Project report on emoji prediction with feature-based methods

Shuo Liu, Jiarong Yu and Shuai Hao

Abstract Recently, emojis are widely used in the communication in social media like Twitter. Since the emojis have played an important role in the tweets, they become more and more important in the Natural Language Processing. This report focuses on the problem proposed in the paper of F. Barbieri *et al.*[1]. We use feature-based method to predict emojis based on text-based tweet messages. After we extracted the features we use several machine learning methods like random forest to validate the correctness of our method, compared with the benchmark. This report also shows the results of the evaluation. It seems that features we extracted only have a slight improvement compared with the benchmark. This report also proposed several ways to improve our projects in the future.

1 Introduction

Recently, short text messages combine with visual enhancements provided a nobel way of communication in social media like Twitter, Facebook, Instagram and so on. Those visual enhancements called emojis. Emojis have become one of the most popular communication form in social media. Since emojis sentiment are popular and meaningful[15] in Natural Language Processing (NLP). There are many studies focused on semantics and usage of emojis in social media [7, 8, 9, 10, 11, 12], or using emoji for sentiment analysis in twitter, using emojis for detecting irony [14]. However there are a few work interplay between text-based messages and emojis remians virtually unexplored and those work are related to deep learning. There are not feature based achievement in investigating the relation between text and emojis. Therefore our project aims to using feature based method to predict emojis based on text-based tweet messages.

2 Related work

Though it is a very interesting area of study, researches on emoji prediction the perticular task is quite few. One of the leading work is by Barbiere *et al.*, who are also organizers of the CodaLab competition[1], using deep learning method BLSTM for their model. Other works earlier or later also used the same deep learning method.

Other researches tried to detect the sentiments or meanings of emojis[10], or using emojis for sentiment or sarcasm ditection of tweets and other short messages of social networks[13, 14]. Some researches used skip-gram for word embedding and received good results[10]. Some of these researches in related topics are using feature-based classification methods[15, 13].

3 Approaches

3.1 Problem modeling

There are many ways of predicting suitable emojis for a short text message, *e.g.* a tweet. In this perticular task, the problem is modeled as a multi-classification problem, where only one emoji is attached to a tweet, and the position is irrelevant. Thus, to predict the emoji of a tweet is to classify a tweet with emojis as labels.

[1] https://competitions.codalab.org/competitions/17344

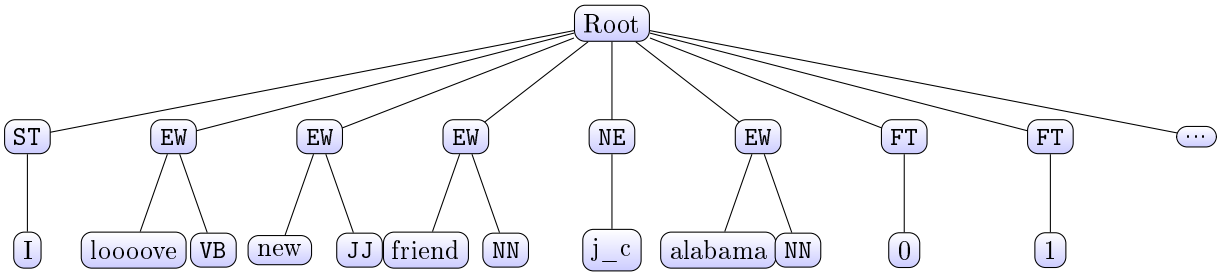


Figure 1: Example tree structure of a tweet. The original treet is 'I loooove new friend j_c alabama'.

3.2 Pre-processing

Punctuations, @user's and other meaningless words are removed. After regular pre-processing we label different words in the tweets with not only regular PoS but also @ to location, NE to non-english words, slang to non-english word but slangs, repeat to words with repeated characters like sadddddddd.

Example 1. *Tokens from a tweet* 'Things got a little festive at the office #christmas2016 @EllisIsland-Cafe' *will be labeled as follows:*
[['Things', 'NN'], ['got', 'VB'], ['a', 'ST'], ['little', 'JJ'], ['festive', 'NN'], ['at', 'ST'], ['the', 'ST'], ['office', 'NN'], ['#christmas2016', 'NE'], ['@EllisIslandCafe', '@'], [0, 'FT'], [0, 'FT'], [1, 'FT'], [1, 'FT'], [1, 'FT']].

FT's in the example means feature we detect if the text contains, in the same order, any slangs, repeated characters, @location's, positive emotion words and negative emotion words. We use online slang dictionary [6] to check if some word is a slang. We also detect repeat character words and use positive and negative words dictionary to check if a word is belongs to positive emotion or negative emotion. Then we create a list to contain those pre-processing infomation. The reason to do so is it is better to remain some non-english word or @location in some classification processing but some other processing need to delete some information, and we leave those information feasible to delete or keep for future processing.

3.3 Feature Extraction

After the pre-processing, we design a tree representation of tweets to combine many categories of features in one convenient representation. This idea comes from [2]. The tree has three levels. Firstly, we initialize the main tree to be root. In the second level, the tree contains four kinds of nodes, namely EW, NE, ST and FT. In these representations, EW stands for "English words", NE stands for "Non-English words", ST stands for "Stop words", and FT stands for "Feature". In the third level, EW nodes contains two children, while other kinds of nodes in second level contains one child. For the EW, the first child is the word itself, the other one is the part-of-speech of this word. For ST's and NE's, the child is the word itself. And for FT's, its child is the value of of the feature obtained in the pre-processing. Figure 1 shows an example of the tree structure.

Table 1 shows the features we designed at the beginning of the project[2]. Those features can be divided into 3 categories. The first category is the features whose type is integer. They can be extracted by performing linear search through all the trees. The second category is the features whose type is boolean, those features have been extracted in the pre-processing procedure. The third category contains Uppercase Ratio, Sentiment Score and Overlapping Ratio. The Uppercase Ratio is the number of capitalized text over all the characters of each tweets. The Sentiment Score is calculated by a python package called "Textblob"[3]. Also, we collected the descriptions of each emoji from their definition to see the overlapping ratio of tweets and those descriptions.

After we got all the value of the features, we found that the results are not as good as we thought. After discussed, we thought that those features work better in other works because their works are mainly binary classification problems so we decided to add some more features[4]. In the end we added three more groups of features based on TF, TF-IDF and χ^2.

Table 1: Features extracted from the tweets

Feature	**Type**	**Feature**	**Type**	**Feature**: Presence of	**Type**
# of `JJ`	`int`	# of `NE`	`int`	`@`'s	`bool`
# of `NN`	`int`	# of `negations`	`int`	Slangs	`bool`
# of `VB`	`int`	Uppercase ratio	`float`	Positive words	`bool`
# of `AB`	`int`	Sentiment score	`float`	Negative words	`bool`
# of stop words	`int`	Overlapping ratios	`floats`×20	Repeated characters	`bool`

Table 2: 20 most frequent emojis in English tweets in the United States

#	0	1	2	3	4	5	6	7	8	9
	❤️	😍	😂	💕	🔥	😊	😎	✨	💙	😘
#	10	11	12	13	14	15	16	17	18	19
	📷	🇺🇸	☀️	💜	😉	💯	😁	🎄	📸	😜

3.4 Baselines

We took two baselines[1] for comparison. (1) **Bag-of-words**: We represent each tweet with a vector of the most informative tokens selected using TF-IDF. (2) **Skip-gram vector average**: We represent each tweet with an average of vectors corresponding to tokens of the tweet. Each tweet m is represented with vector V_m:

$$V_m = \frac{\sum_{t \in T_m} S_t}{|T_m|}$$

where T_m are the set of tokens of the tweet m and S_t is the vector of token t in the skip-gram model.

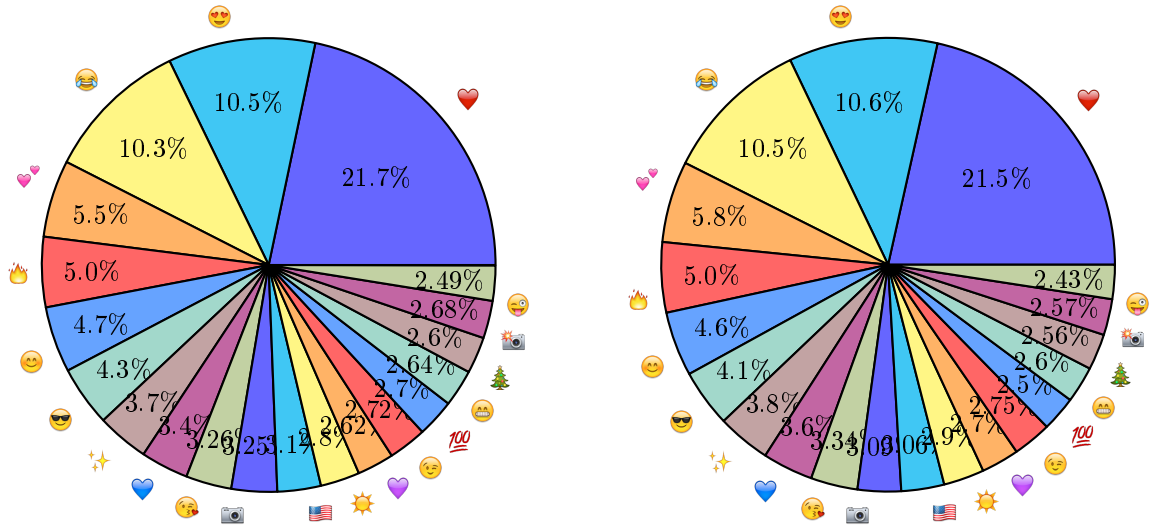
4 Experiments and evaluations

Experiments were conducted for studying the performances of methods and features that we intended to try out.

4.1 Datasets

The datasets we used for our experiments are provided by the CodaLab competition organizers. The training set consists of nearly 500,000 tweets in English which were posted during October 2015 to February 2017, geolocalized in the United States, and retrieved with Twitter APIs. Those tweets contain and only contain one emoji out of 20 most frequently used emojis. The trial set consists of 50,000 tweets with the same characters. Labels (corresponding emojis) of tweets in the training set are also provided as shown in Table 2.

There are some fact we want to point out about the datasets. (1) The classes are imbalanced in both training set and trail set, which is determined by their frequencies. (Figure 2) Although they are the 20 most frequent emojis, the first and the last one take very different portion in the datasets. (2) There are some groups of emojis are quite same (in shape), specifically, emojis 📷 and 📸, emojis 😘 and 😉, and emojis ❤️, 💙, 💜, and 💕. Similar shape means users might not be able to tell if there is significant difference among those groups of emojis, leading to confusion of using. (3) Label (*i.e.* emoji) of a certain tweet is stripped from the text content and stored seperately. No position information is kept for the emoji. Tweets with multiple emojis are treated as multiple tweets with one of these different emojis, so there might be repeated text entries in the training set (but with different labels). Yet there are no repeated entries in the trial set (which might be selections of the organizers), so the case of self-penalization by duplicated tweets in trial set on the scores can be ignored.



(a) (b)

Figure 2: Pie charts for classes in datasets. (a) shows percentages of emojis in training set (500k tweets), while (b) shows the trial set (50k tweets).

Also, a pretrained skip-gram model is used as one of our baselines mentioned above, which is provided by the competition organizers. The model is trained by '20M geolocalized tweets posted from October 2015 to Feburary 2017', and the dimension of the skip-gram vectors is 300[5].

4.2 Methods and Evaluations

The features we used are talked in previous parts of this report. We are going to use different combinations of these features to train different models, and compare the performances of them: tree-based: *random forest*, *extra tree* and a decision tree (*CART*), and boosting: *Adaboost* and *gradient boosting*.

Following the instructions of the competition organizers, we use *macro F1-score* to evaluate the performances of different features and methods. The reason using macro F1-score is that we would like a better overall performance, which 'would inherently mean a better sensitivity to the use of emojis in general, rather than for instance overfitting a model to do well in the three or four most common emojis of the test data' [2].

4.3 Results and analysis

In this subsection we will talk about results of the experiments.

Figure 3 shows the feature combinations receiving highest F1-scores on each method. As we can see, random forest and CART[3] outperform other methods. The detailed results of these two methods are shown in Table 4.

Out of all these methods and feature combinations, we find that using random forest with all features and tf received the best result. Table 5 and 6 shows the confusion matrix and scores for different emojis of this method and feature combination. Deviations of F1-scores of emojis from the macro score is not very much; that is, perfomance on different emojis are balanced. Also, for similar emoji groups we mentioned in the previous subsection, the results indicate that there are some kind of differences of people using these similar emojis, and the classifier was able to capture the differences.

There are some problems indicated by the two tables as well. Firstly, the classifier's performance on the 2 most frequent emojis (❤ and 😍) are relatively bad comparing to the rest. The original experiment of compitition organizer[1] also had this issue. It is our speculation that users use these two emojis in

[2] by specifications of the organizers.

[3] Although the highest score of extra tree is close to CART, scores for other feature combinations are much lower

Table 3: Highest F1-scores for different methods

Methods	**Highest F1-score**	**Corresponding featres**
random forest	**0.5148762**	+tf
CART	**0.450688**	+tfidf
extra tree	0.4452189	+chi
gradient boosting	0.1821114	+chi(WordNet)
Adaboost	0.1605175	+descriptions

Table 4: F1-scores for random forest and CART with different feature combinations

Feature combinations	**Random forest**	**CART**
bag of words	0.35034	0.34952
skip-gram	0.51137	0.44203
skip-gram (uppercase sensitive)	0.50655	0.44178
features only	0.38154	0.36380
skip-gram + features without @	0.50996	0.44317
skip-gram + features include @	0.51169	0.44404
skip-gram + features + descriptions	0.51379	0.44890
skip-gram + features + tf-idf	0.51245	**0.45069**
skip-gram + features + tf	**0.51488**	0.45054
skip-gram + features + chi	0.51162	0.44399
skip-gram + features + chi(WordNet)	0.51007	0.44338
BLSTM by the organizers		0.2725

various occasions (more than 30 percents of all tweets in our datasets), or the two emojis are semantically generalized and do not carry specific meanings in many cases. As a comparison, the emoji 🎄 received the highest F1-score over all 20 emojis, probably because that the emoji would only be used in Christmas related tweets. Another problem is that for all emojis, the most mis-predicted answer is the emomji ❤️, the most frequent one. The reason might also be that people would use this emoji in so many cases that there is no specific (or fixed) meaning in it when people are using it.

5 Conclusion and future work

The results of our experiments show that feature-based methods for emoji prediction is a feasible way to tackle the problem. Random forest as classifier is both easy to train and giving good results. The results also show that the performance of random forest is even and good for 20 most frequent emojis. Problems indicated by the results is that some most frequent emojis are tend to be mis-predicted for their ambiuous usages of Twitter users, while some less frequent emojis are more precisely predicted for their certain meaning and usages.

After the evaluation, we also read some papers after we got the results. However, because the time is limited, we would like to implement them in the future if possible. We want to improve our project in two ways. One is use deep-learning method like BLSTM[1], the other is to find more features. For example, we find pattern feature in another paper[4], which divide words into three categories according to their frequency, namely High Frequency Words, Content Words and Regular Words. With the different permutation and combination, we can get different patterns. And we can use those patterns as features. Other features like punctuation features can also be used in our future work.

Table 5: Normalized confusion matrix for random forest with *'skip-gram + features + tf'*

Actual \ Predicted	❤️	😍	😂	💕	🔥	😊	😎	✨	💙	😘	📷	🇺🇸	☀️	💜	😉	💯	😁	🎄	📸	😜
❤️	.48	.07	.01	.02	.04	.03	.01	.01	.02	.01	.01	.02	.04	.05	.02	.03	.02	.03	.03	.03
😍	.22	.35	.02	.02	.03	.02	.02	.01	.02	.01	.01	.02	.05	.04	.03	.03	.03	.03	.02	.02
😂	.13	.04	.50	.01	.02	.01	.01	.01	.01	.01	.04	.01	.03	.01	.04	.02	.02	.03	.01	.02
💕	.13	.04	.01	.56	.02	.01	.01	.01	.01	.01	.02	.01	.03	.02	.01	.02	.03	.02	.02	.01
🔥	.2	.03	.01	.01	.44	.02	.01	.01	.02	.01	.02	.01	.02	.03	.03	.02	.05	.01	.02	.02
😊	.17	.04	.01	.01	.02	.54	.01	.01	.01	.01	.01	.01	.02	.04	.02	.02	.01	.02	.03	.01
😎	.13	.03	.00	.01	.02	.01	.56	.01	.01	.01	.01	.02	.06	.02	.02	.03	.02	.02	.01	.01
✨	.10	.04	.01	.01	.01	.01	.01	.56	.01	.01	.01	.01	.06	.02	.03	.02	.02	.03	.01	.01
💙	.10	.04	.01	.01	.01	.01	.01	.02	.59	.01	.01	.01	.05	.02	.02	.03	.03	.02	.01	.01
😘	.11	.03	.00	.01	.01	.00	.00	.01	.01	.71	.00	.00	.02	.02	.01	.02	.01	.01	.01	.01
📷	.11	.03	.04	.01	.01	.01	.01	.01	.01	.01	.58	.01	.03	.02	.03	.02	.03	.02	.01	.01
🇺🇸	.09	.03	.02	.01	.01	.00	.01	.01	.01	.00	.01	.63	.04	.02	.02	.02	.02	.01	.01	.01
☀️	.13	.05	.01	.01	.01	.01	.02	.03	.02	.01	.02	.02	.50	.02	.03	.03	.02	.02	.01	.02
💜	.24	.05	.01	.01	.02	.02	.01	.01	.01	.01	.00	.01	.02	.46	.01	.03	.02	.02	.03	.02
😉	.09	.03	.02	.01	.02	.01	.01	.02	.01	.01	.02	.01	.04	.01	.62	.02	.02	.02	.01	.01
💯	.13	.04	.01	.01	.03	.01	.01	.01	.02	.01	.01	.01	.03	.02	.01	.58	.02	.01	.01	.01
😁	.11	.04	.01	.02	.03	.01	.01	.01	.01	.00	.01	.01	.03	.02	.02	.02	.60	.01	.01	.01
🎄	.10	.03	.01	.01	.02	.01	.01	.01	.01	.01	.02	.00	.02	.02	.01	.01	.01	.66	.01	.01
📸	.12	.04	.00	.01	.02	.01	.00	.01	.01	.00	.00	.00	.01	.03	.01	.01	.01	.01	.67	.01
😜	.14	.03	.01	.00	.02	.01	.01	.00	.00	.00	.00	.01	.01	.03	.01	.01	.01	.01	.01	.66

Table 6: F1-scores for random forest and CART with different feature combinations

	Precision	**Recall**	**F1-score**	**Support**
❤️	0.42	0.48	*0.45*	9295
😍	0.64	0.35	*0.45*	9719
😂	0.69	0.50	0.58	7238
💕	0.49	0.46	0.48	3050
🔥	0.58	0.62	0.60	2327
😊	0.46	0.58	0.51	1814
😎	0.47	0.60	0.52	1605
✨	0.47	0.66	0.55	1328
💙	0.46	0.67	0.55	1228
😘	0.44	0.66	0.53	1105
📷	0.55	0.50	0.52	1674
🇺🇸	0.58	0.56	0.57	1585
☀️	0.11	0.44	0.17	357
💜	0.44	0.54	0.48	1095
😉	0.49	0.56	0.52	1195
💯	0.50	0.56	0.53	1112
😁	0.47	0.59	0.52	1038
🎄	0.68	0.71	**0.70**	1228
📸	0.51	0.58	0.54	1134
😜	0.46	0.63	0.53	873
avg	0.54	0.51	0.51	50000

References

[1] Barbieri, F., Ballesteros, M., & Saggion, H. (2017). Are Emojis Predictable?. *arXiv preprint arXiv:1702.07285*.

[2] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011, June). Sentiment analysis of twitter data. In *the workshop on languages in social media* (pp. 30-38). Association for Computational Linguistics.

[3] TextBlob: Simplified Text Processing. Retrieved November 20, 2017, from http://textblob.readthedocs.io/en/dev/

[4] Kanavos, A., Nodarakis, N., Sioutas, S., Tsakalidis, A., Tsolis, D., & Tzimas, G. (2017). Large Scale Implementations for Twitter Sentiment Classification. *Algorithms*, 10(1), 33.

[5] F. Barbieri. How Cosmopolitan Are Emojis?. Retrieved December 11, 2017, from https://github.com/fvancesco/acmmm2016

[6] Walter Rader, Online slang dictionary, Retrieved november 20, 2017, from http://onlineslangdictionary.com/

[7] Aoki, S., & Uchida, O. (2011, March). A method for automatically generating the emotional vectors of emoticons using weblog articles. In Proc. *10th WSEAS Int. Conf. on Applied Computer and Applied Computational Science*, Stevens Point, Wisconsin, USA (pp. 132-136).

[8] Espinosa-Anke, L., Saggion, H., & Barbieri, F. (2016). Revealing patterns of Twitter emoji usage in Barcelona and Madrid. *Frontiers in Artificial Intelligence and Applications*. 2016;(Artificial Intelligence Research and Development) 288: 239-44.

[9] Barbieri, F., Kruszewski, G., Ronzano, F., & Saggion, H. (2016, October). How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In Proceedings of *the 2016 ACM on Multimedia Conference* (pp. 531-535). ACM.

[10] Barbieri, F., Ronzano, F., & Saggion, H. (2016). What does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis. *LREC*.

[11] Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

[12] Ljubešić, N., & Fišer, D. (2016). A Global Analysis of Emoji Usage. In Proceedings of *the 10th Web as Corpus Workshop* (pp. 82-89).

[13] Tian, Y., Galery, T., Dulcinati, G., Molimpakis, E., & Sun, C. (2017). Facebook Sentiment: Reactions and Emojis. *SocialNLP* 2017, 11.

[14] Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., & Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

[15] Novak, P. K., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PloS one*, 10(12), e0144296.

Note

Jiarong Yu contributed to Sections 1 and 2 of this report.
Shuai Hao contributed to Sections 3.2, 3.3, 5, and abstract of this report.
Shou Liu contributed to Sections 2, 3.1, 3.4, 4, 5 and all pictures and tables of this report. He also typeset this report in LaTeX.