LING 406 Spring 2020 Term Project

Introduction

The goal of Sentiment Analysis is to determine the overall emotion and sentiment from a given text data. Sentiment analysis is extremely useful as it allows us to get a public opinion behind certain topics.

There are many benefits and use cases for Sentiment Analysis. Personally, I have seen it being used in a workplace setting for product analysis and market research. Companies often utilize sentiment analysis to gather a public opinion on certain products and ideas. These metrics would allow the company the understand whether the overall reaction towards their idea or product is good or bad. Prior to this, more costly techniques would be used to generate this feedback, through user-surveys and more traditional market strategies.

Another benefit is that sentiment analysis can be done on large datasets. Humans are generally very good at detecting emotion within text, but a task like can become extremely tedious as the dataset grows in size. Designing, training, and utilizing a proper sentiment analyzer could potentially be a lot more efficient than hiring a real person to complete a task like this on potentially hundreds of thousands of reviews.

These are just some of the benefits and use cases for sentiment analysis. Sentiment Analysis is extremely very capable and has a lot of potential to improve our world and its technologies.

Problem definition

Sentiment Analysis is the interpretation and classification of emotions within text data using computational linguistic text analysis techniques. The task is to find patterns within the text, design features and methods that can help us analyze the text, and to be able to classify whether a piece of text is considered overall positive/negative

My proposed system will take in both negative and positive movie reviews, user-written comments on certain movies, and train off of them. Once it is trained, it can then analyze the sentiment of these reviews or any text data source, and output a numeric value indicating whether the review(s) expresses positive or negative emotions.

Previous Work

I had completed tasks before with the Naive Bayes and the Decision Tree Classifier. I wanted to use a BOW model as my baseline, so I also researched a bit on novel implementation methods. I also took a look at certain articles on SVM and other NLP models that I may be using in this project. There are many great papers written on sentiment analysis, but I decided to first look at these three as I felt they implement interesting techniques that were relevant to our task at hand, and specifically relevant to my own implementation. These sources are all cited in my works cited section.

1. Co-training for Semi-supervised Sentiment Classification Based on Dual-view Bags-of-words Representation (2015), written by Rui Xia, Cheng Wang, Xinyu Dai, and Tao Li.

This was a fantastic paper that I came upon. As I am tackling this problem with a Bag-of-words (BOW) approach, I found this paper to be an extremely interesting read. One of the main

issues with the BOW approach is its inability to understand relatively complex linguistic structures such as negation. This a very common problem, where sentences like “It works well” “It doesn’t work well” are considered to be very similar with most statistical learning algorithms.

The authors attempt to solve this issue by implementing a “dual-view BOW” model. The dual-view BOW model analyzes both the original piece of text, and a modified “antonymous” review. The Antonymous review is created from the original piece of text. The original and antonymous views form two different views on one specific text. The authors therefore employ a co-training algorithm for its two separate views.

This is not the first paper that attempts a dual-view BOW model, but their specific techniques did prove to be more successful than previous attempts. The author discovers that having the antonymous view is very useful. On occasion, the antonymous view actually performs better than the original view, suggesting that removing the negations and reversing all sentiment could be more suitable in a BOW representation. Combining the results for both of them, then, would result in a very accurate BOW model.

I attempted to implement something similar to this in my own approach, but quickly realized how time-consuming this would be. I decided in the end to not add this in my own work. I realized that the more traditional BOW model is still relatively reliable. Though this implementation would possibly be very effective, the trade off in development time and runtime is just not worth it.

2. Using Hashtags to Capture Fine Emotion Categories from Tweets (2014), written by Raif M. Mohammad and Svetlana Kiritchenko

The second paper I looked at analyzed emotions from tweets. This paper was extremely interesting because of its in-depth exploration into emotions and its suggestions on how we can better analyze and define emotions. The authors trained their analyzer from a large corpus of tweets labeled with emotional hashtags(e.g. #angry or #sad), and also generated a lexicon with word-emotion associations from the corpus. They extracted what they call is a “fine emotion category feature” and found that their lexicon significantly improved performance of all classifiers.

What I found to be super interesting is that they utilized this lexicon for personality detection, which is the idea of detecting a person’s personality traits from the texts that they have written. This is often a really hard task, but it proves to be functional here with this implementation. Personality detection would be an interesting idea to explore in this task, as we know that emotions often have a relationship with personality. I think it would possibly be worth it to try and understand personalities in a medical sense. The paper explores more complex personality traits such as extroversion vs introversion, neuroticism vs emotional stability, and agreeability vs disagreeability. I think this explores a very interesting idea of training with user consistency in mind. It suggests that if a user always leaves a very negative review and leaves a less negative review, we could possibly weigh it differently. I think it’s possible that this wouldn’t make that big of a difference considering we are analyzing thousands of reviews and averaging the overall sentiment, but I can see it being very useful to predict sentiment on a more accurate scale.

3. Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media (2018), written by Isabel Cachola, Eric Holgate, Daniel Preotiuc-Pietro, Junyi Jessy Li

The third paper I read was on vulgarity. Vulgarity is a very common linguistic expressions, but its meaning has now changed. Understanding its usage and the context it belongs in can aid natural language processing applications. The usage of vulgarity can vary, but it does not always have to exist in a negative connotation. The authors of this paper identified the multiple emotions that could be expressed through the usage of vulgarity. They discovered that vulgarity is often used to intensify certain emotions, and utilizing vulgarity-centric features in their model induces increased sentiment analysis system performance.

I thought this was very interesting because quite a few of the movie reviews utilized vulgarity, though not very many of the Yelp reviews did. Vulgarity-centric features could prove to make our model more accurate when working with movie reviews.

Approach

My baseline approach began with utilizing the traditional bag of words model. I was familiar with this model and knew that it would work. The bag-of-words model is really simple to understand that implement and is usually very successful for language modeling and text classification. Here, each of the movie reviews would be tokenized, cleaned, and then vectorized accordingly.

I implemented several features for this:

1. Removing Punctation: This is a very common feature in pre-processing. Removing punctuation in this case would be beneficial as punctuation marks do not provide any useful information to our analysis.

2. Removing non-alphabetical tokens: I wanted to remove tokens that weren't alphabetical. This would be useful in removing metrics and other junk in the text that would not be useful in helping our analyzer.

3. Filtering out shorter tokens: This feature aims to eliminate the majority of single-character words. This data would be relatively pointless to capture, and often times is a result of a typo or a spelling error.

Then, I decided to implement one more features utilizing the Natural Language Toolkit (NLTK).

4. Removing Stopwords: This is another relatively common feature that has been used in the past for pre-processing text in NLP. Stop words are commonly used words (e.g. "the", "a" "in", etc) that does not provide valuable information to us during our processing. NLTK in python already has a list of stop words that they already provide, my sentiment analyzer reads this list and removes them from my dataset.

Consideration: Negation Handling: Negation handling deals with processing negation in a sentence. Negation is a very common feature that is often used in pre-processing, since it deals with sentences like "The movie was not that good". Traditional BOW models would not capture the "not" in that phrase, which could lead to misleading results in analyzing sentiment. With negation handling, we could identify negation words as negative and add it to our results accordingly.

Negation Handling is known to help improve the BOW model, but it some ways it could also harm it too, so I wanted to explore the effects of this feature. After reading some papers and some articles on negation handling (sourced below), I ultimately decided not to include this in my sentiment analyzer. Though my analyzer isn't highly context-relevant, accurate negation handlers can be very complex to implement. The fear I had was that this feature would be detrimental rather than beneficial. I think negation handling has the potential to help improve accuracy, but it would have to be very well thought out. A paper by Yun Ding explores the idea of different variations of negativity from the combination of words. After reading this paper I changed my mind on negation handling. I would think about pursuing a similar co-training implementation to what I found in the first paper in the previous works section.

I ended up choosing three machine learning classifiers for the movie review dataset. The three machine learning models that I pursued were all classification models. We had used Naive Bayes and Decision Tree in a previous assignment, so I chose those two plainly due to familiarity. Being more familiar with how they worked, I was able to focus more on designing features.

The Naive Bayes is a very popular classifier used in Data Science. It was my first choice as we have seen it already a few times in this course, and we understand its capabilities. There are definitely some drawbacks to using Naive Bayes, but it has three traits that made it my top choice. It is incredibly simple to build, it is relatively accurate, and it has a pretty low runtime.

The Decision Tree is another popular classifier often used in Data Science. We have also looked at this algorithm in past assignments in the course. Here we The Decision Tree, like Naive Bayes, is easy to build and has a relatively low runtime, but unlike Naive Bayes, it usually

falls behind in accuracy. This changes in specific cases, however, so I wanted to see how it would be impacted given my implement features.

The Logistic Regression is the last classification model that I wanted to try and implement. It is similar to linear regression but its number of outcomes is finite instead of continuous. As I was more unfamiliar with this model, I wanted to learn more about it through this project. More importantly, I wanted to explore whether this classifier could be as good or maybe even better than the Decision Tree Classifier.

I implemented the Decision Tree and Logistic Regression classifiers with SKLearn, and in coding this section, I also considered implementing the support vector machine. The support vector machine is known to be more complex and more accurate, and is easily implementable using SKLearn, but I decided not to pursue it. I would predict that it has the potential to be a very accurate model, but in my case, it would not be very efficient since my current computer is very slow. I think that the Naive Bayes and the Logistic Regression algorithms are enough to achieve a relatively fast, accurate result.

I would expect the size fo the various feature sets to directly impact the performance of my analyzer. What we have seen in class and in previous assignments is that too many features can be detrimental to performance, therefore I decided to include some more essential features, but made sure not to overcrowd everything.

Extra Credit 2:

For the Yelp reviews, I had to first classify all of them as stars. The term project assignment guide suggests to sort 3.5 stars into positive classes and the rest into negative classes. I thought this was interesting, but after looking into the dataset, I realized that 3.5 stars is a very

odd unit. Yelp does not contain 3.5 stars, so the majority of "3.5 stars" I could find within the data source were users stating that "This restaurant could have gotten a 3.5 but Yelp does not count half stars". The users would then round this number to 3 or 4. I decided to classify all reviews over 3 as positive and everything under and equal to 3 as negative.

I used the very same bag of words model, then tested it with features 1, 2, and 3, that I had. I knew my original implementation works well, so I also combined it with the three machine learning classifiers that I had chosen earlier. I chose to omit feature 4 since it proved to be detrimental during the analysis of movie reviews.

Looking at the Yelp dataset, I predict my features should be just as effective here as well. Having the stars is also very interesting as it provides context information, which is useful for sentiment. I wanted to explore the idea of utilizing these stars to better weigh some of this sentiment, but in the end decided against it in fear of it generating misleading data.

Results

To analyze my results, I use standard evaluation metrics for calculations. The table below contains the accuracy percentages for each feature that is added onto it.

Accuracy (%)	No features	Feature 1	Feature 2	Feature 3	Feature 4
Bag of Words	59.6	62.9	62.9	62.9	62.7
Naive Bayes	83.1	83.8	81.1	80.5	68.5
Decision Tree	49	50.5	50.1	50.9	49.8
Logistic Regression	80.8	80.9	80.7	82	80.3

I would consider my bag of words model to be relatively successful. Looking at the table, each feature would perhaps slightly improve accuracy for certain Machine Learning classifiers.

My first feature, removing punctuation, seemed to improve overall accuracy for all classifiers. The second feature slightly lowered accuracy all around. My third feature seemed to improve accuracy again with the exception of Naive Bayes. My fourth feature, however, caused a detrimental drop in performance. The different is only minor for the Decision Tree and Logistic Regression classifiers, but the Naive Bayes seems to really take a large hit herer.

I was not surprised to see my Naive Bayes algorithm performing as well as other algorithms. It is a good classifier for this. What was interesting, however, is just how much it was hit by the fourth feature. It began was always very performant, but its seems that the more features I add on, the worse its becomes.

My Logistic Regression classifier performed surprisingly well given the circumstances and I felt that it was a very successful model. My results show that it actually performs better than the Decision Tree Classifier the majority of the time. It almost rivals the Naive Bayes algorithm, and even beats it with some of the features.

The Decision Tree Classifier didn't perform very well, but this was to be expected. I think the Decision Tree Classifier doesn't work very well here. It's very possible perhaps this is due to the fact that my Decision Tree Classifier is from SKLearn, but the more likely reason behind this is just that it isn't a fitting machine learning model for this task.

From my results, it seems like the Logistic Regression classifier is the most successful classifier. The logistic regression classifier was fairly accurate to begin with, and as I added more feature it got even better. I think my Naive Bayes implementation its vey good, but it suffers a bit

once the number of features increases. Though it could be due to my selected features, it would not make too much sense that other classifiers were getting slightly more accurate whilst the Naive Bayes was getting less accurate.

Extra Credit 2:

Accuracy (%)	Feature 1	Feature 2	Feature 3
Bag of Words	75.9	74.9	75.9
Naive Bayes	58	64	65
Decision Tree	86.3	88.2	88.2
Logistic Regression	90.4	89.8	90.1

I was surprised to find that my results were slightly more accurate than above with the movie review datasets. I think this is very interesting because my features barely made a difference in this case. I think this was so successful because of the difference in datasets. I will address these differences more thoroughly in the discussion section, but it seems that the algorithms here are doing the majority of the heavy lifting. It's likely that in this type of text data, there doesn't need to be many implemented features, which actually would be a good thing. The bag of words model by itself does work relatively well, as we have seen in the past, and feature selection usually just helps guide it along better. I think this dataset perhaps is just more fitting for the bag of words model and for the other machine learning classifiers too. Perhaps it's the brevity, perhaps it's the use of a more formal prose in the dataset.

The Logistic Regression Classifier once again wins, which is actually quite interesting. Here, Naive Bayes seems to take a bigger hit, which is different to our results from above. Here, Naive Bayes takes the biggest hit, being less performant than even the baseline BOW model. I

think it's very possible that this is due to my own specific implementation, and I would wonder what this result would look like utilizing NLTK's Naive Bayes Classifier.

Discussions (Movie Dataset + Yelp Dataset)

Something interesting that I noticed from working with both movie review dataset and yelp review dataset is that the classifiers seemed to be more accurate for the yelp dataset. I wanted to understand this a little bit more, and so I started looking into the two respective tapes to look for differences. I think there were perhaps two major factors here that I could have taken better notice of and made changes accordingly for during preprocessing.

The first is that the majority yelp reviews were of a much shorter length, though not all reviews followed this pattern. A possible explanation for this could be perhaps people have less things to say about restaurants versus movies. When we think about it, a meal could possibly take an hour of ones time, whilst a movie could take up two-three hours of ones time, perhaps there is just more to say as it was a longer, more though-provoking experience. This would be difficult to resolve, but there are perhaps some features here that can take advantage of this.

The other possible factor is due to Yelp's reviews having a character limit of 5000 characters. This is very important because it forces users to give feedback in less than 5000 characters. 5000 characters, however, can still mean an estimate of 500-1000 words. My guess is that users feel pressured to give more concise feedback given the 5000 character constraint. This also seems to contribute to the length of the reviews.

If I did this again, I would likely explore some methods that have been implemented in other papers that address sentiment analysis on micro-forms of text data. A paper written in 2014, Sentiment Analysis of Phrases Within Short Texts, written by Ameeta Agrawal and Aijun An explores the idea of sentiment analysis in shorter blocks of text, such as SMS data or more

commonly, tweets. Implementing some of their techniques could improve accuracy for the yelp reviews, and reverse-engineering their techniques could also mean improving accuracy for the larger blocks of texts, like what we see in the movie review.

Something that I also discovered from peeking through the datasets is that often times people would explain the plot of the movie or quote scenes from the movie. This is largely unhelpful to our algorithm, as it therefore captures quotes from the movie into the model. If given the opportunity again, I would try and parse the text for these movie quotes and remove them from the training model.

The prose of the overall movie reviews also slightly differ from the the tone of the Yelp reviews. I noticed that the movie reviews seemed to have a form of casual prose to it, in comparison to the Yelp reviews. Certain movie reviews utilized a more conversational style of speech to it, in comparison to something that is more concise, proper, and straightforward. Users sometimes even begin diving into specific plots and employ techniques of rhetoric to provide a humorous aspect to the movie review. It also contains vulgarity, which seemed to be absent in the majority of the Yelp Reviews. It could be beneficial to implement some of the ideas from paper 2 and 3 in my previous works section for this.

Conclusion

I think my approach allowed me to achieve fairly accurate results. This is more than likely due to the reliable and successful base model that I based my analyzer on, but I think some parts of this could also be due to the features that I had chosen for this task. It also helped that I was able to disable some of those features for a higher accuracy on the yelp dataset.

I think the most valuable lesson I learned from working on this project is the importance of understanding the dataset. If I had spent a little more time digging into the data and understanding its “elements”, I think I would have been able to have a more accurate analyzer. This project taught me how important it is to select good features and preprocess.

What I realized is that different datasets have different characteristics to them that we can take advantage of. An example of would be the shorter length of the yelp dataset, or the more casual prose, almost “conversation-like” movie reviews. I think it is important to consider the context in which these users are writing these reviews. I think it would be very beneficial to employ certain ideas from design and prototyping into designing a sentiment analyzer. If I had examined potential user archetypes in both scenarios, I could have seen earlier that perhaps there is a large difference to where and when people are writing their reviews. If we examine potential user archetypes, we may find that some of the more lengthy movie reviews are done with the intention to dedicate a lot of time and effort into writing it, versus the shorter yelp reviews that might be completed in the time from after a customer finishes their food and the check arrives. The point is, not everyone might spend a lot of time reviewing a meal that they had, but a movie might be different.

If I had noticed these patterns in my original planning phase of this assignment, I could have not only saved myself a lot of time, but also build a much more accurate sentiment analyzer.

Works Cited

Agrawal, Ameeta & An, Aijun. (2014). Kea: Sentiment Analysis of Phrases Within Short Texts. 380-384. 10.3115/v1/S14-2065.

Brownlee, Jason. "How to Develop a Deep Learning Bag-of-Words Model for Sentiment Analysis (Text Classification)." Machine Learning Mastery, 7 Aug. 2019, machinelearningmastery.com/deep-learning-bag-of-words-model-sentiment-analysis/.

Cachola, Isabel et al. "Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media." COLING (2018).

Ding, Yun. "A Quantitative Analysis of Words with Implied Negation in Semantics." Theory and Practice in Language Studies 3 (2013): 1200-1208.

Liang, Xu. "Treat Negation Stopwords Differently According to Your NLP Task." Medium, Towards Data Science, 3 Aug. 2019, towardsdatascience.com/treat-negation-stopwords-differently-according-to-your-nlp-task-e5a59ab7c91f.

Mohammad, Saif & Kiritchenko, Svetlana. (2014). Using Hashtags to Capture Fine Emotion Categories from Tweets. Computational Intelligence. 31. 10.1111/coin.12024.

Nico, Gerard. "Machine Learning - (Univariate: Simple) Logistic Regression." Nicôme - The Data Blog, gerardnico.caom/data_mining/simple_logistic_regression.

Patel, Savan. "Chapter 2 : SVM (Support Vector Machine) - Theory." Medium, Machine Learning 101, 3 May 2017, medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.

Reitan, Johan et al. "Negation Scope Detection for Twitter Sentiment Analysis." WASSA@EMNLP (2015).

Xia, Rui & Wang, Cheng & Dai, Xin-Yu & Li, Tao. (2015). Co-training for Semi-supervised Sentiment Classification Based on Dual-view Bags-of-words Representation. 1. 1054-1063. 10.3115/v1/P15-1102.