DUTH at SemEval-2018 Task 2: Emoji Prediction in Tweets

Dimitrios Effrosynidis **Georgios Peikos** **Symeon Symeonidis** **Avi Arampatzis**

Database & Information Retrieval research unit,
Department of Electrical & Computer Engineering,
Democritus University of Thrace, Xanthi 67100, Greece
{deffrosy,georpeik,ssymeoni,avi}@ee.duth.gr

Abstract

This paper describes the approach that was developed for SemEval 2018 Task 2 (Multilingual Emoji Prediction) by the DUTH Team. First, we employed a combination of preprocessing techniques to reduce the noise of tweets and produce a number of features. Then, we built several N-grams, to represent the combination of word and emojis. Finally, we trained our system with a tuned LinearSVC classifier. Our approach in the leaderboard ranked 18th amongst 48 teams.

1 Introduction

Emojis are used in everyday life to express words or feelings of microblogging users. They are commonly placed at the end of a sentence or alone. In this paper, we show how our emoji prediction framework was applied to SemEval-2018 Task 2 (Multilingual Emoji Prediction) (Barbieri et al., 2018), specifically on Subtask 1 (Emoji Prediction in English).

In the last few years, many studies concentrated on emoji prediction and analysis. The prediction of emojis, the connection of emojis and words, and their separation from content-based tweet messages, based on Long ShortTerm Memory networks (LSTMs), was examined by Barbieri et al. (2017). The combination of emojis and sentiment was investigated by Novak et al. (2015), who developed the first emoji sentiment lexicon and created a sentiment map of the 751 most frequently used emojis. The study of Barbieri et al. (2016) tested several skip-gram word embedding models to measure the difference in performance between machine-learning models and human annotation. Na'aman et al. (2017) analyzed the viability of a trained classifier to differentiate between those emojis utilized as semantic substance words and those utilized as paralinguistic or emotional multimodal markers. Miller et al. (2017) investigated the hypothesis of previous works that emojis in their regular textual contexts would generously reduce and lead to miscommunication, but they found no such evidence; the potential for miscommunication appeared to be the same.

The rest of this paper is organized as follows. Section 2 describes the architecture of our system and the dataset. In Section 3, we discuss the various parameters that were used to fine-tune the system, and present the performance of our framework. In Section 4, we lay out our main conclusions and research issues for further investigation.

2 System Description

The principal goal of SemEval-2018 Task 2 - Subtask 1 was emoji prediction in English. The framework we utilized consists of a bag of-words representation and N-gram extraction. We used the popular machine learning tool for Python, called Scikit-Learn (Pedregosa et al., 2011).

2.1 Preprocessing

For the preprocessing of tweets, we were guided by the results of our previous research (Effrosynidis et al., 2017). We used the effective combination of the following techniques:

- Replace URLs and User Mentions to the tags 'URL' and 'AT USER', as the majority of tweets on Twitter contain a URL and mentions which are considered noise.
- Replace Contractions, as it reduces the dimensionality of the problem and improves speed and accuracy according to the abovementioned paper.
- Remove Numbers, because they do not contain any sentiment.

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 466–469
New Orleans, Louisiana, June 5–6, 2018.

Label	Sentences	Unique Words	Words/Sentence	Tweets	User tags	Hashtags	Unique Hashtags
0	160220	110305	7.191	107669	20471	110655	55287
1	75706	70430	7.351	51923	11097	57839	32922
2	75879	69280	7.881	50988	13884	45182	28984
3	37068	36928	8.113	27517	4284	17984	12128
4	35327	45940	7.693	24625	10164	28940	19140
5	35347	37710	7.598	23318	5204	20811	13802
6	30406	38420	7.418	21323	4097	24013	15842
7	24859	31386	8.196	18388	3316	15783	10215
8	23623	28479	7.585	17024	2508	14594	10157
9	24983	28742	7.125	16167	4405	13862	9496
10	25705	35760	6.895	16045	11186	19929	12237
11	22136	25142	6.930	15365	2012	24937	10192
12	18482	20883	7.441	13882	1282	12848	7478
13	18538	24496	7.524	12981	2382	11317	8158
14	21060	28277	7.900	13472	3023	13113	9831
15	18353	26909	7.835	13408	3210	12886	9206
16	20531	27494	7.360	13094	3084	13044	9582
17	18631	20292	7.032	12853	1477	11607	6338
18	20164	29989	7.116	13255	8918	15628	10065
19	18900	27230	7.622	12307	2421	12653	9359
SUM	725918	764092	7.490	495604	118425	497625	300419

Table 1: Statistics per emoji.

- Replace Repetitions of Punctuation, which merges in the same feature the intensity of emotions. For example, if we find more than two consecutive exclamation, question or stop marks, we replace them with a single one.

2.2 Dataset

The training and testing datasets were provided by the organizers. The training set contained approximately $500,000$ tweets, where each tweet contained a single emoji, before they removed it and set it as class label. That emoji is used as the class label for the particular tweet.

We extracted various statistics for the dataset as it can be seen in Table 1. Some class labels contain more sentences per tweet, like label 10 (📷) and 0 (❤). We also observe that the emoji 💕 has on average much fewer hashtags per tweet, while the emoji 🇺🇸 has much more. All the other emojis range within reasonable limits. The emojis with labels 7 (✨) and 3 (💕) are expressed using more words on average, while the emojis 10 (📷) and 11 (🇺🇸) are expressed with fewer words.

All the above observations are important to understand the dataset and how people are using each emoji. One can use these statistics in order to create more features and test them to see the changes in classification accuracy. For example, one can count the words of each new sentence for classification, and compare them with the ones derived from the training dataset.

In our study, we compared several machine learning algorithms (Ridge, Logistic Regression, Passive-Aggressive, and Linear SVC), and three different word to vector representations (tf-idf Vectorizer, count Vectorizer, and hashing Vectorizer). The macro F-measure score was computed for 10-folds cross-validation on the training set and on the trial set while using the training set for training. We employed word n-grams and character n-grams (n ranging from 1 to 4), with the latter ones performing poorly.

3 Experimental Results

In this section, we describe the different classifiers and vectorizers used and present our results.

	Macro-averaged F-measure			
Vectorizer	**Ridge**	**Logistic Regression**	**Passive-Aggressive**	**LinearSVC**
tfidfVectorizer	26.2	25.1	24.0	26.6
countVectorizer	25.9	26.7	20.9	25.5
hashingVectorizer	24.6	23.0	23.6	25.9

Table 2: Results per classifier and vectorizer using 10-fold unigrams.

3.1 Classifiers

In order to gain a better perspective on the problem, we trained four different classification algorithms. We test each classifier comparing their macro F-measure score. We choose LinearSVC, because of the stability we noticed in the results it returned. Below we discuss every classifier:

- Ridge: an algorithm belonging to the Generalized Linear Models family. Text classification problems tend to be quite high dimensional, and high dimensional problems are likely to be linearly separable; this is one reason why Ridge performs quite well.
- Logistic Regression: despite its name, it is used for classification and fits a linear model as well. In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme.
- Passive-Aggressive: belongs to a family of algorithms for large-scale learning, which does not require a learning rate and includes a regularization parameter C. On the one hand, the aggressive mode of the algorithm means that if an incorrect classification occurs, the model updates to adjust to this misclassified example. On the other hand, the model stays unchanged in every correct classification and this is the passive behavior of the algorithm (Crammer et al., 2006).
- Linear SVC: the purpose of this algorithm is to fit the data by finding a set of hyperplanes that separate space into areas representing classes. The most efficient way is considered to be the max distance between data points and the hyperplane.

3.2 Vectorizers

Nowadays, one can find many vectorizers to use in order to extract features. We used the following three vectorizers provided by Python's Scikit-Learn library (Pedregosa et al., 2011), in order to transform tweets into vectors of features.

- tf-idf Vectorizer: a vectorizer which scales the term frequency counts in each tweet by penalising terms that appear more frequently across the dataset.
- count Vectorizer: converts the collection of tweets to a matrix of token counts.
- hashing Vectorizer: a vectorizer which applies a hashing function to term frequency counts in each document. This vectorizer leads to a sparse matrix holding token occurrence counts (or binary occurrence information).

Each vectorizer we used is efficient under certain circumstances. In addition, we noticed that the combination of the vectorizer and classification algorithm is crucial for our problem. In our work, as we can see in Table 2, the combination of countVectorizer and Logistic Regression leads to the best result. However, the tfidfVectorizer achieves greater results than the countVectorizer and the hashingVectorizer in the majority of the algorithms; for this reason, we proceeded with the tfidfVectorizer.

3.3 Evaluation Results

We evaluate the performance of our system with the macro F-measure score. The macro F-measure score gives equal weight to each emoji category, regardless of its class size. The F-measure per emoji class is the harmonic mean of the precision and recall of the class:

$$\text{F-measure} = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} .$$

The macro-average F-measure score is obtained by taking the average of F-measure values across emoji classes:

$$\text{macro F-measure} = \frac{1}{M} \sum_{n=1}^{M} \text{F-score(n)} ,$$

where M is the total number of classes.

In Table 3 we present the macro F-measure score of tfidfVectorizer combined with LinearSVC classification algorithm. In the first column, the results of 10-folds cross validation on the training set are presented. In the second column we present the results when training with the training data and testing with the trial data. As it can be seen, four-grams performance on trial data has the highest value, but trigrams perform better on 10-folds cross-validation. This is the reason we used trigrams to train our model for the submitted runs.

	10-folds Training	**Trial**
word-unigrams	26.6	46.8
word-bigrams	29.3	61.4
word-trigrams	29.4	63.7
word-fourgrams	29.2	64.4

Table 3: F-measure results for word N-grams.

4 Conclusions

In this paper, we presented the framework we used to participate in the SemEval-2018 emoji prediction competition. We used a tfidfVectorizer combined with a LinearSVC classification algorithm, employing word tri-grams, to train our model. Our team ranked in the 18th place among 48 teams.

For future work, it would be interesting to test Neural Network approaches, to use emoji sentiment lexica (Novak et al., 2015), or additionally include more features. Furthermore, it would likewise be intriguing to investigate the miscommunication of emojis in their natural textual contexts.

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 105–111. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016. What does this emoji mean? A vector space skip-gram model for twitter emojis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Dimitrios Effrosynidis, Symeon Symeonidis, and Avi Arampatzis. 2017. A comparison of pre-processing techniques for twitter sentiment analysis. In *Research and Advanced Technology for Digital Libraries - 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings*, volume 10450 of *Lecture Notes in Computer Science*, pages 394–406. Springer.

Hannah Jean Miller, Daniel Kluver, Jacob Thebault-Spieker, Loren G. Terveen, and Brent J. Hecht. 2017. Understanding emoji ambiguity in context: The role of text in emoji-related miscommunication. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 152–161. AAAI Press.

Noa Na'aman, Hannah Provenza, and Orion Montoya. 2017. Varying linguistic purposes of emoji in (twitter) context. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Student Research Workshop*, pages 136–141. Association for Computational Linguistics.

Petra Kralj Novak, Jasmina Smailovic, Borut Sluban, and Igor Mozetic. 2015. Sentiment of emojis. *CoRR*, abs/1509.07761.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.