Tweety at SemEval-2018 Task 2: Predicting Emojis using Hierarchical Attention Neural Networks and Support Vector Machine

Daniel Kopev, Atanas Atanasov, Dimitrina Zlatkova,
Momchil Hardalov, Ivan Koychev
FMI, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
{dkopev, amitkov, dvzlatkova}@uni-sofia.bg
{hardalov, koychev}@fmi.uni-sofia.bg

Ivelina Nikolova, Galia Angelova
IICT, Bulgarian Academy of Sciences, Sofia, Bulgaria
{iva, galia}@lml.bas.bg

Abstract

We present the system built for SemEval-2018 Task 2 on Emoji Prediction. Although Twitter messages are very short we managed to design a wide variety of features: textual, semantic, sentiment, emotion-, and color-related ones. We investigated different methods of text preprocessing including replacing text emojis with respective tokens and splitting hashtags to capture more meaning. To represent text we used word n-grams and word embeddings. We experimented with a wide range of classifiers and our best results were achieved using a SVM-based classifier and a Hierarchical Attention Neural Network.

1 Introduction

SemEval 2018 Task 2 on Emoji Prediction (Barbieri et al., 2018) is a classical task for supervised learning. Given labeled data consisting of Twitter messages and a corresponding emoji as a label, the aims is to classify new examples (tweets) into 20 categories - the most frequent emojis of two languages: English (Subtask 1) and Spanish (Subtask 2). We participated only in **Subtask 1**. The labels are presented in Figure 1:

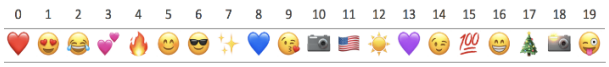

Figure 1: Labels ordered by frequency.

2 Related Work

Prior work includes using LSTM-RNN and CNN models (Zhao and Zeng) utilizing pre-trained Twitter embeddings with the latter achieving very good results. Other works (Barbieri et al., 2017) show that LSTMs have high accuracy and even outperform humans at the emoji prediction task. In (Barbieri et al., 2016) the skip-gram neural embedding model is applied with different dimensions of the vectors and length of the windows applied to both words and emojis.

3 Data

We used the **500k training** and **50k trial** tweets provided by the organizers to train and validate our models respectively. One key mistake we made is that we did not compare those two datasets for duplicate entries. As we found out only after the submission deadline, the train and trial data had a 40% overlap, which unfortunately skewed our expected results and made them unrealistically high. The experimental results presented in Table 2 are on the data with removed duplicates.

We crawled additional 100k tweets via *Tweepy*[1] only 5k of which were compliant with the requirements to contain exactly one emoji. With this external data we aimed to improve the overall performance of our models, but since it was way too small, it did not have much effect.

Finally, when predicting on the test data, we trained our models on the combined train, trial and crawled data.

Looking at the emojis we immediately noticed two problematic groups: *1.* two emojis with a **camera** - one with flash and one without; *2.* four emojis containing a **heart** - three of them exactly the same, different only in color (red, blue, purple), and one with two pink hearts. We approach the second group with color-related features (see Section 4.2)

4 Method

4.1 Data Preprocessing

Replacing Text Emojis: Text emojis like **:)**, **:D**, **:o** and others should in theory carry valuable

[1] http://www.tweepy.org/

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 497–501
New Orleans, Louisiana, June 5–6, 2018.

information, thus we encode them to unique strings that will not be removed in future preprocessing steps. The encoded strings are: __smile__ __laughing__ __very_happy__ __sad__ __cry__ and __surprise__.

Removing Punctuation and Artifacts: The data given by the orginizers comes with user mentions replaced by `@user` and all URLs removed. We remove `@user`, because the user mentions are taken into account in the feature engineering step, even though their position is lost. We also remove automatic location mentions in the form `@ Location`. Non-letter characters are also removed, exception is #, used to identify words in hashtags which we later attempt to split.

Hashtag Splitting: We try to break down each token starting with # to a set of words. The process iterates over the token until a word existing in a corpus is found. Then we take the rest of the token and recursively apply the same procedure until the whole original token is empty. The longest matching word is always taken first. For subtoken word identification we used the *Brown corpus*. As anticipated, adding a slang corpus seemed to worsen the splits. For simplicity we take the first found valid split, but an improvement would be to calculate and take the most probable one.

Tokenization and Lemmatization: A *WordNet* (Miller, 1995) lemmatizer is used on tokenized (*TweetTokenizer* from *NLTK*[2]) and lowercased beforehand part-of-speech annotated tweets. High frequency words are removed.

4.2 Features

Textual Features: Since all we had was the text of the tweet without any metadata or context, we focused on extracting valuable information from the text itself. We gathered statistics like number of words, hashtags, stop-words, user mentions, mean word length and more. Some of those were specifically targeted at predicting certain emojis. For instance, we hoped counting the digits and percentage signs would help identifying 💯. Punctuation such as question marks, exclamation marks or words with all title letters could signify an intensified face emotion like 😂 or 😍.

Semantic features: Looking at the train data, we noticed that 42% of the tweets end in the following pattern: `@ LocationName`, for instance *Happy birthday Nathan!!! @ Boca Gardens*. We figured that this was an automatically assigned location and extracted it as a separate feature.

Cluster Id	Words
00101111010	almost nearly practically alm0st nearlly almst
111010100010	lmao lmfao lmaoo lmaooo lool rofl loool lmfaoo
111010100011	haha hahaha hehe hahahaha hahah aha hehehe ahaha

Table 1: Twitter clusters.

Emotion-related features: To capture emotion, we used the *NRC Word-Emotion Association Lexicon* (Mohammad and Turney, 2013). It contains a list of English words and their associations with eight basic emotions - anger, fear, anticipation, trust, surprise, sadness, joy, and disgust.

Color-related features: Dealing with four emojis with the heart symbol in different colors, we decided to use another NRC Lexicon - on Word-Colour Associations (Mohammad, 2011). It consists of mappings for eleven colors - white, black, red, green, yellow, blue, brown, pink, purple, orange and grey, which covers the four heart colors in question.

Sentiment features: In order to capture sentiment in the tweets, we used *SentiWordNet* (Baccianella et al., 2010) to associate each token in the tweet with a positive and negative score.

Twitter clusters: Another observation we made while looking at the tweets is that there were a lot of misspelled words and words with identical meaning written with different syntax (mainly slang). To handle that we utilized *Hierarchical Twitter Word Clusters*[3]. The clusters also help identify synonymous words. Three exemplary clusters of words are shown in Table 1.

All features were used in all classification experiments, except in some of the stacking, where a subset was used.

4.3 Classifiers

Using the features above, we had represented each tweet into that vector-space. Experiments were made with classifiers from various types: Linear, Non-Linear, and Deep Learning.

Linear Classifiers: For our baseline we used Multinomial Naive Bayes, which we managed to outperform with ease. In the subsequent experiments we used linear classifiers - Logistic Regression with L-BFGS optimizer (Liu and Nocedal,

[2]http://www.nltk.org/

[3]http://www.cs.cmu.edu/ ark/TweetNLP

1989) and Linear SVMs with SGD optimizer (Bottou, 2010).

Non-Linear Classifiers: As we wanted to overcome the linearity of the LR and SVMs we had moved to non-linear classifiers. We had fed our feature vectors into Random Forest with 300 estimators, and AdaBoost with Decision Tree base, again with the same number of estimators.

Stacking: Another idea was to combine count-based and semantic features. For this we applied two versions of Stacking ensembles. The first includes SVM (tf-idf), AdaBoost (embeddings) and Random Forest (semantic and sentiment extracted features). The second one is composed of SVM (tf-idf), AdaBoost (embeddings) and Multi-layer Perceptrion (tf-idf). Both ensembles use hard weighted voting with coefficients 1.5 for the SVM prediction and 1.0 for the rest.

Deep Learning: We applied some of the state-of-the-art neural architectures for text-processing. Our experiments included Multi-layer Perceptrions, Recurrent NNs with LSTM (Hochreiter and Schmidhuber, 1997) and Convolutional NNs.

In the dev phase we achieved best results using *Hierarchical Attention Neural Network* (Yang et al., 2016) (HANN). The idea of HANN is to mimic the hierarchical structure of documents. It has two levels of attention mechanism: for word and for sentence. This enables them to capture and act differently on different levels of content importance. HANNs structure is build up from: word sequence encoder, word-level attention layer, sentence encoder and a sentence-level attention layer. Word Encoder gets word annotations from an embedding matrix summarizing information from both directions of the words. Word Attention (an attention mechanism), extracts the most important words, because not all words contribute equally to the sentence's meaning. Sentence Attention (another attention mechanism), is used to mark the important sentences at sentence level context.

As another experiment we used a two-layered bidirectional LSTM with a dropout rate of 0.35 and the Adam optimizer.

Another interesting approach that we adapted was to apply Convolutional Layer for text (Kim, 2014) that allows our network to learn and capture patterns for adjacent words in sentences. CNN are widely applied for image data, by using them for text classification we can learn and track correlations between close words and inputs. An advantage of CNN over RNN is that CNN are much faster than RNN architectures. CNNs allow our network to see the entire input at once and to parallelize all operations, because a convolutional kernel acts on each patch independently.

The key insight of boosting our Neural Network models was switching from ReLU to ELU as activation function. Proper Dropout Strategy (between 0.35 and 0.4) also improved our validation score.

5 Experiments and Evaluation

5.1 Experimental Setup

We transformed the training tweets into vectors using two mainstream techniques: tf-idf representation and word embeddings. While building the tf-idf weights we formed word 6-grams (without the stop words) and removed entries with DF greater than 0.5. The second approach consisted of using 200-dimensional GloVe embeddings (Pennington et al., 2014) trained on Twitter corpus with 27 billion tokens. Using the embedding of each term we concatenated the component-wise minimum and maximum vectors (De Boom et al., 2016). Some classifiers were tested using both representations when we found that appropriate.

5.2 Results

The results from those experiments on 10k train (sampled from the train dataset) and 1k test (sampled from the trial dataset) data are presented in Table 2. The experimental results are on the data with duplicates removed. The second stacking gave a better result than SVM, but we did not manage to run the model on the whole dataset in time for the submission. We placed 25th in the official ranking.

Precision, recall and Macro-F1 per class (on duplicated data) can be seen in Table 3.

The confusion matrix in Figure 2 reveals that two of the most confused classes are the ones with a camera, which was expected. Less anticipated is the strong confusion between the heart and the sun emojis. Overall, the heart emoji is confused the most with the rest of the classes, but since it's the most common one it's possible that classifiers often falsely predict it.

In terms of features, we found out that the n-gram representation of the tweets was the most important in terms of determining its label and the additional features did not have much influence.

Model	Precision	Recall	Macro-F1
Multinomial Naive Bayes	0.05	0.21	1.763
Logistic Regression with L-BFGS	0.22	0.28	13.16
Multi-Layer Perceptron 2-hidden (ReLU)	0.26	0.26	17.898
Random Forest (300 estimators)	0.20	0.26	16.167
AdaBoost with Decision Tree base (300 estimators)	0.15	0.19	7.825
SVM with tf-idf	0.23	0.27	19.554
SVM with Twitter embeddings	0.16	0.18	8.522
Stacking (SVM + AdaBoost + Random Forest)	0.25	0.24	13.764
Stacking (SVM + AdaBoost + MLP)	**0.25**	**0.28**	**20.106**
Convolutional Neural Network	0.15	0.14	12.034
Recurrent Neural Network with LSTM	0.24	0.17	13.106
HANN	0.30	0.13	15.999
SVM tf-idf	**0.30**	**0.33**	**23.3**
HANN	0.31	0.33	22.518

Table 2: Precision, Recall and F-measure of experimental classifiers on 1k tweets (top) and final classifiers on 50k tweets (bottom).

Emoji	Precision	Recall	Macro-F1	%
❤️	36.53	54.25	43.66	21.6
😍	22.49	29.25	25.43	9.66
😂	35.72	46.52	40.41	9.07
💕	14.78	6.41	8.94	5.21
🔥	46.01	48.12	47.04	7.43
😊	8.94	5.52	6.82	3.23
😎	23.07	11.97	15.77	3.99
✨	35.62	18.52	24.37	5.5
💙	20.42	10.65	14.0	3.1
😘	11.87	4.43	6.45	2.35
📷	19.26	20.11	19.68	2.86
🇺🇸	52.66	60.39	56.26	3.9
☀️	31.33	45.77	37.2	2.53
💜	21.28	5.39	8.6	2.23
😉	8.03	2.91	4.27	2.61
💯	17.62	21.7	19.45	2.49
😁	13.02	4.34	6.51	2.31
🎄	48.58	78.64	60.06	3.09
📸	30.14	11.83	16.99	4.83
😜	7.93	2.77	4.11	2.02

Table 3: Precision, Recall, F-measure and percentage of occurrences in the test set of each emoji.

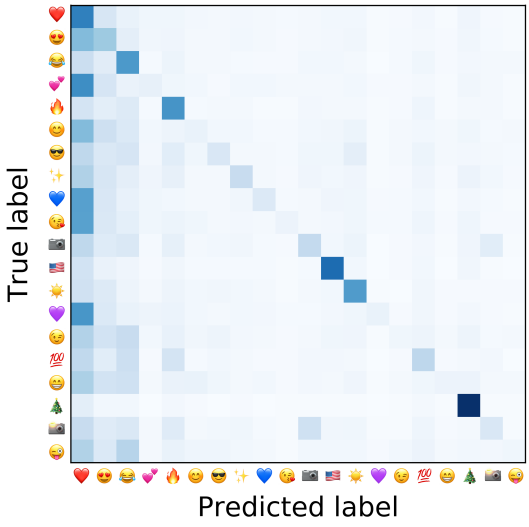

Figure 2: Confusion matrix per emoji type.

6 Conclusion

The work we did on the Emoji Prediction task seems promising, even though we could make our process better by filtering train data, retrieving more tweets and focusing more on the preprocessing of the tweets. There's a lot of room for improvement, given that the task is very challenging - tweets are short and full of slang words and ambiguous emoticons. We tried to combat those through some feature engineering, preprocessing and semantic approach for vectorization.

Improvements could be made with the semantic representation of the tweets. Because our embedding representations use coordinate-wise minimization and miximization, a lot of meaning is lost. Embedding approaches that work on a higher than word level text blocks like Skip-Thoughts vectors (Kiros et al., 2015) could decrease this loss. As future work we plan on using more sophisticated architectures like deeper CNNs and Squeeze-and-Excitation Networks for text.

Acknowledgments

This research was done by MSc students in Computer Science at the Sofia University "St Kliment Ohridski".

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

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, Valencia, Spain. 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 *Language Resources and Evaluation conference, LREC*, Portoroz, Slovenia.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recogn. Lett.*, 80(C):150–156.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3294–3302, Cambridge, MA, USA. MIT Press.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saif Mohammad. 2011. Colourful language: Measuring word-colour associations. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 97–106, Portland, Oregon, USA. Association for Computational Linguistics.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

Luda Zhao and Connie Zeng. Using neural networks to predict emoji usage from twitter data.