Sentiment Classification System for Twitter
A Natural Language Processing Approach

Maochen Guan (castor@castor.im) Tianshuo Deng (td859@nyu.edu)

0. Introduction

Programming Language: Java

To see the latest progress of this project, please send an email to: castor@castor.im

Website implementation: http://www.tweetemotion.com

1. ABSTRACT

As to the potential demanding in data analysis field, this project is served to evaluate the emotional coloring in the tweets gathering from twitter.com.

The system is implemented to automatically classify the sentiment of twitter message given a set of sentences which is semantically related to a certain topic. Tweets like "I love google but I hate apple" should be classified to positive group for topic google but should be thrown into negative group for topic apple.

Data that be fetched to the system are defined by the follow: Given a certain topic, tweets, those short sentences, can be fetched from some data providers. Then after passing through the system, it will be classified into 3 groups, positive, negative and neutral based on the distribution of the probability coming from the MaxEnt algorithm.

For example: "I hate iPad" is considered to be extremely negative comparing to "iPad is not good for typing". This kind of information is extremely useful for both consumers and manufacturers.

2.BASELINE SYSTEM [by Maochen]

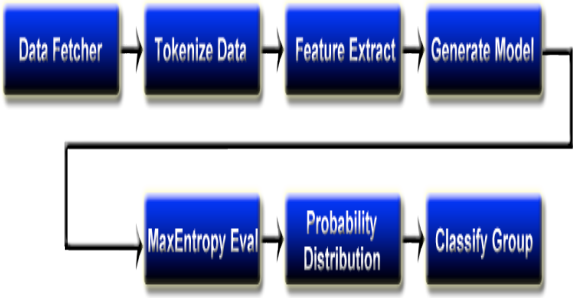
Baseline system is simply achieved from the count of positive and negative words in a sentence. An example of such system is Twittratr.

Here are the links in Twittratr that providing the positive and negative words they are using to classify tweets. The links are located down below this page

Negative words list can be found from (https://docs.google.com/Doc?id=df5m8zwp_93gd2mhkd7&pli=1)
Positive words list can be found from (https://docs.google.com/Doc?id=df5m8zwp_92gvtfm3d9&pli=1)

3. SYSTEM STRUCTURE

The following flow diagram illustrates the whole system's processing procedure:



Comprehensive unit tests are added in test driven development process in order to guarantee the robustness of the system.

4. APPROACH

4.1 Data collection

4.1.1 Data from Sanders Analytics

There is a free data set for training and testing sentiment analysis algorithms on Sanders Analytics. It consists 5513 hand-classified tweets. Each tweet was classified on respect to its own topic.

4.1.2 Data from tweetfeel.com

Data from tweetfeel are also be collected in order to compare and verify the result. Tweets collected from Tweetfeel.com are classfied as they claim.

4.2 Data Preprocessing[By Maochen]

Due to the natures trait of tweets, a kind of discursive structure compared with formal english, the collected data need to be preprocessed to cope with the parsing module so as to diminish the diverges and inaccuracy parsing. Following sections discussed the contributions done in the Data Preprocessing stage.

As to the tweet collected from [Sanders Analytics] and [TweetFeel.com], which is disorganized data, a series of method were taken to normalize the tweet sentence.

Tokenizer is the class responsible for text preprocessing:

1. Lower case all of the tweet data so as to recognize "Apple" and "apple" as the same topic.
2. All of the URLs are removed for URLs has no meaning in the parsing step.
3. Remove the "rt"(Retweet) keyword.
4. Word Completion of those abbreviated form like "can't", "don't" to normal form "can not", "do not", etc.
5. Word Completion of single character like "u" "r" etc.
6. Remove all of these non ASCII code characters.
7. Remove all of the @Tags.
8. Translate those quote tags to NN.

For example: "Uncle Sam" launched a new policy last week. will be translated to: "Uncle_Sam"->NN launched a new policy last week.

4.3 Feature Extraction

In this system, the result of sentimental analysis is enhanced by extracting the grammatical relations in the sentence.

1. Parsing

Stanford Parser is used to generate the parse tree for a given tweets, since the token is already preprocessed before entering the parser, the correctness of the parse tree is improved. For example, I am keen on iPad is parsed as follows:

```
(ROOT [51.097] (S [50.946]     (NP [3.877] (PRP I))     (VP [40.910]
(VBP am)     (ADJP [30.863] (JJ keen)         (PP [19.338] (IN on)
(ADJP [12.262] (JJ iPad)))))))
```

2. Dependency Generation

Typed dependencies provide us a simple description of the grammatical relationships in a sentence. This is helpful for feature extraction described in the following section. A example of typed dependencies is as follows:

```
rel:nsubj  gov:keen   dep:I rel:cop  gov:keen   dep:am rel:root
gov:ROOT   dep:keen rel:prep\_on  gov:keen   dep:iPad
```

4.4 Feature Selection

Tweets have several useful properties.
First, it's concise than usual sentence appeared in WSJ or Times.
Second, it's highly related to the hash tag, a tag that is frequently used to describe the topic of a tweet, that been used within the message.

In the system, two kinds of features are been captured, topic dependent features and independent features. topic dependent features are extracted from the parse tree of a sentence whereas topic independent features are coming from the text of a sentence.

4.4.1 Topic independent features

Topic independent features that been selected to apply to the system is due to the informality of the tweet message which increases the probability of the failure in parse tree generation step. The topic independent feature, a feature that is useful when a bizarre sentence is waiting to be processed, doesn't rely on the parsing of a sentence.

4.4.1.1 Negative Positive Word Count Feature.

In this feature, it will try to match every single word in the tweet to the positive word list and negative word list. Then count a normalized number.

Example 1:
"I bought a beautiful iPhone."
"beautiful" appears in the positive word list, and none of the word in the negative word list.

Total words appeared in the word list: 1
positive words: 1
negative words: 0
normalized positive word counter: pos_count_normalized=1.0
normalized negative word counter: neg_count_normalized=0.0

Example 2:
"Foolish and awful Google were fallen behind than beautiful Apple."
"beautiful" appears in the positive word list, and "foolish", "awful", "fallen behind" are included in the negative word list.

Total words appeared in the word list: 4
positive words: 1
negative words: 3
normalized positive word counter: pos_count_normalized=0.25
normalized negative word counter: neg_count_normalized=0.75

4.4.1.2 Emoji Feature.

This feature does the same operation as 3.4.1 Negative Positive Word Count Feature did. It will also generate the normalized positive and negative data by matching the tweet with posEmoji_list and negEmoji_list.

Example 1:
"I love Apple. :-)"
positive Emoji: 1
negative Emoji: 0
normalized positive Emoji counter: posEmoji_normalized=1.0
normalized negative Emoji counter: negEmoji_normalized=0.0

Example 2:
":(:< I love Apple. :-) :-)"
positive Emoji: 1
negative Emoji: 1
normalized positive Emoji counter: posEmoji_normalized=0.5
normalized negative Emoji counter: negEmoji_normalized=0.5

4.4.2 Topic dependent features

Topic dependent features captures more accurate analysis. There are multiple words in one tweet message and each of them represents positive/negative meanings. And most important, each of them modifies different object. To improve the accuracy of sentiment analysis, a parse tree of a given sentence will be generated and features will be extracted based on their grammatical/syntactical structure.
The Following are the features captured given topic #iPhone

- Transitive verb and Topic relation:
  given "I love iPhone", love_topic should be captured
- Topic and Transitive verb relative:
  given "iPhone rocks!", topic_love should be captured

- Adjective head is the topic, or say, Adjective directly modifies topic[By Maochen]
  given "I bought a beautiful iPhone" , beautiful_topic is captured
- Extract relative clause modifier:
  given "I bought an iPhone, which is beautiful", beautiful_topic is captured
- Extract Adjective+copula+topic:
  given "I am keen on iPad". keen_on_topic should be captured
- Extract Topic+verb+adj:
  given "IPhone runs smoothly comparing to android" topic_run_smoothly is captured
- Extract indirect depdency:
  given "Apple : Siri is amazing !"
  amazing_dep_topic is captured
- Use BFS to extract "subject of noun" relation:
  given "service of apple is rubbish ."
  "topic_nsubj_rubbish" is captured
- Extract propositional adjective modifiers
  given "Disappointing visual merchandising , apple"
  "disappointing_padj_topic" is captured
- Extract adjective modifiers for Noun phrase that contains the topic
  given "Wow , worst Apple customer service experience ever ."
  "worst_nnadjmod_topic" is captured
- Use BFS to capture the path from words of category "JJ", "JJS", "JJR", "RB", "RBR", "RBS", "VB", "VBD", "VBN", "VBG", "VBP", "VBZ" to topic.
  This features is important since it will capture features that are hidden more deeply in the parse tree.
  given "This is the first time I have been unhappy with any apple product ."
  "BFS_unhappy_prep_with_[NN]_nn_topic" is captured
- Capture bigram words
  given "This is the first time I have been unhappy with any apple product ."
  "prev_word_any","prev_prev_word_with" and "next_word_product" are captured

4.4.3 Negation handling

Negation should be handled to get a more accurate result. Negation is extracted from the parse tree of a sentence.

For example: given "I don't like iphone"
"neg_like_topic" is captured. The prefix "neg_" means negation

given "I bought an iphone , which is not good . "
"neg_good_topic" is captured

4.5 Maximum Entropy Modeling

4.5.1 Modification to the Original Wrapper

1. The original wrapper does not deal with features that contain real number as value. As to our approach, real number value is extracted to give some of the feature a different weight rather than match the whole feature string as original wrapper did.
2. Optimize the original wrapper's code, delete some unused functions and extract the common methods. Instead of reading and writing files every time in the prediction step, here, as to the generating model part, the output is a GISModel rather than output to the file every time. And

correspondingly, the input of the predicting part is a GISModel. Those data calculated from MaxEnt will be returned as a Map.

4.5.2 Use Modified OpenNLP Max Entropy package to calculate the most likely outcome.

4.6 Predict the Result

From the probability array generated from MaxEntropy jar program, choose the maximum probability under the three states to determine the pos/neg/neu Tag for this sentence.

5. System Test Approach

5.1 Cross Validation

- The whole dataSet (Approximately 5000 Tweets) is divided into 10 pieces chunks.
- Each time one chunk is taken out as predicting data, and the remaining 9 chunks are served as training data, launching the sentimental predicting system.
- Do the sentimental evaluation 10 times with different chunk as predicting data every time.
- Get the final accuracy percentage every time.
- Average Accuracy: Sum(Accuracy Percentage Per Time)/10

5.2 Testing Result

Approach: Cross Validation (10 rounds). Data Source: 5000 + TweetFeel

5.2.1 Baseline System

		Precision	Recall	F-Score
Round 1	Positive	63.97	63.50	63.74
	Negative	73.11	64.44	68.50
	Neutral	66.53	71.49	68.92
Round 2	Positive	60.00	59.15	59.57
	Negative	60.48	60.48	60.48
	Neutral	61.86	62.39	62.13
Round 3	Positive	65.56	63.87	64.71
	Negative	71.56	68.42	69.96
	Neutral	65.42	67.97	66.67
Round 4	Positive	63.77	61.11	62.41
	Negative	67.20	65.63	66.40
	Neutral	65.82	68.42	67.10
Round 5	Positive	60.26	69.12	64.38
	Negative	82.26	68.46	74.73
	Neutral	66.82	68.37	67.59

Round 6	Positive	64.43	68.09	66.21
	Negative	72.87	62.25	67.14
	Neutral	64.86	69.23	66.98
Round 7	Positive	66.91	61.49	64.08
	Negative	61.48	56.82	59.06
	Neutral	63.64	70.00	66.67
Round 8	Positive	63.24	57.33	60.14
	Negative	34.77	90.98	50.31
	Neutral	12.50	0.92	1.72
Round 9	Positive	57.24	57.64	57.44
	Negative	70.15	62.25	65.96
	Neutral	61.54	66.34	63.85
Round 10	Positive	65.94	67.41	66.67
	Negative	71.55	62.41	66.67
	Neutral	69.51	73.71	71.55

5.2.2 This Project's Enhanced System

		Precision	Recall	F-score
round 1	positive	77.10	73.72	75.37
	negative	74.10	76.30	75.18
	neutral	76.96	77.63	77.29
round 2	positive	69.48	75.35	72.30
	negative	64.54	73.39	68.68
	neutral	75.61	66.24	70.62
round 3	positive	76.71	72.26	74.42
	negative	70.07	78.07	73.86
	neutral	75.77	74.46	75.11
round 4	positive	79.05	81.25	80.14
	negative	73.76	81.25	77.32
	neutral	82.94	76.75	79.73
round 5	positive	74.24	72.06	73.13
	negative	84.06	77.85	80.84
	neutral	74.35	79.53	76.85
round 6	positive	73.43	74.47	73.94
	negative	80.69	77.48	79.05
	neutral	71.23	72.60	71.90
round 7	positive	77.04	70.27	73.50
	negative	75.74	78.03	76.87
	neutral	76.42	79.55	77.95

round 8	positive	72.50	77.33	74.84
	negative	66.89	75.94	71.13
	neutral	74.60	64.98	69.46
round 9	positive	68.75	68.75	68.75
	negative	76.92	72.85	74.83
	neutral	72.77	75.61	74.16
round 10	positive	68.87	77.04	72.73
	negative	73.85	72.18	73.00
	neutral	77.17	72.84	74.94

5.3 The Wilcoxon Matched-Pairs Signed-Ranks Test

The Wilcoxon Matched-Pairs Ranks test is a non-parametric test that is often regarded as being similar to a matched pairs t-test. The Wilcoxon Matched-Pairs Ranks test is used to determine differences between groups of paired data when the data do not meet the rigor associated with a parametric test.

http://www.fon.hum.uva.nl/Service/Statistics/Signed_Rank_Test.html

F-Score collected from part 5.2:

First column is the F-score of enhanced system, Second column is of baseline system

Positive:

75.37 63.74

72.30 59.57

74.42 64.71

80.14 62.41

73.13 64.38

73.94 66.21

73.50 64.08

74.84 60.14

68.75 57.44

72.73 66.67

W+ = 55, W- = 0, N = 10, p <= 0

Negative:

75.18 68.50

68.68 60.48

73.86 69.96

77.32 66.40

80.84 74.73

79.05 67.14

76.87 59.06

71.13 50.31

74.83 65.96

73.00 66.67

W+ = 55, W- = 0, N = 10, p <= 0

Neutral:

77.29 68.92

70.62 62.13

75.11 66.67

79.73 67.10

76.85 67.59

71.90 66.98

77.95 66.67

69.46 1.72

74.16 63.85

74.94 71.55

W+ = 55, W- = 0, N = 10, p <= 0

6. Conclusion

From the above tests, the result coming from our system is more stable than the baseline system as the variance indicated. Also, the system enhanced the accuracy by nearly 10% which is far more better than

original expectation. The difficulty of this project is that, since Tweets are not well-structured sentence which contains emotional tags, some informal abbreviations in contrast to WSJ or some academic papers, the Stanford parser failed to parse such sentences correctly. Due to this reason, the accuracy of the result is limited.

7. Reference

1. Target-dependent Twitter Sentiment Classification By Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, Tiejun Zhao
2. Deeper Sentiment Analysis Using Machine Translation Technology By KANAYAMA Hiroshi NASUKAWA Tetsuya WATANABE Hideo
3. Thumbs up? Sentiment Classification using Machine Learning Techniques By Bo Pang, Lillian Lee and Shivakumar Vaithyanathan
4. Shivakumar Vaithyanathan By Alexander Pak, Patrick Paroubek