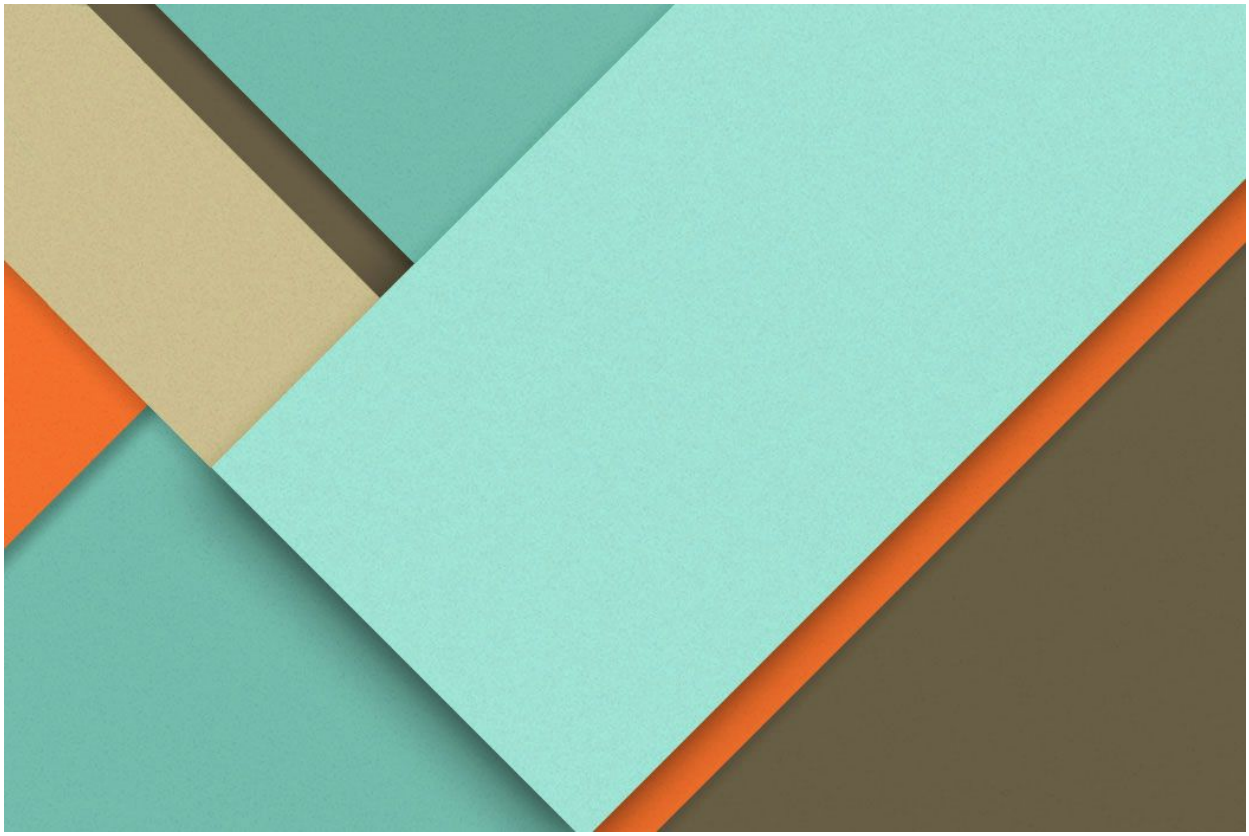

Sentiment Analysis of Microblog Data Streams

Steven Kester Yuwono (A0080415N)

Choo Jia Le (A0116673A)

1. Overview

The project aims to develop a classifier to classify input stream of tweets into multiple sentiment classes. The objective of this project is to find out the sentiment of the tweets with relation to one of these organisations, namely Microsoft, Google, Apple and Twitter. Our system utilises sentiment lexicons, both standard lexicons and newly learnt sentiment terms (which include internet slangs not listed in standard English dictionary). We explore the use of Naive Bayes, Maxent, and SVM to train our classifier and performance of each system is investigated thoroughly.

2. Implementation Details

2.1. Training Data

The training data used by the program contains tweets JSON together with theirs classified sentiments. Please refer to the training data summary below:

	# Tweets	# Positive	# Negative	# Neutral	# Irrelevant
Development	447	43	47	208	148
Training	2729	266	298	1256	908
Testing	1300	126	140	602	431

2.2. Information Extraction / Features List

The following fields within Twitter JSON are being utilised as features for classifier training purposes. They are listed as followings:

No.	Features	Descriptions
1	Text	The content of the tweet (with/without POS tag)
2	Contains Topic	1 if the tweet contains the topic, 0 otherwise
3	Positive Lexicon count	The number of positive lexicon present in the text of the tweet
4	Negative Lexicon count	The number of negative lexicon present in the text of the tweet
5	Retweets	The number of retweet for this particular tweet
6	Timezone	The timezone of the user's location when this tweet is posted
7	Friends Count	The number of users this account is following (AKA their "followings")
8	Followers Count	The number of followers this account currently has.
9	Favourites Count	The number of tweets this user has favorited in the account's lifetime.
10	User ID	ID of the person posting this tweet

Table 1. Feature List extracted from a tweet

2.3. Pre-Processing of Tweets

Before the text of the tweets are used for text classification, the text is pre-processed to remove noise in the data. The steps taken to pre-process the tweet is as follows:

1. Replace all newline and tab characters with space.
2. Use regular expression matching to generalize all URL mentioned in the tweet. (In other words, replaces http://t.co/someID urls with a standardize "[url]" string.)
3. Use Stanford Tokenizer to tokenize the text.
4. Convert all letters to lowercase.

2.4. Learning the model

We explored three machine-learning algorithms (classifier) namely:

1. Naive Bayes (Stanford CoreNLP)
2. Maximum Entropy (Stanford CoreNLP)
3. SVM (SVM-Light, multiclass)

Before exploring all of the features mentioned above. We tested each classifier with just the text (the tweet itself). Stanford CoreNLP is able to accept the text (any string) to learn the model. However SVM-Light is only able to accept real numbers features, not String features.

To overcome this problem, we have implemented tf/idf to be able to use the SVM classifier. The definition of the implemented tf/idf is as follows:

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

$$IDF(t) = log_e(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it})$$

We calculated the tf/idf value for each word in the text and use them as a feature.

After running the three classifiers with just the text feature. We found out that Maxent outperforms both Naive Bayes and SVM significantly. Therefore we proceed with **Maxent** as the classifier in our system.

The result of the three classifiers is shown in Table 3 in the Evaluation section (next section).

2.5. Learning new sentiment words/terms

Since we are using Maximum Entropy in our system, we can obtain the weights of all of the features (including each word in the text) contributing to each class (e.g. ["good",positive] = 0.3 ; ["slow", negative] = 0.2).

We utilize this property to get the top 1000 features (sorted in decreasing order of weights). And then print a list of possible sentiment words (positive or negative) from the learnt model.

We run the classifier on the development set and compiled a list of possible sentiment words which will be used to improve the system's performance further (Table 6).

Some of the new sentiment terms that we have found are as follows:

Negative	Positive
angst #palmface #ceo #fail malfunctioningagain wait #notcool #neednewipadguide #ripstevejobs #thenonsensepersists #fatfuckingchance n't wonky #fuckingpissed fucks wont whhyy eclipsed still humpt pissing #crankywithnophone	#awesome #prime power king biggest innovations lmfao singing telling pulling details simple upgrade replaced #genius #honest gratis

Table 2. Newly discovered positive and negative sentiment terms

3. Testing and Evaluation

3.1. Methodology

To develop and tune our classifiers, we use the development set given, with 10-fold cross validation.

However in this report, only testing results are shown. For testing, we used both training and test set combined together and 10-fold cross validation. We believe this is a better and more accurate measure than always using the same training set and test set.

For each fold, we obtain the TP (True Positive), FP (False Positive), and FN (False negative) count for each class (positive,negative,neutral, and irrelevant). To obtain overall score, we add all the counts from all of the 10 folds and the 4 classes. The evaluation score to assess the performance of the system will be F1-Score (Harmonic mean of precision and recall).

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

3.2. Results

The score shown in all of the tables below is the overall F1-Score (of all the classes and 10-fold cross validation).

Features	Naive Bayes	Maxent	SVM
Text only	0.4013	0.7892	0.2605
Text + POS Tag	0.3804	0.7758	-

Table 3. The performance of the 3 classifiers

POS tags are found to be not very useful from the result shown in Table 3. Therefore we used the text without the POS tag in further development and evaluation.

Features	Maxent Unigram	Maxent Bigram
Text only (1)	0.7892	0.7964
Text + Topic (1+2)	0.7894	0.7964
Text + Topic + PosLex + NegLex (1+2+3+4)	0.7951	0.8013
Text + RetweetCount + Timezone (1+5+6)	0.7892	0.7986
Text + UserID (1+10)	0.7941	0.7961
Text + Topic + PosLex + NegLex + UserID (1+2+3+4+10)	0.7964	0.8033
Text + FriendCount (1+7)	0.4932	0.5801
Text + FollowersCount (1+8)	0.4621	0.4636
Text + FavouritesCount (1+9)	0.4793	0.4857

Table 4. Maxent classifier's performance with combinations of features

Maxent N-Gram (N)	**Best features** Text + Topic + PosLex + NegLex + UserID (1+2+3+4+10)
1-gram (unigram)	0.7964
2-gram (bigram)	0.8033
3-gram (trigram)	0.8026
4-gram	0.8003

Table 5. Maxent classifier's performance with best features and N-gram

To improve further, we have manually looked through all the possible additional sentiment terms for both positive and negative lexicon, learnt from the development set. The result below shows the classifier with the best performance and the additional lexicons.

Maxent N-Gram (N)	**Best features + learnt lexicon** Text + Topic + PosLex + NegLex + UserID (1+2+3+4+10)
1-gram (unigram)	0.7991
2-gram (bigram)	0.8061
3-gram (trigram)	0.8041
4-gram	0.8028

Table 6. Maxent classifier's performance with best features, N-gram, and learnt lexicon

4. Conclusion

We noticed that the SVM performance is extremely low. It might be caused by our wrong usage of the SVM classifier. We tried to debug and use the classifier properly for some time but to no avail. Therefore we decided to move on with the Maxent classifier instead.

From the result shown in table 3, we observed that POS-tags do not improve our performance. Therefore it is not used in our final system.

After trying combinations of features, it is observed that only topic, positive lexicon count, negative lexicon count, and user ID are useful. The rest does not show significant improvement of the score when incorporated into the class as shown in Table 4. User ID is useful because some users tend to post either negative tweets or positive tweets quite consistently.

Retweet count and timezone is not really useful as the increase in performance is almost negligible when used. Friends' count, followers' count, and favourites' count is the least useful features, as shown by the significant drop in the classifier's performance when these features are used.

From the results above, it is also observed that the best classifier is the bigram classifier, showing the best result as compared to unigram, trigram and even 4-gram.