Lab 2: Sentiment Analysis of Microblog Streams

Jinghu Lei

March 4, 2019

1 Introduction

Sentiment classification has become a prevalent topic because of its various usages for companies. In this assignment, I am given a basic sentiment classifier (sentiment analyzer) and asked to modify it with additional features. The base model achieves a precision of 0.7112 and recall of 0.7461 for a F1 score of 0.7283. My improved model achieves a precision of 0.8359 and recall of 0.8333 for a F1 score of 0.8346.

2 Program Structure

2.1 Preprocessing

For the initial preprocessing that was given, many of the stemming and tokenization that is normally done was removed for the sentiment classifier to work. For my improved classifier, I decided to try out both Support Vector Machines as well as Multinomial Naive Bayes. However, I believed it would be useful to still have the given classifier be used as an additional feature, like a stacking technique with ensemble methods. So, for the original (sentiment analyzer) classifier, I tokenized it and removed stop words. For the SVM and Naive Bayes, I did the same but also removed punctuation from the string.

2.2 Predictions

For the improved prediction portion, I first created three different pipelines for a feature union. The first being the TF-IDF features, which would take the completely preprocessed tweet. Secondly, I had the sentiment class scores (e.g. positive, negative, neutral, compound) which would take an alternative version of the tweet with its punctuation. The third, are metadata and contextual parts of the tweet such as the description, follower and friend count, and favorite and retweet count. My approach consisted of switching between as well as trying out different combinations of these pipelines to find the optimal. Once the features were unioned, a feature selector is used to select out the top x percentile of useful features. Then the model is put through the 10-Fold cross validation to report out an average score.

3 Training and Testing

To obtain reports for each class so I could see any flaws of necessary features, I introduced three more class specific averages for precision and recall. Thus, at the end of the execution you can see the averages for each class instead of just the total. This proved very useful as I was seeing how well each feature benefited or made worse each class. My testing and training strategy was to test each feature individually and see how they affected the model. Then look at the individual scores and see if combining features might solve each other's flaws. After each addition and removal, another round of tuning was done by simply increasing and decreasing the hyper parameters and observing how the F1-score changed. After each change the results were tabulated into an excel sheet (*data.xml* in the zip file).

The results of the initial sentiment classifier, SVM, and Multinomial Naive Bayes are shown in table 1.

Table 1: Initial Outputs

Classifier	$F1$ Score
Sentiment Analyzer	0.72826
MultinomialNB	0.77007
SVM	0.81017

4 Feature Exploration

4.1 Sentiment Scores

It was clear that the sentiment scores were insightful information that should be used in the classifiers. At first it wasn't clear how this would be possible because the given classifier did not act like a normal fitted classifier. I decided to add it into both the support vector machines and multinomial naive bayes through a feature union. The results of the normal fitted classifier would output four scores: positive, neutral, negative, and compound. I would use my additional classifier to build/stack onto the results of these in addition to the TF-IDF features it obtains. Through research, I realized this score had a dual functionality. The positive/negative/neutral scores were representations of what percentage the tweet was for that class. Secondly, the compound score led to label prediction as a combination of the three. By adding these into my own classifiers, it provides another vote for what the correct label should be. The results of this action can be seen in figure 1.

4.2 Ensemble Methods

Observing the individual class precision and recall scores for each of the three classifiers I tried led me to try an ensemble method. As can be seen from table 2, the results of the three classifiers are diverse and have different strongest classes. I chose to use a Voting Classifier with the Multinomial Naive Bayes and SVM as well as the sentiment class results

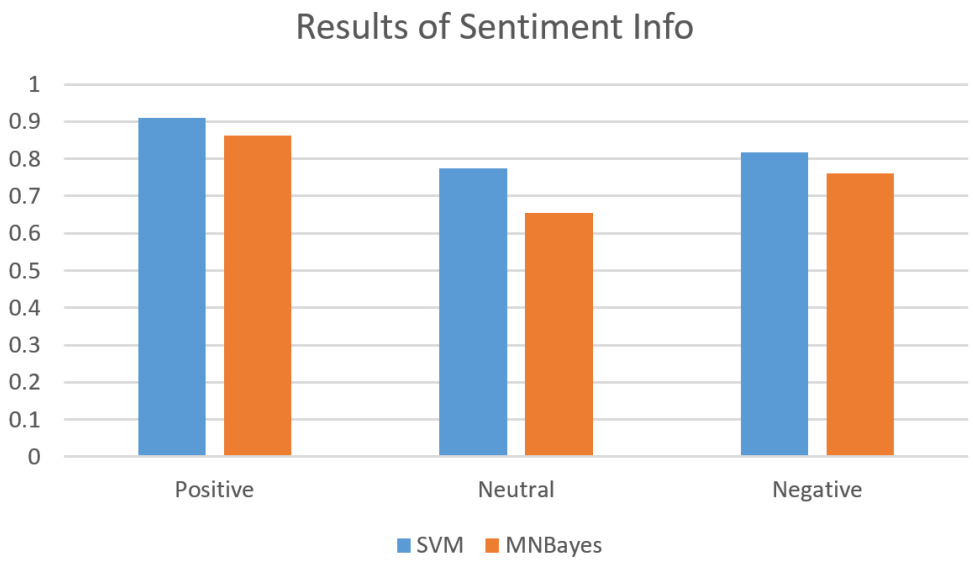

Figure 1: F1 Class Results of Sentiment Scores

from the sentiment analyzer as an additional feature vector. This would hopefully meld the weaknesses and create a more balanced classifier. The results of the new ensemble method can be seen at the bottom of 2.

Table 2: Ensemble Classifier

Classifier	Class	Precision	Recall	F1
SVM	Positive	0.900483	0.854496	0.876886
	Neutral	0.686734	0.818578	0.74688
	Negative	0.883701	0.718351	0.79249
	Total	0.823639	0.797141	0.810173
Bayes	Positive	0.779223	0.96577	0.862525
	Neutral	0.798105	0.554005	0.65402
	Negative	0.79973	0.727222	0.76175
	Total	0.79235	0.748999	0.77006
Ensemble	Positive	0.901034	0.923745	0.91224
	Neutral	0.765834	0.75354	0.759637
	Negative	0.791758	0.762237	0.77671
	Total	0.81954	0.81317	0.816345

We can see that an improvement was made by placing the three models together in this way, albeit minor. It still shows that they balance each other's predicitions through the voting decision.

4.3 Friends and Followers

Friends and followers for the user was added into the each classifier and tested rigorously. I first tried them separately (late fusion) and then attempted first adding and normalizing them using a MinMaxScaler, which scales the number to the max number as a positive integer. However, it was seen that no improvement was made either way. This can be easily assumed due to the irrelevant nature of those numbers as compared to the details of the tweet. Thus, this route was not continued forth.

4.4 Retweet and Favorite Count

Retweet and favorite count seems like a promising feature. Observationally, the more retweets or favorites a tweet got, the more away from neutral it became. Simply put, more opinionated tweets get more traction. Similarly to the friends and followers, I tried both late and early fusion with these two features. The best results were from adding the two together and normalizing them using the same MinMaxScaler provided by Sklearn. Individual F1 results for the classes are shown in figure 2

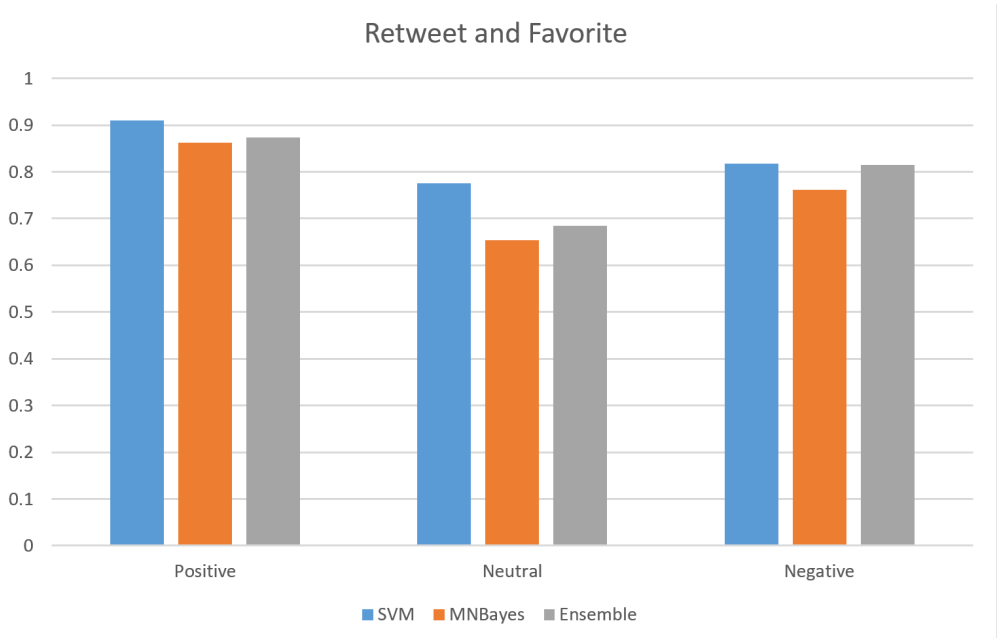

Figure 2: F1 Class Results of Retweet and Favorite Count

4.5 Part of Speech Tagging and Length of Tweet

Many words have different meanings depending on their position or role in a sentence. For example the use of a verb as a noun can lead to a neutral sentence instead of the natural classification of the verb. This lead me to think about part of speech tagging for each word before having it be processed in the Tf-IDF vectorizer. The result would be first labeling it using the nltk tagger, and then concatenating the result to the end of a word (e.g. cat as a

noun will be come cat_NN). Thus each word will be differentiated additionally by its place. Moreso, I also chose to test by adding the length of the tweet and trying combinations of the two features. The length of the tweet could benefit because it could be seen that longer texts tend to be more neutral because they consider both sides in their statement. Shorter texts tend to be more emotional and succint with less backing following their statement. After

the first few tests, it became apparent that the ensemble method performed worse than the SVM on most occasions. Thus, I decided to continue my tests without it anymore. The results of this are shown in figure 3

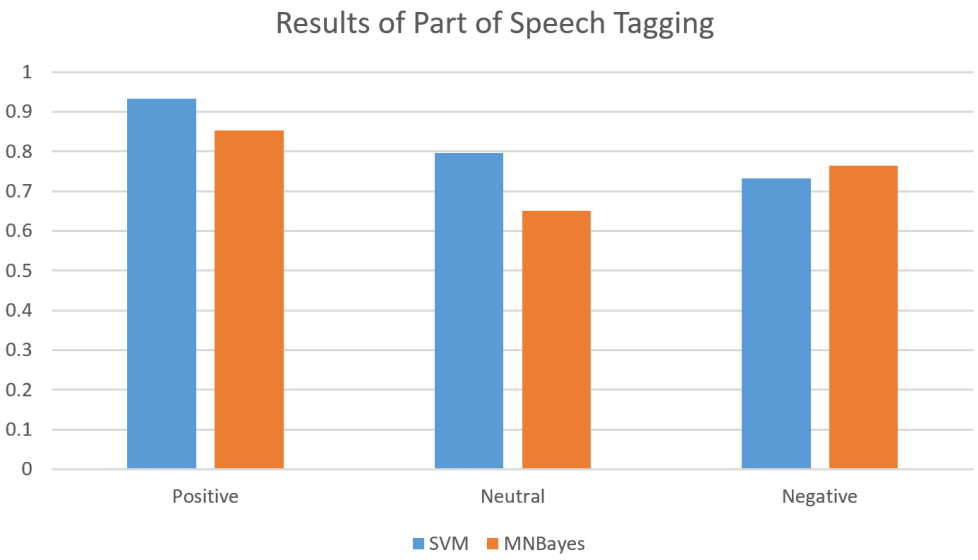

Figure 3: F1 Class Results of Part of Speech Tagging

Surprisingly it did not add that much benefit to the classifiers, and even adding worse performance to the neutral class. Perhaps by doing this tagging there are too many features or it is not as helpful because of the short nature of each tweet.

4.6 Bi-Grams

Bi-grams provide a more rich feature space if the two-word subsets of the space are useful for the classification. My assumption was that they would be because of the often pairing of words in expressing emotions (e.g. super bad). Unfortunately, this assumption was wrong and it actually led to worse F1 scores across both classifiers. I believe this could be since in addition to useful pairs of words, there are more useless pairings that can confuse the classifier (e.g. not so bad vs. so bad, the bi-gram will only see a part of the phrase). Results are tabulated in tabel 3

Table 3: Bi-Gram Results

Classifier	Precision	Recall	F1
SVM	0.84461	0.79114	0.81700
MNBayes	0.79459	0.70273	0.74584

5 Analysis

From the start, there was a skew in the results for each of the classes. Neutral was the hardest to predict because the scores tended to lean to either side. This could be due to the short nature of the tweets which provides the classifier less information to balance its decision. There are much fewer 'true neutral' words than negative and positive. I tried both Multinomial Naive Bayes and SVM in the experiment, however Naive Bayes fell short this time. Even will putting the two classifiers together with a voting mechanism, it seemed like the results of the SVM was dragged down by the opinion of the Naive Bayes. This can be seen from the graphs where the ensemble score was lower than the SVM. The features I have found to be most useful is simply the TF-IDF of the words in addition to the sentiment class scores from the base classifier. This ensemble sort of classifier had a higher F1 score than any other even with additional features. Even after tuning, it was still this model that outperformed the others. The best results of each feature is shown in table 4.

Table 4: Best Results for Improved Classifier

Feature	Precision	Recall	F1
Sentiment Scores	0.835936	0.833344	0.834637
Retweet and Favorite	0.835302	0.8327586	0.834028
Parts of Speech Tag	0.86131	0.803645	0.8314792
Bi-Gram	0.8446073	0.7911426	0.817001

6 Conclusions

The final improved classifier is a Support Vector Machine classifier with C=1.0, gamma=1.0, building ontop of the sentiment class scores of the original classifier (Sentiment Analyzer) and using TF-IDF vectors of the words in addition. This classifier produces a precision of 0.835936 and a recall of 0.833344 for a F1-Value of 0.8346379.