive_Bayes: 0.8122925726927239 Average Accuracy for Naive_Bayes: 0.8094999999999999 Average F1 for Decision_Tree: 0.6289104664990448 Average Precision for Decision_Tree: 0.6297543971193763 Average Recall for Decision_Tree: 0.6296682135222522 Average Accuracy for Decision_Tree: 0.63 Average F1 for Max_Entropy: 0.8291474844842842 Average Precision for Max_Entropy: 0.829674437203848 Average Recall for Max_Entropy: 0.82979839891865 Average Accuracy for Max_Entropy: 0.8295

Features 1,2,3,4	Average F1 for Naive_Bayes: 0.8079380883785785 Average Precision for Naive_Bayes: 0.8119796400847742 Average Recall for Naive_Bayes: 0.810929550344745 Average Accuracy for Naive_Bayes: 0.808 Average F1 for Decision_Tree: 0.6191526614885313 Average Precision for Decision_Tree: 0.6203174239600939 Average Recall for Decision_Tree: 0.6201262244789335 Average Accuracy for Decision_Tree: 0.62 Average F1 for Max_Entropy: 0.8247291223489034 Average Precision for Max_Entropy: 0.825198737913604 Average Recall for Max_Entropy: 0.8256080354798001 Average Accuracy for Max_Entropy: 0.825

F1, Precision, Recall, and Accuracy for Yelp Review Models

Baseline	Average F1 for Naive_Bayes: 0.7692706881026681 Average Precision for Naive_Bayes: 0.8027028271589446 Average Recall for Naive_Bayes: 0.7619383632441006 Average Accuracy for Naive_Bayes: 0.7877950410192952 Average F1 for Decision_Tree: 0.7244753466219658 Average Precision for Decision_Tree: 0.72359220613212 Average Recall for Decision_Tree: 0.728510063287598 Average Accuracy for Decision_Tree: 0.728804938333331

	Average F1 for Max_Entropy: 0.8114827694225596 Average Precision for Max_Entropy: 0.8097807062380287 Average Recall for Max_Entropy: 0.816564657087943 Average Accuracy for Max_Entropy: 0.8144547820197483
Features 1,2,3	Average F1 for Naive_Bayes: 0.7584718020007071 Average Precision for Naive_Bayes: 0.7920262145674155 Average Recall for Naive_Bayes: 0.7516127360751985 Average Accuracy for Naive_Bayes: 0.7781713741290257 Average F1 for Decision_Tree: 0.7044502706174083 Average Precision for Decision_Tree: 0.7039491563811546 Average Recall for Decision_Tree: 0.7088475800899572 Average Accuracy for Decision_Tree: 0.7085941221648633 Average F1 for Max_Entropy: 0.7815582254630229 Average Precision for Max_Entropy: 0.7845094965499742 Average Recall for Max_Entropy: 0.7850035557347632 Average Accuracy for Max_Entropy: 0.7873148275458188
Features 1,2,4	Average F1 for Naive_Bayes: 0.6748063625244479 Average Precision for Naive_Bayes: 0.8224607136285511 Average Recall for Naive_Bayes: 0.6812179906757473 Average Accuracy for Naive_Bayes: 0.7323623512266438 Average F1 for Decision_Tree: 0.718858070191166 Average Precision for Decision_Tree: 0.7180758662104312

	Average Recall for Decision_Tree: 0.7233627758116061 Average Accuracy for Decision_Tree: 0.7228356714400987 Average F1 for Max_Entropy: 0.822543516304471 Average Precision for Max_Entropy: 0.8208692991369885 Average Recall for Max_Entropy: 0.8252525471104615 Average Accuracy for Max_Entropy: 0.8262909585334993
Features 1,3,4	Average F1 for Naive_Bayes: 0.6561152411720775 Average Precision for Naive_Bayes: 0.8088441680931904 Average Recall for Naive_Bayes: 0.6664051302937948 Average Accuracy for Naive_Bayes: 0.719371125434648 Average F1 for Decision_Tree: 0.6971854316463884 Average Precision for Decision_Tree: 0.6964935900311036 Average Recall for Decision_Tree: 0.7011011422732017 Average Accuracy for Decision_Tree: 0.7016640579681966 Average F1 for Max_Entropy: 0.7961181438574745 Average Precision for Max_Entropy: 0.7946521499849565 Average Recall for Max_Entropy: 0.7984965256870391 Average Accuracy for Max_Entropy: 0.8004999349561428
Features 2,3,4	Average F1 for Naive_Bayes: 0.6541499983468677 Average Precision for Naive_Bayes: 0.811056014833728 Average Recall for Naive_Bayes: 0.6650772971846436

	Average Accuracy for Naive_Bayes: 0.7186008765412037 Average F1 for Decision_Tree: 0.6951045712566917 Average Precision for Decision_Tree: 0.6945979579892805 Average Recall for Decision_Tree: 0.699141772153801 Average Accuracy for Decision_Tree: 0.699547748440915 Average F1 for Max_Entropy: 0.8013077837487581 Average Precision for Max_Entropy: 0.7997831834839935 Average Recall for Max_Entropy: 0.8040987207533308 Average Accuracy for Max_Entropy: 0.8054080379393179
Features 1,2,3,4	Average F1 for Naive_Bayes: 0.6517759398183108 Average Precision for Naive_Bayes: 0.8104557260714695 Average Recall for Naive_Bayes: 0.6633313515079564 Average Accuracy for Naive_Bayes: 0.7171571343852383 Average F1 for Decision_Tree: 0.7021578127740109 Average Precision for Decision_Tree: 0.7016248286030222 Average Recall for Decision_Tree: 0.7064874184440393 Average Accuracy for Decision_Tree: 0.7063801311154535 Average F1 for Max_Entropy: 0.8008659114297764 Average Precision for Max_Entropy: 0.7993236096223834 Average Recall for Max_Entropy: 0.8037433728391212 Average Accuracy for Max_Entropy: 0.8049267133963959