EmojiIt at SemEval-2018 Task 2: An Effective Attention-Based Recurrent Neural Network Model for Emoji Prediction with Characters Gated Words

Shiyun Chen, Maoquan Wang, Liang He
Department of Computer Science and Technology
East China Normal University
Shanghai, P.R.China 200241
51174506002@stu.ecnu.edu.cn, maoquanwang@ica.stc.sh.cn, lhe@cs.ecnu.edu.cn

Abstract

This paper presents our single model to Subtask 1 of SemEval 2018 Task 2: Emoji Prediction in English. In order to predict the emoji that may be contained in a tweet, the basic model we use is an attention-based recurrent neural network which has achieved satisfactory performs in Natural Language processing. Considering the text comes from social media, it contains many discrepant abbreviations and online terms, we also combine word-level and character-level word vector embedding to better handling the words not appear in the vocabulary. Our single model[1] achieved 29.50% Macro F-score in test data and ranks 9^{th} among 48 teams.

1 Introduction

SemEval-2018 shared task 2 (Barbieri et al., 2018) provides a platform for us to explore the relationship between text and emoji in Twitter. The participants are expected to predict, given a tweet in English or Spanish, its most likely associated emoji. As the number of frequently-used emojis up to 20, this is also a multi-category classification task. Overall, both the prediction of emoji and the multi-category classification have undoubtedly increased the difficulty of task 2.

Before choosing a suitable model, we first analyze the data characteristics in detail. For each emoji, we count the total number of all tweets words under this emoji and enumerate the top 10 meaningful words with the highest frequency. And we can explore the following observations from the statistics:

- In almost all emojis, the first three words that appear frequently include "@user" suggesting that Twitter is a highly interactive social networking site (all the person names unified as "users" during the data preprocessing). Such a tweet should include an unwritten but important context information.

- Because people do not always have the same understanding of emojis, some words can evoke more than one emoji. The greater the repetition of high-frequency words among different emojis, the harder it is to distinguish among these tweets.

- There are also some easily recognizable words. They are just common words under a certain emoji, almost do not appear under others, such as "lit" and "fire" under 🔥; "photo" under 📷; "usa", "america" and "vote" under 🇺🇸; "beach" and "summer" under ☀️.

- When glancing over the data, we find that tweets have the following characteristics: a) Non-standard language, some discrepant shorthand or Internet jargon causes the model to get confused about the word; b) Misspelled words, although the human can easily realize the correct words, but if the model takes word-level as the processing unit, it cannot process them correctly. All of these can affect the ability of text representation seriously.

Taking all of these factors into consideration, we explore solutions form three perspectives:

- The second observation indicates that traditional statistical-based model may not achieve good performance, so we decide to adopt a neural network.

- Considering that the importance of each word is different in one tweet although text is very

[1] https://github.com/wwmmqq/SemEval-2018-Task-2-Multilingual-Emoji-Prediction

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 423–427
New Orleans, Louisiana, June 5–6, 2018.

short no more than 140 words, we introduce an attention mechanism to extract the key words in tweets.

- As the last observation can generate some tokens do not appear in the vocabulary (OOV), we utilizes both word-level and character-level word vector embedding.

In a word, our single model is an attention-based neural network with word-level and character-level sentence embedding.

2 System Description

Our model architecture is depicted in Fig. 1 and it consists of the following three subparts: word embedding layer, BiLSTM layer and attention layer.

2.1 Word Embedding Layer

Assuming there are n words in a giving sentence x, we denote it as $x = \langle w_1, w_2, \cdots, w_n \rangle$. At each time step t, both the word lookup table and a bidirectional LSTM take the same word w_t as an input.

Word-level embedding The word-level input is projected into a high-dimensional space by a word lookup table $\mathbf{E} \in \mathbb{R}^{|\mathbf{V}| \times \mathbf{d}}$, which you can get from publicly available *word2vector* [2] tool, where $|V|$ is the vocabulary size and d is the dimension of a word vector. Then we refer to the obtained vector as $\boldsymbol{x}_{w_t}^{word}$.

Character-level embedding The character-level input is converted into a vector by using a recurrent neural network. In order to better capture the interaction between adjacent characters, we use a bidirectional LSTM (BiLSTM) (Graves, 2005) to model the words. BiLSTM contains a forward LSTM processing the input from the first char to the last char, and a backward LSTM performing the opposite action. The last hidden states of the forward and the backward recurrent networks are linearly combined to $\boldsymbol{x}_{w_t}^{char}$.

Combine word-level and character-level embedding We generate the final vector representation of a word by combining two distinct representations of the word:

$$\boldsymbol{x}_{w_t} = g_{word}\boldsymbol{x}_{w_t}^{word} \oplus g_{char}\boldsymbol{x}_{w_t}^{char} \quad (1)$$

where $\oplus$ is a concatenation operator, g_{word} and g_{char} are weights which can be calculated as:

$$g_{word} = \sigma(\boldsymbol{W}_1\boldsymbol{x}_{w_t}^{word} + b_1) \quad (2)$$

[2] https://https://code.google.com/p/word2vec

$$g_{char} = \sigma(\boldsymbol{W}_2\boldsymbol{x}_{w_t}^{char} + b_2) \quad (3)$$

here $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, b_1 and b_2 are trainable parameters, $\sigma(\cdot)$is a sigmoid function.

2.2 BiLSTM Layer

Recurrent Neural Network (RNN) (Elman, 1990) (Mikolov et al., 2010) (Chung et al., 2014) is proposed for modeling long-distance dependence in a sequence, but it tends to suffer from the gradient vanishing and exploding problems. RNN with Long Short-Time Memory Network unit (Hochreiter and Schmidhuber, 1997) solves such problems by introducing a memory cell and gates into the network. In order to better model the tweets, we use a bidirectional LSTM(BiLSTM) (Graves et al., 2014) (Graves, 2005) to process the inputs.

Each single LSTM ($LSTM_{forward}$ and $LSTM_{backward}$) can be formalized as shown in (Gers and Schraudolph, 2003):

Then we use $\oplus$ to obtain the hidden layer representation as below:

$$\overrightarrow{\boldsymbol{h}}_{w_t} = LSTM_{forward}(\boldsymbol{x}_{w_t}, \overrightarrow{\boldsymbol{h}_{w_{t-1}}}) \quad (4)$$

$$\overleftarrow{\boldsymbol{h}}_{w_t} = LSTM_{backward}(\boldsymbol{x}_{w_t}, \overleftarrow{\boldsymbol{h}_{w_{t-1}}}) \quad (5)$$

$$\boldsymbol{h}_t = \overrightarrow{\boldsymbol{h}}_{w_t} \oplus \overleftarrow{\boldsymbol{h}}_{w_t} \quad (6)$$

here $\overrightarrow{\boldsymbol{h}}_t$, $\overleftarrow{\boldsymbol{h}}_t$ are the hidden layer states for single word w_t, which generate from the combination of the current word information $\boldsymbol{x}_{w_t}$ and previous state information $\overrightarrow{\boldsymbol{h}_{t-1}}(\overleftarrow{\boldsymbol{h}_{t-1}})$. For simplicity, we note all the n $\boldsymbol{h}_t$ as $H = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n)$.

2.3 Attention Layer

In our model we introduce an attention mechanism to focus on certain part of the tweet (Santos et al., 2016) (Yang et al., 2017). The motivation mainly comes from the following two observations: a) although tweet is short, it always introduces much noise; b) the emoji decided by a relative small part of tweet content itself. The attention layer produces a weight vector applied to hidden states of BiLSTM.

The attention mechanism takes the whole hidden states H as input, and outputs a vector of weights a:

$$a = softmax(\boldsymbol{W}_3 tanh(\boldsymbol{W}_4 H^T)) \quad (7)$$

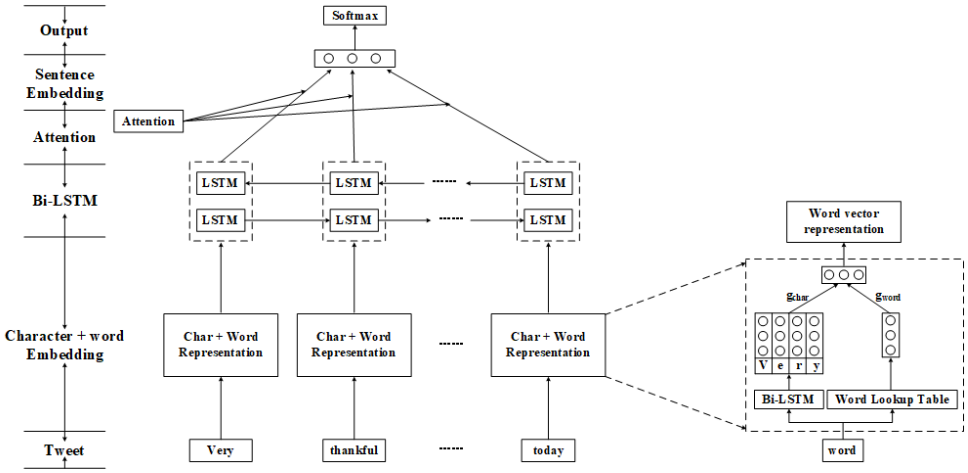

Figure 1: Our model architecture

here $\boldsymbol{W}_3$ and $\boldsymbol{W}_4$ are trainable parameters, the $softmax()$ ensures all computed weights sum up to 1.

Then we sum up the BiLSTM hidden states H according to the weight provided by a to get the input representation $\boldsymbol{r}$:

$$\boldsymbol{r} = aH \tag{8}$$

2.4 Output

At the output layer, we need to predict the emoji evoked by the tweet. We use the $softmax$ function to predict the probability distribution over all emojis:

$$p(\hat{y}|\boldsymbol{r}) = softmax(\boldsymbol{W_5 r} + \boldsymbol{b}_5) \tag{9}$$

where $\boldsymbol{W}_5$ and $\boldsymbol{b}_5$ are the parameters of $softmax$ function.

The training loss based on cross-entropy is defined as follow:

$$J(\theta) = -\frac{1}{N}\sum_{t=1}^{N} \mathscr{H}(y_i, \hat{y}_i) + \beta \left\|\theta\right\|^2 \tag{10}$$

where θ is the whole trainable parameters of the model, and β is the weight for the regularization term. $\mathscr{H}$ is the cross-entropy function for instance i between the gold category y_i and predict category $\hat{y}_i$. N is the number of training instance.

3 Experiment

3.1 Dataset

SemEval-2018 provided 500k tweets in English and 100k in Spanish for task 2, while our model is only for English. Dateset is split into trial, training and test sets. However, it was unable to download all the training set because some tweets were deleted or not available due to modified authorization status, and we finally collected $471,455$ training tweets.

3.2 Data Pre-processing

All of the tweets before feeding to any model are pre-processed as follows: all tweets are lowercased; URLs are replaced by $< url >$ token; USERNAMEs are replaced by @user token; NLTK3 is employed to tokenize input tweets.

3.3 Training

We use the 200-dimensional vectors (Barbieri et al., 2016) to initialize our words matrix E, and set max length of one word to 10, max length of one sentence to 30. All of the models are trained for 10 epochs, and the final model is selected by trial dataset. Adam optimizer (Kingma and Ba, 2014) with initial learning rate of 0.001 is used to minimized cross-entropy loss. The learning rate is reduced by a factor of 10 for the first 3 epoch. The models were implemented in TensorFlow and experiments were run on an Intel(R) Core(TM) i7-4790 CPU.

3.4 Baselines

We compare our model with the following baselines: FastText (Joulin et al., 2016), NBOW, Convolutional Neural Networks (CNN) (Kim, 2014) (Kalchbrenner et al., 2014), Recurrent Convolutional Neural Networks (RCNN) (Sentences are first separately embedded with CNN, and then joined up with RNN) (Kalchbrenner and Blunsom,

Model		Representation	F1	P	R	Acc
Basic	FastText	Word Only	25.19	30.07	24.60	34.74
	NBOW		24.62	28.19	24.28	32.98
	CNN		20.34	22.54	23.00	34.73
	RCNN		26.89	26.97	29.62	32.53
	BiLSTM		20.36	22.41	23.00	34.74
	BiLSTM+ATT		24.95	34.87	26.34	36.44
Basic	C2W2S+ATT	Char Only	27.77	28.39	28.01	34.56
Our Submit	C2W2S+ATT	Char+Word	**29.50**	**35.17**	**29.91**	**39.21**
Rank 1: cagri		-	35.99	36.55	36.22	47.09

Table 1: Experimental results on test data and result of rank 1 system

2013), BiLSTM (Graves et al., 2014) and BiLSTM+ATT (Graves et al., 2014).

3.5 Results and Discussion on Training Data

A series of comparison experiments on training set have been performed to explore the performance of our model in macro F-score(F1), precision(P), recall(R) and accuracy(Acc). The experimental results are shown in Table 1 in percentage. For the first part, we choose some basic models which only use the word-level word vector embedding as input; for the second part, we present ablation experiments showing usefulness of word-level and character-level embedding.

Statistics show the following three conclusions: 1) among the basic model, the best performance according to different evaluation metrics achieved by RCNN (F1), BiLSTM+ATT (Precision), RCNN (Recall) and BiLSTM+ATT (Accuracy) respectively; 2) when compare the two model (Word Only and Char Only) about BiLSTM+ATT, word-level model and character-level model complement each other on the four measurements. So we consider combining the two to obtain Char+Word BiLSTM model; 3) as the ablation experiments show, our model can get best result when combine word-level input with character-level input.

4 Results on Test Data

Detailed official results of our model are shown in Table 2. And we focus on analyzing performance of our model under each emoji combined with the characteristics of tweets summarized in section 1.

On the one hand, our model achieved promising results in the following emojis:🎄 (F1: 69.06), 🇺🇸 (F1: 64.01), and 🔥 (F1: 55.46). As for the reason, section 1 gives us inspiration: the common words evoke these emojis are easily recognizable, in other words, the word "christmas" evoking emoji 🎄 is barely used in emoji 🇺🇸.

Emo	P	R	F1	%
❤️	40.3	64.38	49.57	21.6
😍	30.82	32.96	31.86	9.66
😂	38.6	55.56	45.55	9.07
💕	31.23	4.57	7.97	5.21
🔥	57.83	53.28	55.46	7.43
😊	13.39	8.56	10.44	3.23
😎	22.11	17.94	19.81	3.99
✨	33.41	26.26	29.41	5.5
💙	36.05	10.26	15.98	3.1
😘	20.81	7.83	11.38	2.35
📷	31.9	53.7	40.02	2.86
🇺🇸	64.94	63.11	64.01	3.9
☀️	36.24	50.59	42.23	2.53
💜	56.49	7.81	13.72	2.23
😉	16.33	9.19	11.76	2.61
💯	32.18	26.13	28.84	2.49
😁	20.39	6.42	9.76	2.31
🎄	64.12	74.82	69.06	3.09
📸	43.31	23.05	30.08	4.83
😜	12.95	1.78	3.13	2.02

Table 2: Precision, Recall, F-measure and percentage of occurrences in the test set of each emoji.

On the other hand, for those emojis that are easily confused, our model does not perform well, such as 😜, 💕 and 😁. So we analyze data and find that these three emojis have the highest rate of repetition of common words with other emojis. For example, the common words under emoji 💕 include "love" (top2) and "happi" (top5), but the two words also evoke ❤️ and 😊 frequently.

Overall, the Macro F-score of our model ranks 9^{th} in the subtask 1.

References

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, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. 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.

Junyoung Chung, Caglar Gulcehre, Kyung Hyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Eprint Arxiv*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Felix A. Gers and Nicol N. Schraudolph. 2003. *Learning precise timing with lstm recurrent networks*. JMLR.org.

Alex Graves. 2005. *2005 Special Issue: Framewise phoneme classification with bidirectional LSTM and other neural network architectures*. Elsevier Science Ltd.

Alex Graves, Navdeep Jaitly, and Abdel Rahman Mohamed. 2014. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding*, pages 273–278.

Seppu Hochreiter and Jrgen Schmidhuber. 1997. *Long short-term memory*. Springer Berlin Heidelberg.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *Computer Science*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Eprint Arxiv*, 1.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September*, pages 1045–1048.

Cicero Dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2017. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.