Emotionally-Triggered Short Text Conversation (STC-3) using Attention-Based Sequence Generation Model

Sébastien Montella[1, 2], Chia-Hui Chang[1] and Frederic Lassabe[2]

[1] National Central University, Taoyuan, Taiwan
[2] University of Technology of Belfort-Montbéliard, Belfort, France
sebastien.montella@utbm.fr, chia@csie.ncu.edu.tw, frederic.lassabe@femto-st.fr

Abstract. Recent studies have significantly contributed to high-quality text generation. Different frameworks were leveraged to build sophisticated models for language generation. However, automatic generation of emotional content has barely been studied. We propose in this paper an Attention-Based Sequence Generation Model for Emotionally-Triggered Short-Text Conversation (STC). We use emotion category embeddings to represent different emotions and to trigger the generation of the stated emotion. The emotion vectors are learned during training allowing the model to have a degree of freedom to modify those vectors accordingly. Our attention mechanism is customized to include emotional information by using gated convolutional neural networks (GCNNs) in order to create an emotional context vector. Moreover, we utilize distinct Start-Of-Sentence (SOS*)* token for each emotion category in order to further push our sequence-to-sequence model into a specific *emotional generation mode*. This approach avoids the implementation of different generative models for each emotion. Experimental results demonstrate the ability and difficulty of our architecture to generate affective answers.

Keywords: short-text conversation, emotional intelligence, generative model.

Team Name: WIDM

Subtask: Chinese Emotional Conversation Generation (CECG)

1 Introduction

Natural Language Generation (NLG) is often pointed as one of the most difficult challenge in Artificial Intelligence (AI). Nowadays, improvements in both grammar and response-consistency indicate the remarkable progress of language generation. In addition, the amount of data on social media has massively helped the development of deep learning-based generative models.

Meanwhile, emotional intelligence is getting more and more attention. Indeed, generating adapted sentences in term of affective content can reduce the gap between the user and the machine. Sentiment and emotion classifiers have already proven encouraging results on different datasets. Thus, sentimental text generation using deep learning techniques has begun to attract researchers' attentions. Previous works focused mainly on dictionary-based or rule-based approaches. With the great success of deep learning methods, the use of large-scale data can alleviate the cost of building rules or a corpus for each emotion category.

In this study, we aim to generate grammatically correct and emotionally consistent response to an input post with regard to a given emotion. We consider this emotionally-triggered short-text conversation task as a sequence generation problem by evaluating the probability to generate a character given the previous generated characters with respect to the input post and the emotion. The challenge here is double. Indeed, generated answers should not only be meaningful to the post but should also come with the correct emotion.

2 Related Work

With the expansion of data from the social media, the amount of data available for Short-Text Conversation (STC) became incredibly huge. Early works focused on building retrieval-based systems which rely on the ranking model/a scoring function to retrieve comment for response generation. The scoring function is therefore the key component of such system. Usually, a first phase consists of retrieving a list of pairs as response candidates with a first scoring function, and during the second phase, a re-ranking is performed to further estimate scores for those candidates. Retrieval-based approach is extremely useful when the comment needs to be grammatically correct since selected comments are human-generated. However, such systems heavily rely on the scoring function which may not catch the intrinsic meaning.

With recent works on words and sentences representation [1] [2], deep learning is widely used in STC task [3] [4]. For example, Recurrent Neural Network (RNN) and the encoder-decoder architecture are designed to map sequential input to sequential output [5] [6]. More recently, Generative Adversarial Networks (GANs) [7], a new trend to train generative models using an adversarial loss, is adopted for language generation. From this work, SeqGAN were introduced [8] to generate textual content, using a policy gradient due to the discrete variable outputs. However, the training of GANs is often unstable, and hard to converge. Nevertheless, [9] has successfully reused SeqGAN to build a new framework coined SentiGAN to generate texts of different sentiment labels. SentiGAN uses k generators to generate k different sentences of different sentiment, and a multi-class classifier as the discriminator. The drawback of this approach is the need of multiple generators.

In [10], an emotional chatting machine is modeled based on the encoder-decoder architecture with an internal and external memory. The internal memory consists of an internal emotion state. During the generation, the emotion state is decayed and should reach zero when the generation is finished. The external memory is assigning

probabilities to generate whether emotion words ('amazing', 'terrible', 'awesome', 'bad', etc.) or generic words ('car', 'computer', 'issue', etc.). A list of emotion words need to be established.

In this work, we propose an Attention-Based Sequence-to-Sequence using a single model and without any external emotion corpus for Emotionally-Triggered language generation. We make use of emotion embeddings to represent emotion and to trigger the generation of the specific emotion. In addition, we use different Start-Of-Sentence (SOS) token for each emotion category to put our model into the specific emotion mode.

3 Attention-Based-and-Emotionally-Triggered Sequence-to-Sequence

3.1 Background : Sequence-to-Sequence with Attention Mechanism

Our Sequence-to-Sequence is modeled using the encoder-decoder architecture with LSTM cells. In this work, the encoder is a Bidirectional LSTM that takes as input a sequence $X = (x_1, x_2, \dots, x_{N-1}, x_N)$ and generates the hidden states $H = (h_1, h_2, \dots, h_{N-1}, h_N)$ as shown in Eqs. (1) - (3), where x_j is the character embedding of character p_j of the input sentence, $x_j = e_{char}(p_j)$. We denote [] as the concatenation operator.

$$h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}] \tag{1}$$

$$\overrightarrow{h_j} = \overrightarrow{LSTM}(\overrightarrow{h_{j-1}}; x_j) \tag{2}$$

$$\overleftarrow{h_j} = \overleftarrow{LSTM}(\overleftarrow{h_{j+1}}; x_j) \tag{3}$$

The last hidden state h_N, called the context vector, is given to the decoder as the initial hidden state $s_0 = h_N$ from which the decoding process starts when the Start-Of-Sentence (SOS) token is also given. At each time step, the decoder predicts the next character by generating the probability distribution over the entire vocabulary with a softmax function, where the output y_i is computed from Eq. (4) with $s_i = [\overrightarrow{s_i}; \overleftarrow{s_i}]$ being the concatenation of BiLSTM operation with the embedding of the previously generated character y_{i-1}.

$$y_i = softmax(W_o s_i) \tag{4}$$

$$\begin{cases} \overrightarrow{s_i} = \overrightarrow{LSTM}(\overrightarrow{s_{i-1}}; [v_i; c_i]) \\ \overleftarrow{s_i} = \overleftarrow{LSTM}(\overleftarrow{s_{i+1}}; [v_i; c_i]) \end{cases} \tag{5}$$

$$v_i = [e_{char}(q_i); E_q] \tag{6}$$

During training phase, the input to decoder layer is the output sequence $Q = [q_1, q_2, \dots, q_M]$, where $q_1 = \text{SOS}$ denotes the start of a sentence. During testing, the *i*th

input q_i is the output of the decoder at time step $i-1$, *i.e.* the previous generated character y_{i-1}.

To further improve the decoding process, a common technique used recently is attention mechanism which makes the generation sensitive to the input sentence [11] [12]. Thus, the forward and backward hidden states are updated by merging the context vector c_i to the decoder input computed at time step i by Eqs. (7) – (9).

$$c_i = \sum_{j=1}^{N} \alpha_{ij} h_j \tag{7}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{N} exp(e_{ik})} \tag{8}$$

$$e_{ij} = f_a^T . tanh(W_a[h_j;\ s_{i-1}] + b_a) \tag{9}$$

The parameters $f_a, b_a \in R^d$ and $W_a \in R^{d \times m}$ are trainable parameters with the encoder hidden state h_j and the decoder hidden state s_{i-1}. However, in order to emphasize on the given emotion during decoding process, we make use of different techniques describes in the next subsections

Emotion Embedding

Emotions can be described with different methods. Instead of using simple one-hot encoding, one may consider using arousal and valence dimensions to have a 2-D vector to represent the emotion of a sentence. Arousal and valence characterize the intensity and the polarity of an emotion, respectively. In this paper, we adopt an emotion embedding learning approach. We randomly initiate a vector for each discrete emotion. During the training phase, these emotion embeddings are automatically updated with regard to the given emotion needed to be expressed for the sentence Q. Therefore, at each time step, the decoder is not only fed with the character embeddings of the ground truth of the previous time step character, but also with the emotion vector. We expect by using such technique to better guide our model to generate sentences with the specified emotion. In addition, since those emotion vectors are learned during training, it gives to the model some degree of freedom.

However, considering this lonely technique doesn't guarantee the generation of emotional-consistent response. Thus, we adopt extra means to strongly emphasize on the emotion.

3.2 Emotional Context Vector

In the vanilla attention mechanism, a dynamic context vector is computed at timestep i to emphasize on which part of the input the decoder should focus to generate the i-th character (Eq. (7)). Here, our intention is to mix our emotion and the context vector together to get a more sophisticated context vector, termed *emotional context vector*. The challenge is to generate characters that are more meaningful and consistent with the post and the emotion respectively. The emotional context vector is built by firstly concatenating the vanilla context vector c_i with the emotion vector E_Q. The

resulting vector is fed into a simple feed-forward neural network to produce a new vector $\in R^d$, with trainable parameters $W_b \in R^{d \times m}$ and $b_b \in R^d$ (Eq. (10)).

$$ec = tanh\left(W_b\left[c_i; E_Q\right] + b_b\right) \tag{10}$$

$$\begin{cases} A = W_A \bigoplus ec + b_A \\ B = W_B \bigoplus ec + b_B \end{cases} \tag{11}$$

$$c_i^+ = A \odot \sigma(B) \tag{12}$$

The derived vector ec can be seen as a mixture of the information from c_i and E_q in each dimension. Next, we use gated convolutional neural networks (GCNNs) on ec to supervise and control the information that should be provided to the decoder. We hope that GCNNs can learn latent features from ec, *i.e.* the mixture of input sequence and the given emotion, and to select parts of each dimension that should be further streamed to the decoder. Formally, W_A, W_B and b_A, b_B are weights and biases of two CNNs respectively, and we note $\bigoplus$ the one-dimensional convolution operation and $\odot$ the Hadamard product (Eq. (11) and Eq.(12)). The first CNN creates a new vector $A \in R^d$ that is modulated thanks to a sigmoid function with $B \in R^d$, the output of a second CNN. The new obtained vector c_i^+ is the emotional context vector for time step i. The emotion context vector replaces our vanilla context vector in Eq. (5).

3.3 Emotion Start-Of-Sequence Token

Traditionally, a SOS token is used to start the generation process in the decoder. The SOS token triggers the generation of the first character. Then, the embedding of the first character is used as input for the next time step to trigger the generation of the second character and so on.[1] However, to our knowledge, no previous work considered customizing the SOS token to initiate the generation of different content. Therefore, we make use of multiple SOS tokens. More specifically, we assign a specific SOS token for each emotion. Then, given the emotion to convey in the generated output comment, we choose the corresponding emotion SOS token to start the decoding process. Instead of training one decoder for each emotion, we expect that utilizing emotion SOS token will have similar effect and push our model into a specific *emotion generation mode*. The overall framework is shown in **Fig. 1**.

[1] In a teacher forcing approach.

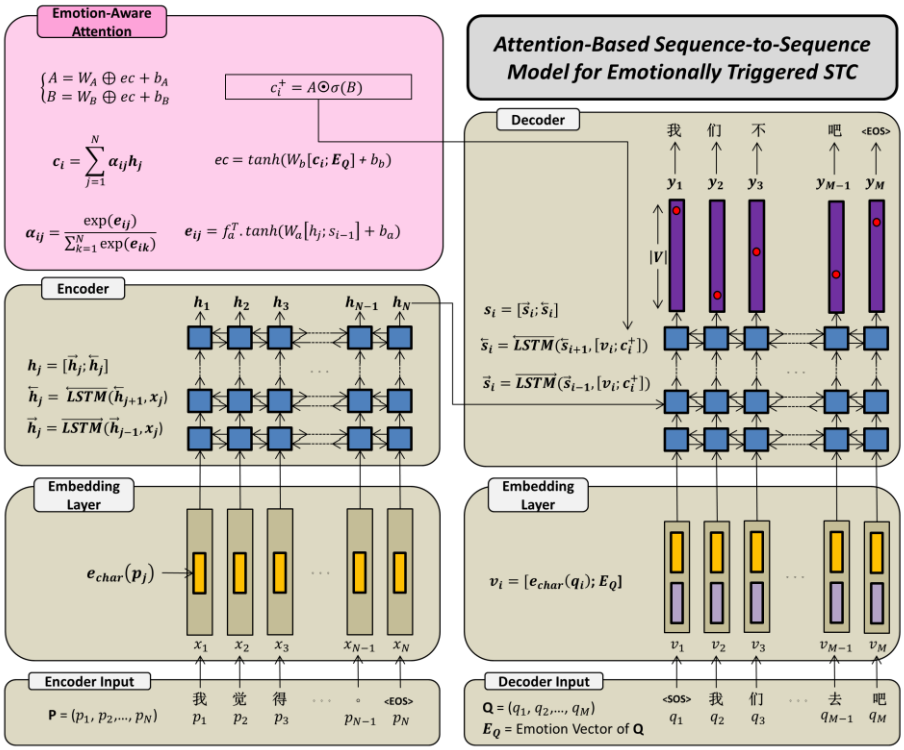

Fig. 1. Architecture of the Attention-Based Seq2Seq Model for Emotionally Triggered STC.

4 Experiments

Data and Data Pre-processing. We use a large corpus of post and comment pairs retrieved from *Weibo*[2]. We limit the length of the sentences to 30 characters to exclude too long phrases that might include some noise. Since emotions can be expressed with punctuation, excessively repetitive sequence of characters, and emoticons, the preprocessing should not remove too much of those. We then only apply a light preprocessing step on our sentences. We firstly transform traditional Chinese characters into simplified Chinese to reduce size of our vocabulary. We don't set any vocabulary size limit. In contrast of a word-based approach, a character approach allows to reduce the vocabulary size. Furthermore, in Chinese language, technical and complex words are often made of simple characters. Therefore, by using a character approach, we avoid to exclude such words of our vocabulary. In addition, when repetitive Chinese character such as '哈' and '呵' appear, those are replaced with only three occurrences to limit the size of the sentence so that more sentences can be included in our training data.

[2] *Weibo* is a Chinese social media, similar to Facebook or Twitter.

Moreover, both post and comment come with an emotion label. The different labels are 'Like', 'Sad', 'Disgust', 'Angry', 'Happy' and 'Other'. In this work, we only make use of the comment emotion label to guide the generation of the comment. The emotion labels are provided thanks to a pre-trained BiRNN LSTM classifier performing 64% accuracy on a six-way classification. Thus, emotion labels contain misclassified pairs that may significantly affect the performance of our model. To reduce the effect of misclassification, a solution consists in selecting pairs that are much more likely to be less noisy. Therefore, we decide to train two distinct models, *Model A* and *Model B*, on two different datasets, D_A and D_B respectively.

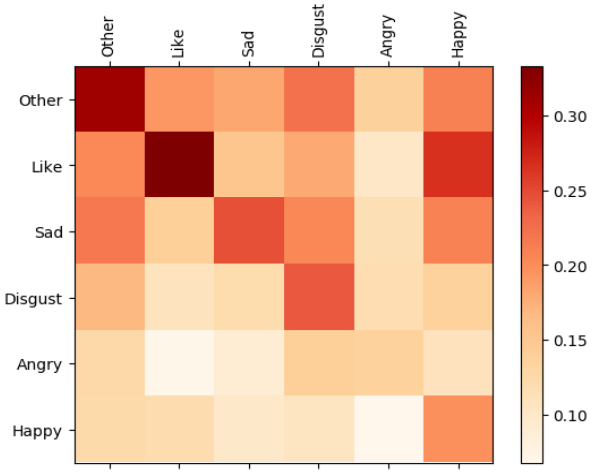

Fig. 2. Interaction between emotion of posts and comments.

The first dataset D_A is using the whole data without any pairs selection while D_B is only making use of post and comment pairs sharing the same emotion labels. As described in **Fig. 2**, there are higher probabilities that the emotion of a response matches the emotion of the input post. We then assume that emotion labels of those responses are less noisy and may be of better quality. However, the size of D_B is reduced by a factor of 4. Indeed, D_A and D_B gathers 1,537,182 and 387,060 pairs, respectively.

8

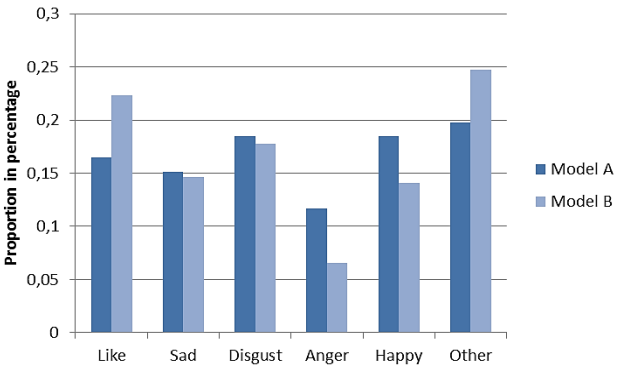

Fig. 3. Comments' emotions proportions (in %) in *Model A* and *Model B* training data.

We also plot in **Fig. 3** the proportion of each comment emotion appearing in each dataset used to train *Model A* and *Model B*. We observe that 'Sad' and 'Anger' emotions are less frequent than other emotions in our data.

Training. *Model A* and *Model B* are trained with the same parameters for comparison purpose. Implementation is done using Pytorch [3]. The character embeddings are trained during the learning process. We have for *Model A* and *Model B* a vocabulary size of 6,825 and 5,721, respectively. Size of the embeddings is set to 300. We use 4 layers with 1000 neurons per layer for both our encoder and decoder. In addition, we make use of 3 stacked GCNNs in our emotion-aware attention mechanism. A dropout of 0.1 acts as our regularization technique to avoid overfitting. Models are trained with a batch size of 100 by using the Adamax optimization algorithm [13] with a learning rate of 0.005. However, since the size of D_B is much smaller than D_A, we can faster train *Model B* compared to *Model A*. As a result, we decide to train *Model A* on 20 epochs while *Model B* is trained on 45 epochs. With similar parameters but different number of training iterations, we can compare the convergence of the models and compared the link of convergence in our training phase with our evaluation results.

4.1 Evaluation Results

In language generation, the evaluation of trained models is also challenging. It is hard to define metrics to automatically score the responses of the generative model. BLEU and ROUGE metrics are often mentioned in related work. These metrics try to compare the generated answers with human-generated ones based on their common words. However, there are different ways to express one's idea with different words, mainly when it comes to emotions. Thus, we use handcrafted evaluation where human can evaluate each prediction and lead to a better evaluation quality. In our case, 3

[3] Version 0.4.1 used (more at https://pytorch.org/).

people give score for each predicted sentence. Scores are actually number from 0 to 2 and are given based on its coherence, fluency and emotion consistency.

Therefore, to evaluate each model, we define an overall score that sums up scores of each sentence divided by the number of sentences (Eq. (13)). We note N_t the number of generated sentences and Num_i the number of sentences with the label i .

$$OverallScore = \frac{1}{N_t}\sum_{i=0}^{2} i * Num_i \quad (13)$$

If we plot the training loss of both *Model A* and *Model B* (**Fig. 4**), we can see that *Model B* converges and decreases its errors much faster than *Model A*.

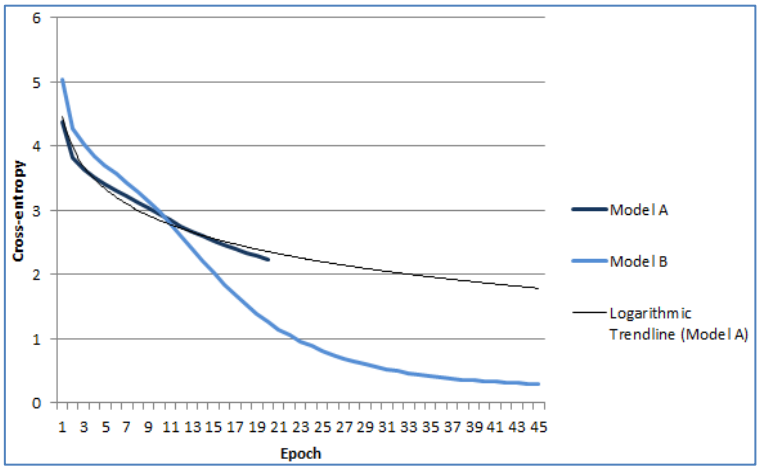

Fig. 4. Comparison of convergence of *Model A* and *Model B* during the training phase.

With less data, the second model is minimizing its loss function easier. On the contrary, *Model A* seems to be learning and converging with much more difficulty. However, from the comparison results in **Table 1**, we can see that *Model A* performs much better than *Model B*.

Table 1. Comparison of scores of *Model A* and *Model B* for each emotion

	Model *A*	Model *B*
Like	0.600	0.245
Sad	0.515	0.310
Disgust	0.495	0.180
Anger	0.455	0.210
Happy	0.525	0.200
Overall Scores	**0.518**	**0.229**

The *Model A* is outperforming the *Model B* for all emotion generation. Although the chance of a better data quality, we deduce that the size of D_B doesn't allow the *Model B* to learn how to generate grammatically correct and emotionally consistent

responses given the post and the emotion. On the contrary, *Model A* shows the capacity to produce reasonable responses. However, it is still difficult for *Model A* to explicitly express affective answers. The effect of misclassified sentences might be a core issue here where the model is learning how to generate characters from another emotion for the stated emotion category.

5 Conclusion & Future Work

In this paper, we propose an Attention-Based-and-Emotionally-Triggered Sequence-to-Sequence using a character-based approach allowing to generate responses that are meaningful given an input post and emotionally consistent to a given emotion parameter. In future work, we consider applying generative adversarial nets in order to make use of a discriminator acting as emotion classifier that can detect the emotion of generated responses. Such discriminator can give much meaningful feedback to the sequence-to-sequence model and may be giving less noisy feedback.

6 Acknowledgement

This work was done for the NTCIR Workshop [14] [15] that provided the dataset from *Weibo*. We also thank the anonymous annotators for labelling our data.

References

1. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean: Distributed Representations of Words and Phrases and their Compositionality, *Proceedings of the 26th International Conference on Neural Information Processing Systems* (2013).
2. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (*2013).
3. Yu-Han Chen, Sébastien Montella, Wei-Han Chen, Chia-Hui Chang: WIDM at the NTCIR-13 STC-2 Task, Proceedings of *NTCIR-13* Conference, Tokyo (2017).
4. Lifeng Shang, Zhengdong Lu, Hang Li: Neural Responding Machine for Short-Text Conversation, ACL (2015).
5. Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (2016).
6. Oriol Vinyals, Quoc V. Le: A Neural Conversational Model, Proceedings of the 27th International Conference on Neural Information Processing Systems, Montréal (2014).
7. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio: Generative Adversarial Nets, NIPS, 2016.
8. Lantao Yu, Weinan Zhang, JunWang, Yong Yu: SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient, Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, (AAAI-17).
9. Ke Wang, Xiaojun Wan: SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (2018).

10. Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, Bing Liu: Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory, Association for the Advancement of Artificial Intelligence (2018).
11. Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio: Neural Machine Translation by Jointly Learning to Align and Translate, ICLR (2015).
12. Ashish Vaswani, Noam Shazeer,Niki Parmar,Jakob Uszkoreit,Llion Jones,Aidan N. Gomez,Łukasz Kaiser,Illia Polosukhin: Attention Is All You Need, 31st Conference on Neural Information Processing System (2017).
13. Diederik P. Kingma, Jimmy Lei Ba: ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, ICLR (2015).
14. Huang M, Ye Z, Zhou H. : Overview of the Emotional Conversation Generation Challenge Task at NLPCC, NLPCC (2017).
15. Yaoqin Zhang, Minlie Huang: Overview of NTCIR-14 Short Text Generation Subtask: Emotion Generation Challenge, Proceedings of the 14th NTCIR Conference (2019)