Final Project: Twitter Sentiment Analysis

David Stern
Papa Nii C. Vanderpuye

CPS-360: *Machine Learning*

Abstract:

This project is based on studying the application of machine learning algorithms on Twitter sentiment analysis. For the project, we looked at ways of getting properties of tweet texts and experimented with Multilayer Perceptron, Naive Bayes and Decision Tree in figuring out text sentiment based on the attributes we predefined. This paper consists the various aspects of our experimentation, the method in which we decided to analyze texts, The different algorithms and the parameters we used to analyze the text, the evaluation of the results and accuracy of each algorithm, and finally the conclusion, suggestions, and improvements we decided to make for further research on the topic.

Introduction:

Sentiment Analysis is the process of using a machine learning algorithm to extract, identify or categorize the emotion behind a text. It is becoming a popular area of research as there seems to be a growing demand for it in social media research and data mining. The concept stems from the study of text mining, which is generally focused on identifying opinion polarity. It can be used to figure out if a text sentiment is positive, negative or neutral. It is can also be used for deeper evaluation, for checking if the sentiment is angry, happy, sad or anyone of the main human sentiments. While it's often not very accurate, it can still be useful. Companies use it to mine through reviews to check up on their publicity, and AI speech programs use it to figure out how best to respond to text posed toward them. We decided to research on this topic for our project because it is a topic that both of us have been interested in since we started to pursue the Computer Science major program. We wanted to figure out how exactly the system works to be as accurate as possible in examining texts, so that we can better understand how other sentiment analysis models over the internet perform their tasks. We chose twitter posts to analyze because we feel that it is one of the rawest forms of human communication, with less attention being made to punctuation and grammar. We feel that if we can be accurate with these kinds of text, our algorithm could potentially perform well with any other form of text.

Dataset:

Our main dataset is `text_emotion.csv,` of which we converted to a text file called `data.txt.` `text_emotion.csv` is a dataset containing 40,000 tweets total. The format in order is tweet_id, sentiment of tweet, author of tweet, and the tweet. The dataset classified tweets as one of 13 emotions. The emotions the tweets could be coded by were happiness, relief, fun, enthusiasm, love, sadness, hate, worry, anger, surprise, empty, neutral, and boredom. Even Though it was was a robust dataset it contained some balancing issues. For example, 8459 tweets were categorised as *worry,* while only 110 were categorised as *anger*. After poor initial results (around the 35% accuracy range in an MLP) and the previously mentioned balancing issues, we decided to reclassify the tweets as *positive* or *negative*. This was also done so that the data would reflect better with other language datasets we found online that typically use positive and negative as their classifier. When reclassifying we ended up putting all tweets classified as *happiness, relief, fun, enthusiasm*, and *love* as *positive* sentiments. After modifying this we obtained a total of 13,112 values classified as *positive*. We then classified *sadness, hate, worry, anger, boredom,* and *empty* as *negative* sentiments and obtained a total of 16,063 classifications. We eliminated the tweets classified as *neutral* and *surprise* to bring our new total to 29,175 tweets. We eliminated *neutral* because by definition it is neither *positive* nor *negative*. We eliminated *surprise* since it can be categorised as *positive* or *negative*. Though we experimented with other datasets, `text_emotion.csv` is the main one we used. An example of another dataset we experimented with was `newtest.txt.` This file contains 40,278 tweets that are coded number, positive or negative (1,0), source and tweet. We lightly used this dataset to make sure our results were not because of the dataset we were using and would match up with any tweet dataset. Overall the focus of our experiment was on the `text_emotion.csv` dataset.

Methods:

One of the problems we were trying to answer was what is the best way to turn language into numbers so that they could be properly coded. Starting with just plain language, we tried different combinations of different sentence stats. With these combinations we would run them through a simple mlp on weka to see if they had any promise whatsoever. If they did they would be included in the larger experiment if not they were eliminated. We did this preliminary screening because it was not efficient to try every combination of every property especially if it showed no sign of adding any benefit. As we will talk about later, one real run through testing took awhile and we would not have the time to run 100's of attributes through it, especially for **MLP**. For example, when comparing the amount of *positive* or *negative* words in a sentence we could choose to count each total or do a binary if *positive* or *negative* words appeared. Not only did the second option produce much worse results on an initial run, but it also didn't make sense in the context of some tweets. For example, "I'm not sad because today is a great day" would be 1 positive 1 negative. Thus we narrowed down the types and combinations of attributes we were testing to 10. They are separated by 3 subtypes word binaries, tweet stats, and combination of both. Word binaries contain binaries of any word that appeared in the first balanced 1,111 tweets (101 each for the remaining 11 original attributes that make up positive and negative) in over 10 tweets. As they went up levels in the binary we would be testing more and more word pairs or pairs of words with each level containing the complete binary of the level below (ex if "that is" word pair was included "that" and "is" separately would also be included). The next type is word stats. These contained methods such as punctuation counts, positive and negative word lexicon counts, number of characters, ect. The lexicon counts were counting the amount of positive or negative words which appeared in positive and negative lexicons developed during a linguistic study of word sentiment. We used the count method when coming up with the value for tweet stats because it would be generous hopefully accounting more for misspellings and bad grammar. This though leads to a higher complexity when converting the tweets to numbers as it will always be stuck at O(n) and in its worst case where as the binaries can end early as they use in. This leads to much shorter build times for the binary based approach. The final type was a combination of both. These would contain the word binaries and some of the more impactful stats that we found.

Experimental design:

The experiment was to take the methods chosen above and put them through 10 fold (this is what weka defaulted to) cross validation to compare their average mean squared error, its standard deviation, minimum, and maximum error. We also included average accuracy. This was put through a **MLP** on weka. We standardized the **MLP** to always have 5 hidden layers and only train for 100 iterations. This was done not because they produced better results but because some of the methods would take too long to finish otherwise as they had a large amount of inputs. For each method we would run **MLP** 10 times to get the results using the random seeds 0-9 on all of them for consistency. We also ran **Naïve Bayes** and **Decision Tree** over each of the methods to see how its accuracy was affected by different attributes. After which we compared a better (200 iterations over 100 trained) MLP of one of the word pair binary + lexicon results against **Naïve Bayes** and **Decision tree**.

Results:

Attribute Experimentation

Model	single word binary	word pair binary	word tri pair binary	single word binary +lexicon	word pair binary +lexicon	word tri pair binary +lexicon	punc	punc + lexicon	punc + lexicon+ word count and other tweet stats	lexicon ratio + other stats
MLP mean	0.3949	0.3952	0.3959	0.3716	0.3738	0.3755	0.4797	0.4291	0.4152	0.4217
MLP std	0.003	0.0045	0.0039	0.0037	0.0087	0.0034	0.0019	0.002	0.0017	0.0011
MLP min	0.3885	0.389	0.3887	0.3656	0.3657	0.37	0.476	0.426	0.4125	0.4196
MLP max	0.398	0.4033	0.3998	0.378	0.3934	0.381	0.4823	0.4326	0.4183	0.4232
MLP Avg accuracy	0.6917	0.6925	0.6889	0.7157	0.7161	0.7181	0.5848	0.6546	0.667	0.6571
Naïve bayes accuracy	0.6888	0.6896	0.6894	0.7174	0.7174	0.7177	0.5669	0.6458	0.6518	0.6573

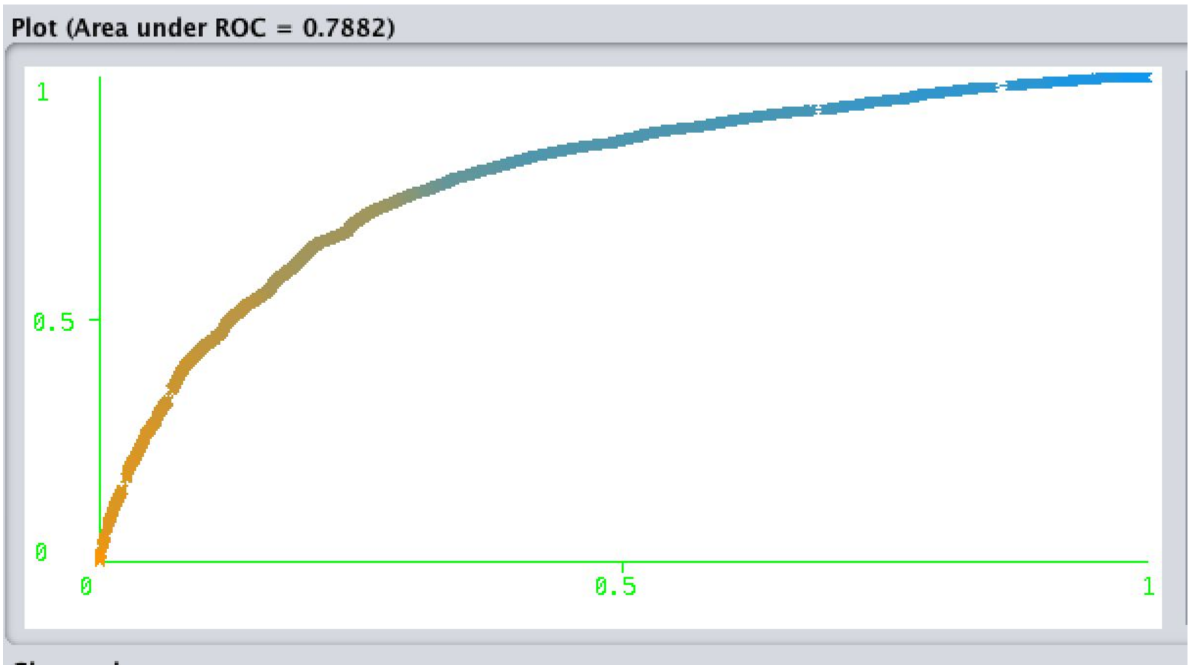
Attribute Confusion Matrix

Attribute	TP	TN	FP	FN	Precision	Sensitivity	MCC	ROC	Accuracy
S word	9200	10611	3912	5452	0.684	0.679	0.360	0.752	67.904
Bi word	7493	13059	3004	5619	0.706	0.704	0.398	0.753	70.4439
Tri word	8490	11705	4622	4358	0.692	0.692	0.377	0.753	69.2202
lexicon	9213	9169	3899	6894	0.643	0.630	0.274	0.685	63.006
lex+punc	8936	9961	4176	6102	0.655	0.648	0.300	0.701	64.7712
Bi-word +lex+punc	8868	12320	3743	4244	0.726	0.726	0.445	0.789	72.6238

Algorithm Experimentation

Model	TP	TN	FP	FN	Precis ion	Sensi tivity	MCC	ROC	Accuracy
Naïve Bayes	8028	12920	3143	5084	0.718	0.718	0.426	0.769	71.8012
Decision tree	8433	11490	4573	4679	0.683	0.683	0.359	0.673	68.2879
MLP	8868	12320	3743	4244	0.726	0.726	0.445	0.789	72.6238

MLP ROC Curve



ROC plot of MLP using bi-words, punctuation, and lexicons.

Analysis:

For the word binaries the added layers of word groupings did not have any real effect on the results of the **MLP.** The results for these three matched across the board when averaged out. This points to Single word binaries being the best of the word binaries, as it would contain the least amount of attributes. Of the tweet stats based methods use of a *positive* or *negative* word lexicon proved to have the greatest impact combining it with other statistics was able to slightly boost up performance in terms of accuracy and mean squared error while lowering its standard deviation. The increase in attributes is negligible considering that it is still in the 10's while the binaries are around 200. The best results though came from the combinations. The combinations were able to take the word binaries and improve them lowering their mean squared error by 2% and raising the accuracy into the 70% range. When looking at the confusion matrix it makes sense as the binaries approach favored predicting negative while the stats based approach favored positive more, meaning they were favoring opposite approaches. **Naïve Bayes** mostly mirrored these results. **Naïve bayes** and **Decision Tree**, when compared directly with **MLP**, all using bi-word +lexicon + punctuation, were outperformed. **Decision Tree** was the worst of the three with the lowest accuracy. **Naïve Bayes** performed well and it can be argued that it is the best approach because of its much lower training time complexity. That being said **MLP** can still improve as the best settings were not discovered and it was improving just from raising the amount it trained on before it stopped. It also performed the best and after it is trained the time it took to make a prediction was not long at all.

Application(Emotion Predictor Game):

After our experiment, we got MLP model by training MLP on the attributes we defined through the whole dataset, we then then we exported the model and used it to code a game that predicts the sentiment of a sentence the user enters, whether *positive* or *negative.* This can be found in the project folder.

Conclusion:

This study tells us that high attribute word binaries are the way forward to continuing to improve language processing. While the binaries did not improve on one another, it provided a start which had the most significant effect on results of any type of method. The effectiveness of word binaries would probably increase if we looked for more high impact common words, as some words such as *"the"* provide no real insight. Along with this, any other type of single word group was vastly overwhelmed by the number of single words, probably hindering its effectiveness. While **MLP** can be argued to be the best, it was shown that **Naïve Bayes** performed very comparable to it with a much smaller complexity time. Future projects could point to Naïve Bayes being more effective as it will be better at adding a higher number of attributes in the future. One aspect that we tried to implement was the analysis of emoticons. To experiment, we created an attribute that checked if a positive emoticon was present in the tweet and an attribute that checked if there was a negative one. This addition did not seem to change the results because emoticons like ":)" or ":(" appeared in less than 10% percent of the tweets dataset. We, however, think that these additions are relevant because in the real world emoticons are used much more frequently than what is represented in our dataset. This means that, in the future, we will aim to find a more robust tweet data set that is more reflective of the current generation. The sentiment analysis method we created can be applied to other problems apart from tweets. Using the game we created from our trained **MLP** model, we experimented with other inputs like movie reviews and Amazon product reviews. Our program got around 90% of the inputs we fed it correct. It should also be mentioned that humans in general only agree around 80% of the time when evaluating the sentiments of texts themselves because there is always a discrepancy based on the situation or context each statement is said. For instance, the tweet "It's a snow day" could be categorized differently if a high school student said it, then say the high school Superintendent. That being said, the fact that we were able to get our method to be above 70% accurate for tweets is a big achievement for us.