Helena Chi, helenac2
LING 406
Due Wednesday, May 8, 2019

Term Project Spring 2019

Introduction

Natural Language Processing (NLP) is one of the biggest topics in the computational linguistics field today. A popular challenge in NLP is sentiment analysis, also known as opinion mining, which creates systems to attempt to identify and analyze opinions people have about a certain topic, in addition to lower-level details of those opinions. This is done by determining the subjectivity or a piece of text first, then determining polarity, as objective texts are neutral and do not have polarity.

Sentiment analysis is becoming increasingly important in today's world of data analytics, as there are so many applications that utilize its power. Especially since a vast majority of stored textual data is not properly formatted, it is crucial, more than ever before, to extract the general sentiment, polarity, and opinion holder of an opinion. Sentiment analysis can be used to mine for reviews of almost anything, predict the outcomes of social events, improve text-to-speech synthesis, help with ad placements online, and summarization, amongst other such business and social problems.

Previous Work

Seeing that sentiment analysis is on the rise, I would like to discuss some work that people have published regarding sentiment analysis before we discuss my experience tackling this task.

The first paper I read was on the topic of "Sentiment Analysis of Short Informal Texts, not too dissimilar from movie reviews. In their article, Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad describe their sentiment analysis system that detects the sentiment on the message-level and term-level. Their approach was very similar to the approach I used, explained in the next section, as seen by their use of general-purpose sentiment lexicons and polarity categorization. They also consider negation's role in determining sentiment for their dataset. After testing a number of different kernel types for SVM models, they eventually determined that a linear-kernel performed better than the other types, similar to what I've discovered in the Results section of this paper. Their results showed that using tweet-specific lexicons resulted in better performance in comparison to general-purpose lexicons.

The next paper I would like to discuss called "Detecting Sadness in 140 Characters: Sentiment Analysis and Mourning Michael Jackson on Twitter", written by Elsa Kim and Sam Gilbert in 2009. What I find interesting about their findings is that not only did they find commonalities among the tweets in response to Michael Jackson's passing, they were also able to discover something about impact of this incident on society through the peak frequency of tweets per second that related to it and the characters of these tweets compared to "everyday" tweets. This exemplifies the applications of sentiment analysis even outside just focusing on the meaning of texts themselves and the ability to analyze all of these tweets as a whole. Regarding the dataset, they looked out for data that contained words in their list of search terms, which is a

very effective way to retrieve relevant data and decrease noise. Unfortunately, they did not account for tweets in languages other than English, which may have a significant impact on the general analysis of the all the tweets in regard to the worldwide effect Michael Jackson's death may have had. In addition, they had four rating categories – expressing sadness, not expressing sadness, mixed emotions, and unclear sentiment. Such categories are not that common in the world of NLP but would be applicable to the task of determining movie review polarity, too. For their task, they used the Affective Norms for English Words (ANEW) dataset, allowing them to specifically analyze the emotional ratings of the tweets. Their results showed that similar to the paper discussed above, using ANEW, a tweet-specific lexicon on emotions, had better performance than using general purpose lexicons. However, they also discovered that hand-coding the emotion in tweets allows for more nuanced data that ANEW still can't reach, implying that human coding techniques are helpful techniques to sentiment analysis.

The last paper is most relevant to this project's task, as it covers Kuat Yessenov and Sasa Misailovic's experience with "Sentiment Analysis of Movie Review Comments". For their task, they classified the comments in the same way as done in the first paper, indicating that categorizing texts as subjective or objective first then to positive or negative attitudes is a common standard for many sentiment analysis tasks. They used Naïve Bayes, Decision Trees, and Maximum Entropy, some very popular models in machine learning, to classify their data. Similar to the previous two articles, negation was also part of their data pre-processing. Even though they only ran their analysis on reviews of the movies *Quantum of Solace* and *Startrek*, their results yielded some very decently high accuracies for classification by subjectivity. Their findings also concluded that, at least for their dataset, a simple bag-of-words model preforms relatively well and can be further refined through feature choices.

Problem Definition

For this project, the task is to determine the polarity (positive or negative) of a given piece of textual data and eventually the accuracy of models used to evaluate data -- specifically, movie reviews collected by Bo Pang and Lillian Lee in 2004 for their paper called "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". This was done by testing a few different machine learning algorithms on the movie reviews, all of which were also used in at least one of the papers discussed above. In the rest of this report, I will be guiding you through my process of implementing a sentiment classifier for this task.

Approach

This task required multiple steps to take before arriving at the polarity-determining sentiment classifier. The first step was to create a baseline model for future comparisons with other classifiers. My baseline model for these movie reviews was the Bag-of-Words model, which takes each individual word in a piece of text as a key in a dictionary and assigns the number of occurrences in that piece of text as the key's value. This dictionary in this case would be the feature representation for this model. I did not have a fixed feature set size because of this, however, I think that the smaller a feature set, the less accurate the results, as words do not fall into a finite number of categories and would definitely benefit more from having more categories that happen to be more descriptive and helpful to the task at hand.

The next step to this task was to determine how to pre-process the given data to ensure that the models will be working with a quality feature set, free of noise and full of meaningful components. However, it is quite difficult to make any set of data noise-free, as the importance of kinds of words varies from application to application. Because of this, I first experimented with a couple feature selection methods to better prepare training data for our classifiers. The first strategy I implemented in the pre-processing portion of the task was the removal of stop words. Stop words can vary from task to task, but the most popular "definition" of it is a list of words that do not add much meaning and can be removed without altering the meaning of a piece of text. Examples of this would include articles such as "a" and "those", prepositions like "of" or "to", and pronouns and possessives, and more categories and words. A stop-word list can be created manually or found online, but the lists I used are the stop-word lists offered by the NLTK library. The other strategy I implemented was lemmatization. Lemmatization is a popular technique used in many NLP tasks to extract the base form of words in order to prevent different inflections of the same word as two different and unrelated words. For this project, I used WordNetLemmatizer, which is also provided in the NLTK library. I initially thought that lemmatizing would be the most crucial feature, as it groups all the inflections of one word together, however, as I will discuss later in this paper, the Bag-of-Words itself seems to be very helpful already, as it seems like the movie reviews provided to not repeat words as often as articles or research paper do, making it enough to make each classifier pretty accurate.

For creating the feature sets, I compiled two lists of features: positive words and negative words. I did this because this follows a similar structure to how the movie reviews were stored and also allowed for a more proportionate distribution of positive and negative entries for both the training and testing sets. To do this, I extracted all the words from the pre-processed of the data from the positive and negative reviews into respective lists and divided both of those lists in the cross validation portion. I chose to implement cross validation, as it significantly reduces bias and varies the data used for training.

Lastly, I chose to compare the baseline model with three different machine learning algorithms: Naïve Bayes, Decision Tree, and Support Vector Machine (SVM). The Naïve Bayes classifier assumes that each word in a feature set is independent and calculates the probability of the word prior to the current one. In sentiment analysis, it calculates the probability of the given piece of text, then assigns it to the most likely polarity. I chose to use the Naïve Bayes model because it is a very popular and straightforward method in machine learning. The second model used is the Decision Tree. This determines the probability of the polarity of a given piece of text through many input variables, which in this case are the polarities of each word in the text. I used this model because it is generally very helpful for visual analysis when graphed and is not too complex upon first glance at such a graph. The last model is the SVM, which works by dividing its training data with an optimal hyperplane that categorizes the polarities of the words in the training set. I chose to implement this classifier because depending on how one treats the data, the way in which the SVM divides data into categories can be very flexible and the graphical representations can vary but still be very intuitive to analyze.

Results

Through this project, I have learned that Decision Tree Models are relatively variable in terms of accuracy and performance in comparison to the other two models I experimented with. As shown in the Table 1 and Table 2, the SVM model (values marked in red) yields the highest

values for performance, just like it did for just the bag of words. This makes sense because, as described previously, the SVM approach allows for a very inclusive method of dividing data to their respective polarity categories, while both Naïve Bayes and Decision Tree models are more rigid, much more black-and-white, when determining polarity.

Table 1. Accuracy Metrics on Bag of Words Baseline Model

Accuracy	Naive Bayes	Decision Tree	SVM
Accuracy	0.89915	0.87887	0.95331
Positive precision	0.83754	0.87750	0.95277
Negative precision	0.99010	0.88048	0.95397
Positive recall	0.99198	0.88157	0.95405
Negative recall	0.80632	0.87617	0.96257

Table 2. Accuracy Metrics on Fully Pre-processed Data

Accuracies	Naïve Bayes	Decision Tree	SVM
Accuracy	0.89205	0.88192	0.95236
Positive precision	0.82755	0.88714	0.95162
Negative precision	0.98988	0.87718	0.95323
Positive recall	0.99195	0.87577	0.95328
Negative recall	0.79215	0.88807	0.95143

The most important thing that stood out to me through gathering data for the accuracy of different levels of pre-processed data was seeing which feature strategy was the most impactful in generating a model with the highest accuracy.

The bolded values in Table 3 signify the level of pre-processing for which each model does the worst in. Because these are comparisons of the accuracy, the data suggests that the lower the accuracy value, the more crucial that feature is for the corresponding model. Here we see that the lowest accuracy value for the Naïve Bayes classifier occurs when we do not filter out the stop words. …. Doing no pre-processing steps for data, aka just testing on the Bag of Words, yields the highest accuracy values for all three of the models. However, this does not entirely suggest that using the Bag of Words as a feature is the least useful, as that was the only form of "pre-processing" and still yields very similar values to the other ones. This makes sense because the Bag of Words is the primary source of features in the feature sets for these models.

Table 3. Accuracy on for Different Levels of Pre-processed Data

Accuracy	Naïve Bayes	Decision Tree	SVM
Just Bag of Words	0.89915	0.87887	0.95331
Stop-words not filtered	**0.89682**	0.87758	0.95198
Unlemmatized	0.89799	**0.88429**	**0.95332**

To gather the data shown in the next three tables, I recorded the values of accuracy, positive precision, negative precision, positive recall, and negative recall for each model and each level of pre-processing in order to get a better look at the impact of the feature strategies and the meaning behind the accuracy value. "Positive Precision" shows the percentage of positive data entries categorized correctly with positive polarity. In turn, the higher the positive

precision value, the smaller the value is for false positives for the positive polarity category. "Positive Recall" shows the percentage of positive files being categorized as positive. This means that there are very few false negatives in the positive category. "Negative Precision" means that any file determined to have negative polarity has that percent chance of being correct. As a result, this means that the complementary percentage describes the percentage of false positives. "Negative Recall" is the probability that negative pieces of data are categorized with the incorrect polarity. This means that the higher the value for negative recall, the lower the value of false negatives for the negative label.

In Tables 4-6, the bolded values represent the lowest performance value for each level of pre-processing. It is interesting to note that the lowest values for both the Decision Tree model and SVM model are recall values, meaning that the calculating the correct polarity for a given data is their greatest weakness.

Table 4. Naïve Bayes on Different Levels of Pre-Processed Data

Accuracy	Fully Pre-processed	Stop-words (Unfiltered)	Unlemmatized
Accuracy	0.89205	0.89682	0.89799
Positive precision	0.82755	0.89682	**0.83594**
Negative precision	0.98988	**0.83396**	0.99025
Positive recall	0.99195	0.99071	0.99203
Negative recall	**0.79215**	0.80110	0.80395

Table 5. Decision Tree on Different Levels of Pre-Processed Data

Accuracy	Fully Pre-processed	Stop-words (Unfiltered)	Unlemmatized
Accuracy	0.88192	0.87928	0.883458
Positive precision	0.88714	0.88056	0.88341
Negative precision	0.87718	0.87891	0.88359
Positive recall	**0.87577**	**0.8789**	0.88372
Negative recall	0.88807	0.87967	**0.88320**

Table 6. SVM on Different Levels of Pre-Processed Data

Accuracy	Fully Pre-processed	Stop-words (Unfiltered)	Unlemmatized
Accuracy	0.95236	0.95198	0.95332
Positive precision	0.95162	0.94865	0.95737
Negative precision	0.95323	0.95574	0.94944
Positive recall	0.95328	0.95613	**0.94833**
Negative recall	**0.95143**	**0.94783**	0.9578

Discussions and Conclusions

After completing this task of building a sentiment classifier with different models, I have learned a lot and discovered some things that I would like to change or include in the future. I would first experiment with using a bigram or trigram model as the baseline for this task, as those models take into account the relationship words in close proximity to each other, instead of individually, like the Bag-of-Words or unigram models do. I would also like to leave in the numerical ratings each review had as features and take those into account when determining the

polarity of a complete review, since the relationship between polarity and ratings among individuals are variable. In addition, I would only remove stop-words of low frequency, as sometimes they do contribute to the sentiment of a piece of text. I also learned through reading the Michael Jackson article that there exist lists of words based on such specific tasks and did not cross my mind at all when I was preparing for this task. If I could expand on this part, I would include a list of words, something like descriptive words in the film and entertainment industry, allowing for a better grasp of polarity in movie reviews

I have learned that every type of classifier has its strengths and weaknesses, and taken advantage of, could be used in combination with others to yield. The potential seems so incredible, considering just using these models individually already produce results very close to one-hundred-percent. This task could also be expanded into running related classifiers on the same model and comparing the performance results to see in action what the differences between such closely related models would reveal. This could possibly help in developing even new classifiers that combine crucial elements of different ones together. Another thing that this task could use might be the inclusion of more lexical resources, as many cultures have intersecting languages and forms of writing, such as shorthand abbreviations and emoticons, as mentioned in the article on short informal texts. In order to account for all of that, there needs to be more lexical resources to process that kind of data. Lastly , the biggest challenge I can think of that this task needs is how to pre-process in an optimal manner to minimize noise but also keep all relevant features while testing on a maintainable-sized dataset. A possible solution to this would be to test as many combinations of pre-processing feature strategies as possible and analyzing the results of that.

Citations

1. 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
2. Kim, Elsa and Sam Gilbert. “Detecting Sadness in 140 Characters: Sentiment Analysis and Mourning Michael Jackson on Twitter” Web Ecology Project, August 2009. http://www.webecologyproject.org/2009/08/detecting-sadness-in-140-characters/
3. Kiritchencko, Svetlana and Zhu Xiaodan and Mohammad, Saif M. “Sentiment Analysis of Short Informal Texts” National Research Council Canada, August 2014. http://saifmohammad.com/WebDocs/NRC-Sentiment-JAIR-2014.pdf