Emotional Classification of Japanese Tweets with RapidMiner

Hiroshi Yamaguchi s1200239 Supervised by Maxim Mozgovoy

Abstract

Microblogs have radical spread among substantial number of people all over the world in recent years. Twitter is one of them, and it is a very popular tool among Japanese Internet users. They post short messages called ”tweets” on Twitter. They tweet their moods, opinions, share information and sometimes various silly phrases. Hence, Twitter offers many options for text classification. In this paper, I focus on studying which approach is better for emotional classification, experimenting with various combinations of vector creation, term extraction and machine learning algorithms. Experimental results show that such classification is possible, but there are still points that can be improved.

1 Introduction

Microblogs attract a large number of people in the world in recent year. Twitter is the leading microblogging platform, and it is especially popular in Japan for Japanese-language blogging. People use Twitter to share information, express their emotional state, discuss incidents, natural disasters, etc. Therefore, Twitter is ample source of data for potential text classification and mining. Hence, it is helpful source data for companies to market their products. They want to know how their products, service or news impress people. They can find out the reaction to new products from consumers on Twitter. Applying emotional classification to Twitter is an important task for companies to improve their service. There are existing research efforts of classifying documents using machine learning algorithms in English [1] [2].

In this paper, I investigate which approach is able to get better result by testing different combinations of vector creation, term extraction and machine learning algorithms to build a classifier for emotional classification of Japanese documents. I used Twitter corpus [3] and RapidMiner [4] to experiment.

2 Experiment

I have a corpus which contains 3852 tweets marked with eight emotions: as *anger*, *disgust*, *sadness*, *surprise*, *fear*, *happiness*, *pride* and *embarrassment* (see Table 1). I experimented with two kinds of emotional classification. The first experiment is to create a classifier for categorizing tweets into happiness and non-happiness categories. The happiness emotion was selected because text classification needs big datasets, and the happiness emotion is the most widespread emotion in our corpus (so there is a considerable sample of tweets marked with ”happiness”).

Table 1: Organization of corpus

	number of tweets
anger	1.7%
disgust	2.0%
sadness	6.5%
fear	0.6%
happiness	11.1%
pride	1.6%
surprise	5.0%
embarrassment	0.5%
non-emotion	72.7%

The second experiment is to categorize tweets into positive and negative. The positive group included the tweets marked with happiness and pride emotions. The negative group included the tweets marked with anger, disgust, sadness and fear.

2.1 RapidMiner

For the experiments, I use RapidMiner, which is an open source predictive analytics tool (free community version). RapidMiner includes a number of machine learning methods, enough for our purposes. It basically requires no coding, and users just drag and drop the necessary operators that process the data. We can analyze the data without writing machine learning algorithms, and use different options of operators quickly and easily.

2.2 Experimental setup

I formed the two groups of tweets as described above. The first group includes happiness- and non-happiness-marked tweets. There are about 11% of tweets marked with happiness and about 73% marked with non-happiness. ”Non-happiness” here means all tweets of the dataset, excluding tweets marked with happiness. I put all ”happiness” tweets into one folder and all the rest into another folder. There are 423 ”happiness” tweets, so I prepared seven folders for ”non-happiness” tweets, each including 423 ”non-happiness” tweets as well, to experiment seven times in each combination to get average results.

The second group consists of ”positive” and ”negative” tweets. I prepared 389 tweets for each group. The number of tweets is smaller in this experiment, because we do not have more tweets in our collection marked with negative emotions. Each set was placed into a separate folder.

2.3 Term extraction

Extracting terms from documents is a fundamental task in natural language processing, and we have to do some preparations to extract terms from Japanese tweets before doing experiments in RapidMiner.

First, I performed filtering. Each tweet is composed of a username and a content line such as ”@sample 鍋料理で楽しい一家団欒♪カニなんか最高ですね！”. We need only the content of tweets, so that we have to remove Twitter usernames and English words. I split tweets to elements by spaces, and removed the terms having as the first character an English or non-letter symbol, so that the usernames are completely removed.

Next, we need to consider how tweets segment to terms. I adapted several methods based on n-grams and morphological analysis. An n-gram is an n-character sequence of document content. Morphological analysis allows to divide the documents at the morpheme level. It is possible to segment documents with n-gram in RapidMiner, but there is no built-in function for morphological analysis, so I used MeCab [5] for this work. Mecab is an open source morphological analysis engine. I prepared five tokenized collection for the experiments, using the following elements as tokens: unigrams, bigrams, trigrams, morpheme unigrams and morpheme bigrams.

The following Japanese texts are examples of each tokenization after filtering. The character ’＄’ is used as a split character.

original text after applying filtering

鍋料理で楽しい一家団欒♪カニなんか最高ですね！

unigram of character unit

鍋＄料＄理＄で＄楽＄し＄い＄一＄家＄団＄欒＄♪＄カ＄ニ＄な＄ん＄か＄最＄高＄で＄す＄ね＄！＄

bigram of character unit

鍋料＄料理＄理で＄で楽＄楽し＄しい＄い一＄一家＄家団＄団欒＄欒♪＄♪カ＄カニ＄ニな＄なん＄んか＄か最＄最高＄高で＄です＄すね＄ね！＄

trigram of character unit

鍋料理＄料理で＄理で楽＄で楽し＄楽しい＄しい一＄い一家＄一家団＄家団欒＄団欒♪＄欒♪カ＄♪カニ＄カニな＄ニなん＄なんか＄んか最＄か最高＄最高で＄高です＄ですね＄すね！＄

unigram of morpheme unit

＄鍋＄料理＄で＄楽しい＄一家＄団欒＄♪＄カニ＄なんか＄最高＄です＄ね！＄＄

bigram of morpheme unit

鍋料理＄料理で＄で楽しい＄楽しい一家＄一家団欒＄団欒♪＄♪カニ＄カニなんか＄なんか最高＄最高です＄ですね＄ね！＄

2.4 Implementation

I performed two experiments for building a classifier on RapidMiner as follows.

First, we load the annotated tweet corpus to RapidMiner, divided into two classes: happiness class and non-happiness class.

Second, we need to apply tokenization. RapidMiner has numerous built-in functions (”operators”), in particular, for n-gram generation. For the case of unigrams, bigrams and trigrams of characters I apply such an n-gram generation operator to obtain the tokenized collection. For the case of morpheme units I had to load tweets that were already tokenized with MeCab.

Third, we set the options of vector creation operator. It creates a word vector from our tokenized collection. A word vector expresses features of our input data, which will be used in machine learning algorithms. I used three different vector creation methods: *binary term occurrences*, *term frequency* and *tf-idf* [6]. Binary term occurrences mean the fact of presence of the term in the document (a Boolean value). Term frequency is the weight proportional to the number of times the term occurs in the document. TF-IDF is a metrics of ”term frequency-inverse document frequency”. Inverse document frequency reduces the weight of the term that occurs often, and increases the weight of the term that occurs rarely. TF-IDF is able to get the combination of both weight of term frequency and inverse document frequency.

Finally, the created word vector is used in the machine learning algorithms to create the model and evaluate the it. I used the following machine learning algorithms: support vector machine, naive Bayes, k nearest neighbor and decision trees [7].

To test the model a cross validation method is used [8]. It evaluates the accuracy of the model. It divides our tokenized collection into a training set and a test set. I selected ten number of validations. It means, divide our tokenized collection into ten same size and measure the model for each iteration. So it is repeated ten times to evaluate the model and get the average of evaluating. RapidMiner will show how the model performed.

3 Results

We obtained accuracy, precision and recall values from the confusion matrix generated by RapidMiner (shown in Table 3 Table 4, Table 5, and Table 6) [8]. A confusion matrix describes information about actual and predicted classification done by a classification system. [9] For example, Table 2 is a confusion matrix that is classified into positive class and negative class, I explain its component below Table 2.

We got the result of experiment such as Table 3. It is the one of the result as classifying happiness and non-happiness. This is expressing confusion matrix for a performance in combination with support vector machine, bigram of character unit and tf-idf of classifying happiness and non-happiness.

Table 2: Confusion Matrix

		Actual	
		Negative	Positive
Predicted	Negative	TN	FN
	Positive	FP	TP

- **TP** is true positive that the model correctly predicted positive class and the actual class is positive.
- **TN** is true negative that the model correctly predicted negative class and the actual class is negative.
- **FP** is false positive that the model incorrectly predicted positive class and the actual class is negative.
- **FN** is false negative that the model incorrectly predicted negative class and the actual class is positive.
- **Accuracy** is the percentage of total number of predictions which are correct.
- **Precision** is the percentage in case of predicted positive(negative) cases which are correct.
- **Recall** is the percentage in case of predicted positive(negative) cases which are correctly identified.

They are obtained with the following calculations:

$$\begin{aligned}
accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\
precision\ (positive\ class) &= \frac{TP}{TP + FP} \\
recall\ (positive\ class) &= \frac{TP}{TP + FN} \\
precision\ (negative\ class) &= \frac{TN}{TN + FN} \\
recall\ (negative\ class) &= \frac{TN}{TN + FP}
\end{aligned}$$

Table 3: Confusion matrix of SVM-bigram-tfidf

		Actual	
		happiness	non-happiness
Predicted	happiness	358	205
	non-happiness	65	218

Table 4: Precision and recall of SVM-bigram-tfidf

	Happiness	Non-happiness
Precision	63.59	77.03
Recall	84.63	51.54

According to the Tables 3 and 4 the value of accuracy is **68.09** for the combination of SVM, bigram, and TF-IDF methods. Most results in classifying happiness and non-happiness are similar. The recall value of happiness is always higher than of non-happiness, and the precision value of happiness is lower than of non-happiness (like in Table 4).

Table 5: The result of classifying tweets to happiness and non-happiness categories. Each value is the average accuracy over seven tests. Bold values indicate the best results in each combination of methods.

token	word vector	SVM	NB	K-NN	DT
unigram	TF-IDF	63.66	58.05	58.57	**56.53**
	TF	64.46	**58.44**	**60.31**	55.91
	BTO	**64.91**	56.48	59.44	54.78
bigram	TF-IDF	64.05	61.05	59.37	**59.99**
	TF	**64.76**	61.53	59.77	59.53
	BTO	64.34	**61.62**	**60.23**	56.75
trigram	TF-IDF	59.17	57.77	59.12	**58.48**
	TF	59.81	58.53	**59.73**	58.47
	BTO	**60.58**	**58.72**	49.48	56.78
morpheme unigram	TF-IDF	62.87	56.76	**59.99**	56.5
	TF	**63.37**	**58.06**	59.13	**56.70**
	BTO	62.39	57.23	55.26	54.92
morpheme bigram	TF-IDF	56.08	57.64	**59.01**	**53.75**
	TF	**56.60**	**58.03**	58.84	53.07
	BTO	56.28	57.80	48.97	51.58

Table 6: The result of classifying tweets to positive and negative categories. Each value shows accuracy. Bold values indicate the best results in each combination of methods.

token	word vector	SVM	NB	K-NN	DT
unigram	TF-IDF	71.59	62.20	69.81	63.87
	TF	**74.54**	**62.33**	**70.44**	**68.25**
	BTO	71.85	58.61	68.77	67.87
bigram	TF-IDF	79.82	71.46	72.99	64.77
	TF	79.82	**74.55**	**74.80**	63.50
	BTO	**80.59**	74.16	68.38	**65.69**
trigram	TF-IDF	76.47	**71.98**	75.83	**65.17**
	TF	76.48	71.84	**76.60**	64.26
	BTO	**77.37**	71.84	54.23	62.60
morpheme unigram	TF-IDF	78.79	70.30	**71.33**	**66.32**
	TF	**79.69**	**72.62**	68.25	65.81
	BTO	78.28	67.74	70.05	61.70
morpheme bigram	TF-IDF	70.17	68.50	**71.71**	**58.87**
	TF	**70.69**	**69.02**	71.21	57.85
	BTO	68.37	**69.02**	53.73	57.33

4 Conclusion

In this research, I investigated how various combinations of methods contribute to the performance of model for classifying Japanese tweets. Classification tweets into the categories of happiness and non-happiness are not very reliable, while classification into positive and negative tweets show relatively high accuracy. The latter classification produced better results despite the small size of our dataset. We can know the best results from Tables 5 and 6. I could get accuracy **64.91** in classification of happiness/non-happiness and **80.59** in classification of positive/negative. Summarizing the results, we can highlight the following points.

- Support vector machine algorithm and ensured the best performance in all experiments.
- The choice of n-grams or morphological analysis method had virtually no impact on the quality of classification.
- There is also not much difference in each vector creation algorithm.

5 Future work

For the future research, there are possible improvements that should increase the overall performance. First, we needed a larger annotated emotion dataset. Second, morphological analysis with Mecab is not always reliable. It cannot deal with unknown words, because it uses a dictionary to perform morphological analysis. If there are unknown words in the document, it will not process them correctly. Furthermore, it is not always able to find the boundaries of words in the document. These problems are hard to address and need more research.

Acknowledgements

I would like to thank Prof. Mozgovoy for his suggestions and help in my research.

References

[1] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, Vol. 10, pp. 1320–1326, 2010.

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment

classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86. Association for Computational Linguistics, 2002.

[3] Anna Danielewicz-Betz, Hiroki Kaneda, Maxim Mozgovoy, and Marina Purgina. Creating english and japanese twitter corpora for emotion analysis. *Knowledge Engineering*, Vol. 1(2), pp. 120–124, 2015.

[4] Rapid miner. `http://rapidminer.com/`.

[5] Mecab: Yet another part-of-speech and morphological analyzer. `http://www.mecab.googlecode.com/svn/trunk/mecab/doc/index.html`.

[6] Li-Ping Jing, Hou-Kuan Huang, and Hong-Bo Shi. Improved feature selection approach tfidf in text mining. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, Vol. 2, pp. 944–946. IEEE, 2002.

[7] Kjersti Aas and Line Eikvilm. Text categorisation: A survey. *Raport NR*, Vol. 941, , 1999.

[8] Evaluating the model. `http://docs.rapidminer.com/studio/getting-started/5-evaluating-model.html`.

[9] Confusion matrix. `http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html`.

[10] Rapid miner documentation. `http://docs.rapidminer.com/`.

[11] Sentiment analysis in rapid miner. `http://www.corequant.com/?p=1`.

[12] Twitter. `http://twitter.com`.