Peperomia at SemEval-2018 Task 2: Vector Similarity Based Approach for Emoji Prediction

Jing Chen , **Dechuan Yang** , **Xilian Li** , **Wei Chen** [*] and **Tengjiao Wang**

Key Lab of High Confidence Software Technologies (MOE), School of EECS,
Peking University, Beijing, 100871, China
{chenjing.amy, yangdechuan, xilianli, pekingchenwei, tjwang}@pku.edu.cn

Abstract

This paper describes our participation in SemEval 2018 Task 2: Multilingual Emoji Prediction, in which participants are asked to predict a tweet's most associated emoji from 20 emojis. Instead of regarding it as a 20-class classification problem we regard it as a text similarity problem. We propose a vector similarity based approach for this task. First the distributed representation (tweet vector) for each tweet is generated, then the similarity between this tweet vector and each emoji's embedding is evaluated. The most similar emoji is chosen as the predicted label. Experimental results show that our approach performs comparably with the classification approach and shows its advantage in classifying emojis with similar semantic meaning.

1 Introduction

Participants for SemEval 2018 Task 2 (Barbieri et al.) are asked to predict the most likely associated emoji given the tweet. For simplicity purposes, each tweet contains one and only one emoji, which belongs to the 20 most frequent emojis. We participate in its subtask 1: Emoji Prediction in English.

With the wide-spread use on many social platforms, emoji has attracted more and more attention of researchers recently. Miller et al. (2016) explored whether emoji renderings or differences across platforms gave rise to diverse interpretations of emoji. For the same emoji, the sender and the receiver may have different interpretations of its meaning. This misinterpretation occurs when *joint perceptual experience* of sender and receiver lacks or the platforms' rendering style differs. Some efforts have been devoted to studying emoji through its distributed representation. Barbieri et al. (2016a,b) trained emoji embeddings with a skip-gram model through millions of tweets, and explored the similarity and relatedness among these embeddings in various languages. Their results suggested that the overall semantic of emoji was preserved across languages, but some emojis were interpretated differently due to users' socio-geographical differences. Eisner et al. (2016) trained emoji embeddings with their short descriptions and demonstrated that emoji embeddings trained through this way were beneficial to sentiment analysis task.

We believe that the key to better classify emojis is understanding their meaning, since people intend a particular meaning when they send an emoji. People view the same characters during the exchange of plain text. Unlike plain text, emoji is not definite enough and doesn't have a general acknowledgement of how we should use it. It is common for different readers to have different interpretations of the same emoji, which naturally results in different ways of using emoji. Na'aman et al. (2017) investigated a wide range of emoji usage and showed that emojis served at least two very different purposes: content and function words or multimodal affective markers.

Word embeddings (Bengio et al., 2003; Mikolov et al., 2013a,b) are continuous distributed representations of words, with two good properties: 1. take word's semantic meaning into account, 2. distances between words are interpretable and can be measured using cosine distance. Based on such previous work, we proposed our *vector similarity based approach* for emoji prediction: first the neural network model is trained to generate a 300-d[1] vector, which is considered as the overall sentence vector of the tweet. Then this tweet vector's semantic similarity with

[*]corresponding author.

[1]The embedding dimension is 300 in this paper.

each emoji's pre-trained embedding is evaluated. The predicted label is the one with the highest similarity.

2 Approach Description

This section describes our approach in detail. It consists of two parts, one is tweet representation, the other is similarity computation between tweet vector and emoji embedding. Whether tweet vector or emoji embedding is text representation. Thus we start by discussing previous researches about text representation.

Many efforts have been made to generate vectors for variable-length texts such as phrases, sentences, paragraphs or documents (Mitchell and Lapata, 2010; Larochelle and Lauly, 2012; Mikolov et al., 2013b; Le and Mikolov, 2014). The generated vectors are of fixed-size, which can be used as input features for many machine learning methods.

Word embeddings are distributed word representations trained using word2vec models such as CBOW and skip-gram, which can be interpreted as the probability distribution of the context the word exists in. If we take emoji as a normal token and train it together with its context words using word2vec models, then its embedding represents the context this emoji may exists in. We associate a tweet with its related emoji using tweet's vector and emoji's embedding.

Formally, our vector similarity based approach can be described as follows: first the tweet's vector $\hat{\boldsymbol{y}}$ is generated using the neural network model, then its most similar emoji p is decided by calculating the cosine similarity(1) between $\hat{\boldsymbol{y}}$ and each emoji's embedding $\boldsymbol{y}_i$[2] in the candidate emoji set E whose size is 20.

$$cosine(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) = \frac{\hat{\boldsymbol{y}} \cdot \boldsymbol{y}_i}{||\hat{\boldsymbol{y}}|| \cdot ||\boldsymbol{y}_i||} \tag{1}$$

$$p = \underset{i}{\operatorname{argmax}}\ cosine(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \tag{2}$$

where $||\cdot||$ is L2 norm and p is the predicted emoji label.

During training, we use the opposite of cosine similarity as the loss function, which aims to make the generated tweet vector $\hat{\boldsymbol{y}}$ closer to the target emoji's embedding $\boldsymbol{y}$.

$$loss_1 = -\sum \left(\frac{\boldsymbol{y}}{||\boldsymbol{y}||} \cdot \frac{\hat{\boldsymbol{y}}}{||\hat{\boldsymbol{y}}||} \right) \tag{3}$$

[2] $\boldsymbol{y}_i$ can be found in pre-trained embeddings.

We also tried another loss function which has similar idea with SVM. That is, minimize the cosine distance(4) between $\hat{\boldsymbol{y}}$ and target emoji embedding $\boldsymbol{y}$, meanwhile maximize the minimum cosine distance between $\hat{\boldsymbol{y}}$ and non-target emoji embedding $\tilde{\boldsymbol{y}}$. We hope to make $\boldsymbol{y}$ more distinctive when similar emojis exist.

$$d(\boldsymbol{a}, \boldsymbol{b}) = 1 - cosine(\boldsymbol{a}, \boldsymbol{b}) \tag{4}$$

$$loss_2 = \alpha d(\hat{\boldsymbol{y}}, \boldsymbol{y}) - (1-\alpha) \min_{\tilde{\boldsymbol{y}} \in F} d(\hat{\boldsymbol{y}}, \tilde{\boldsymbol{y}}) \tag{5}$$

where α is a parameter to control the proportion of each part, F is the set which consists of 19 non-target emojis' pretrained embeddings for this tweet.

3 Models

This section describes the two models we used for generating tweet vector. Barbieri et al. (2017)'s previous work showed that LSTM neural networks performed well in emoji prediction. Inspired by their research, we implement two LSTM based models: a 2-layered LSTM model and a BiLSTM model.

3.1 2-layered LSTM

Our first model is a 2-layered LSTM model. This model consists of one trainable embedding layer for mapping words into vector representations, two stacked LSTM layers for processing and extracting useful information from the tweet, and one dense layer outputs the tweet vector.

Our experiments show that 2-layered LSTM works better than single layer LSTM. When stacked LSTM layer num gets larger than 2, the system performance doesn't increase much. Besides, deeper network structure costs more time to train and more parameters make it easy to overfit.

Long Short-Term Memory network, or LSTM (Hochreiter and Schmidhuber, 1997) is an enhanced version of basic recurrent neural network (RNN), which uses purpose-built memory cells to store information selectively (Graves et al., 2013). LSTM model can better exploit long range context, and is widely used in natural language processing tasks.

3.2 BiLSTM

Our second model is a bidirectional LSTM (BiLSTM) model (Schuster and Paliwal, 1997). This model consists of one trainable embedding layer, one bidirectional LSTM layer, and one dense output layer.

BiLSTM splits the neuron of a regular LSTM into two directions, one for positive time direction (forward states), 5and another for negative time direction (backward states). Output state $\boldsymbol{o}_i$ can be the concatenation or summation of the forward and backward state $\boldsymbol{fw}_i$ and $\boldsymbol{bw}_i$:

$$\boldsymbol{o}_i = \boldsymbol{fw}_i \odot \boldsymbol{bw}_i \tag{6}$$

where $\odot$ operator can be concatenate, element-wise add, etc.

4 Experiments

Our system is implemented using Keras[3] and the code is available on github[4]. We use the official evaluation metric *macro f1*, which evaluates both precision and recall of each class regardless of its sample num.

Three groups of experiments are achieved to evaluate our approach and models. To compare the vector similarity based approach with the classification approach, we implement the above 2-layered LSTM model and BiLSTM model with the same experiment settings for both approaches. To figure out which model structure is better, we compare the 2-layered LSTM model and BiLSTM model's performance on both approaches. We also test the loss functions $loss_1$ and $loss_2$'s effects on 2-layered LSTM model. Next, we will describe the key experiment settings. More detailed model settings can be found in Table 1.

Item	**Value**	**Description**
data_set	466,233	train set size
optimizer	Adam	train optimizer
batch_size	128	batch size
max_len	20	fixed sequence len
num_words	58,205	vocab size
2-layered LSTM model		
lstm1_size	300	lstm1 output size
lstm2_size	300	lstm2 output size
BiLSTM model		
lstm_size	300	lstm output size
bilstm_size	300	bilstm output size
merge_mode	sum	bilstm merge mode

Table 1: Experiment settings.

Model		**Valid**	**Test**
CL	2-LSTM	27.119	**25.678***
	BiLSTM	26.492	25.166*
VS	2-LSTM($loss_1$)	**27.188**	25.444
	2-LSTM($loss_2$)	27.089	25.496
	BiLSTM($loss_1$)	26.441	25.281

Table 2: Experiment Results. CL for classification approach, VS for vector similarity based approach, 2-lstm for 2-layered LSTM model. The results marked with asterisk (*) are our submissions for final evaluation.

Text Preprocessing: The whole tweet is lowercased. We split it into token sequence using Keras' default tokenizer, which split a sentence by spaces and following punctuations: `!"#$%&()*+,-./:;<=>?@[\]^_`~\t\n{|}`. Long sequences are truncated and short ones are padded with 0s from the head to meet fixed length 20.

Embedding Layer: The embedding layer is set to be trainable. It is initialized by looking up from a pre-trained twitter embedding matrix (Barbieri et al., 2016a), `<UNK>` is initialized as $\mathbf{0}$.

Output Layer: For classification approach, the output layer's unit num is 20 (same with the num of emoji classes). For vector similarity based approach, the output layer's unit num is 300 (same with the size of the pre-trained emoji embedding).

Training Loss: For our vector similarity approach, $loss_1$ and $loss_2$ described in section 2 are tested separately. For classification approach, *categorial_cross_entropy* is used.

5 Discussion

As is shown in Table 2, the vector similarity based approach's performance is comparable with the classification approach on both validation set and test set.

For both our vector similarity based and classification approaches, the 2-layered LSTM model outperforms the BiLSTM model, which shows that a deeper network structure contributes to capturing higher level features. Our 2-layered LSTM model consists of two stacked LSTM cells which are combined vertically. The first LSTM layer learns the shallower representation of the tweet, the second LSTM layer learns more abstract representation. Our BiLSTM layer also consists of two

[3]https://keras.io/

[4]https://github.com/MonkandMonkey/amyjing_emoji_predict.git

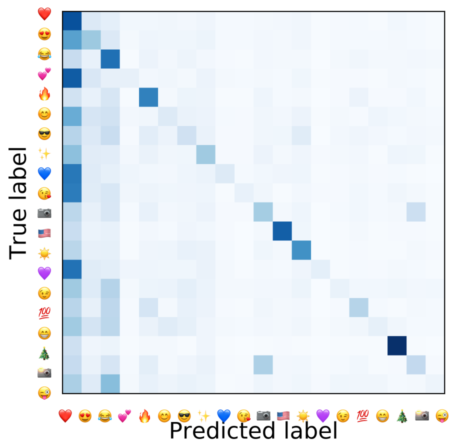

Figure 1: classification BiLSTM: confusion matrix

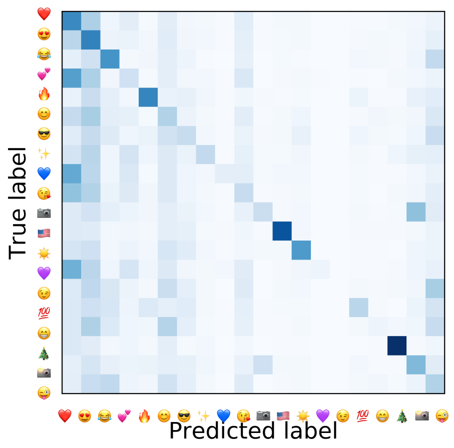

Figure 2: vector similarity BiLSTM: confusion matrix

LSTM cells, but are combined in a horizontal way. With the same number of parameters, the deeper structure (2-layered LSTM) works better than the wider structure (BiLSTM).

Experiments show that the $loss_1$ is slightly better than the $loss_2$ function. They are different in that $loss_1$ only considers the most similar emoji's distance, whereas $loss_2$ considers both most similar emoji's distance and the second most similar emoji's distance. We tested several α values in $loss_2$ from 0.8 to 0.99, and 0.9 gives the best performance, which is also dominated by the most similar emoji's distance.

Figure 1 and Figure 2 plot the confusion matrix of BiLSTM model's predictions for classification and our vector similarity approach. The ❤️ (red heart) column in Figure 1 shows that the classification approach tends to misclassify other classes into the most frequent emoji ❤️. And for emojis with similar semantics, it is more likely to confuse them. Like the 😘 (face-throwing-a-kiss) row, the classification approach misclassified most 😘 to ❤️, whereas the vector similarity based approach only misclassified a smaller part of them, and its correctly predict num is relatively higher. In short, the classification approach is good at distinguishing emojis with concrete meanings, such as 🔥, 🇺🇸, and 🎄, but poor at distinguishing emojis with similar semantic meanings. The vector similarity based approach can make a trade-off between both situations.

Besides, for both our proposed approach and classification approach, the performance on *test set* is relatively lower than that on *validation set*. Thus dataset will also make an influence, especially for tweet, which is time-sensitive text. If the test set contains many words that unseen during the training stage or its class distribution differs from the training set, the performance will be influenced.

6 Future Work and Conclusion

The pre-trained embeddings we used are trained with a skip-gram model, which treats emojis and words equally, whereas for this task we need to concentrate more on emoji's semantic, instead of its syntactic. Thus we suppose that treating emoji in a different way from word during the training stage will do a favor. That is, whether the emoji is in the head, center or tail of the tweet, its relative part can be used to train the emoji's embedding, despite it is outside the context window. Another attempt worth trying is to use a Logistic Regression or Linear SVM classifier to find tweet's most appropriate emoji, instead of cosine similarity.

In this paper, we present our work for SemEval-2018 task 2: Multilingual Emoji Prediction. We propose a vector similarity based approach which generates a vector for tweet and then use cosine similarity to find its most appropriate emoji. Through which we hope to explore the relationship between words and emojis. Experimental results show that the vector similarity based approach performs comparably with the classification approach. It provides an innovative thinking for solving the emoji prediction problem.

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. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*.

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016a. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016b. What does this emoji mean? a vector space skip-gram model for twitter emojis. In *LREC*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

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

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pages 2708–2716.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Hannah Miller, Jacob Thebault-Spieker, Shuo Chang, Isaac Johnson, Loren Terveen, and Brent Hecht. 2016. Blissfully happy" or "ready to fight": Varying interpretations of emoji. *Proceedings of ICWSM*, 2016.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Noa Na'aman, Hannah Provenza, and Orion Montoya. 2017. Varying linguistic purposes of emoji in (twitter) context. In *Proceedings of ACL 2017, Student Research Workshop*, pages 136–141.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.