RUCIR at the NTCIR-13 STC-2 Task

Yutao Zhu*, Xiangbo Wang, Xiaochen Zuo, Shuqi Lu, Zhengyi Ma, Xinyi Zhang
Renmin University of China, Beijing, P.R.C
ytzhu@ruc.edu.cn*

Zhicheng Dou
Renmin University of China, Beijing, P.R.C
dou@ruc.edu.cn

ABSTRACT

This paper describes the RUCIR's systems which participated in the Short Text Conversation (STC-2) task (Chinese) at NTCIR-13[9]. Contrary to the STC task in NTCIR-12, generation-based methods are considered and participants are encouraged to explore some effective ways to combine both retrieval-based and generation-based approaches to get a more intelligent chatbot. This paper introduces two systems we proposed: (1) a retrieval-based system (2) a generation-based system combined with retrieval-based methods. Besides, we do some analysis and discussion on the two systems based on the results.

Team Name

RUCIR

Subtasks

Short Text Conversation (Chinese)

Keywords

Sequence to Sequence Model, Attention Mechanism, Local and Global Keywords

1. INTRODUCTION

Human-computer conversation is a challenging task in natural language processing (NLP). Contrary to generic conversation system, short text conversation (STC) system focuses on handling short text dialog (i.e., both inputs and outputs of the system are shorter than normal sentence).

STC was first appeared as a task in NTCIR-12 (STC-1[1]) which tackles the following research goal: reuse an exists comment from repository to reply a given post. The data is collected from social media such as Twitter and Weibo, therefore the post and comment construct a single-round dialog. Intuitively, the retrieval-based methods considered in STC-1 are limited on generating "new" comments since all comments are retrieved from repository. STC-2 involves generation-based methods and encourages participants to design ensemble system which combine both retrieval-based approaches and generation-based approaches.

Retrieval-based methods consider the post as a query and search for a most relevant comment in the database as the result. Common retrieval-based dialog systems work in a retrieval-and-reranking strategy and focus on extracting more features to train a better ranking model or designing models with more complicated structure[5, 10]. Based on recent researches, we consider that an effective way to measure the similarity between post and comment is the keypoint for this task. Therefore, the process of our approach can be described as follows: (1) use open-source retrieval system (like Solr) to obtain the candidate post-comment pairs, (2) involve word embedding to represent each word as a vector and construct the sentence vector by weighted sum operation, (3) calculate the similarity between the input post and the candidate post-comment pairs by various distance functions (Cosine and Euclidean), (4) rank the candidate list and return the top n comments as the results.

Recent years, neural network based methods bring new opportunities to many NLP tasks, especially for dialog generation[8]. Sequence to sequence (Seq2Seq) model[1] is one of the most elegant RNN-based frameworks. Since the structure of RNN is naturally suitable to model time-series data, the Seq2Seq model can capture semantic and syntactic relations between posts and comments. Besides, the attention mechanism[1, 6] was proposed to enhance the model on learning patterns from data. In this paper, we use Seq2Seq as our base model and add local and global keywords which we excerpt from our retrieval-based system as extra information to build our novel ensemble conversation system. The response is generated in a two-step fashion: First, we predict 3 global keywords and 3 local keywords. The global keywords are chosen according to their the point-wise mutual information (PMI)[7]. The PMI of each word is calculated over the whole repository, we therefore denoted this kind of keywords as global keywords. On the other hand, we introduce TF-IDF to choose local keywords because the TF-IDF is calculated on the post-comment pairs related to the input post. We then use a modified Seq2Seq model to synthesize a response based on the input, global keywords and local keywords.

Our contribution is two-fold: 1) build a retrieval-based system based on similarity, 2) propose an advanced response generation system based on Seq2Seq model combined with local and global keywords. And the remainder of this paper is organized as follows. Section 2 discusses our system architecture in details. Section 3 describes our experiment and analysis on the experimental results. Finally, we conclude our paper in section 4.

2. SYSTEM ARCHITECTURE

As we submit results by both retrieval-based and generation-

[1]To make it clearly, we use STC-1 to denote the STC task in NTCIR-12 and STC-2 for NTCIR-13

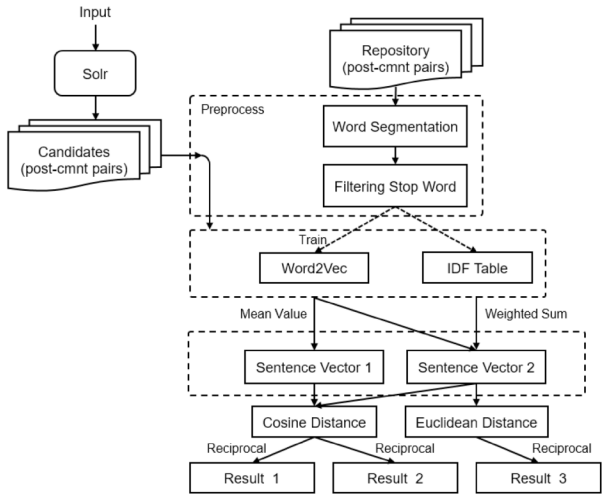

Figure 1: The structure of retrieval-based system

based approaches, we introduce them separately in the following subsections.

2.1 Retrieval-based System

Our retrieval-based system consists of preprocessing, similarity calculation and ranking. The architecture is illustrated in Figure 1.

The first module in our retrieval-based system is preprocessing. Unlike English sentences which are separated by spaces naturally, Chinese texts are written continuously without any characters between word phrases. Therefore the first thing we need to do is segmenting the sentences into various word phrases. After this step, we filter out the stop words which are some auxiliaries and do not contain any substantive content.

Based on the materials after preprocessing, we use Google Word2Vec[2] to obtain the continuous distributed representations of words. Since Google Word2Vec has been proved to be an effective and efficient tool to obtain real-value vector for words, and it consider the synonyms in the same position, we use these word vectors to calculate the representation of sentence. In information retrieval, the inverse document frequency (IDF) measure the importance of each document and here we introduce IDF as the weight for each word when constructing the sentence vector. The IDF of the i-th word is:

$$IDF_i = log\frac{|D|}{|\{j : t_i \in d_j\}|}$$

where $|D|$ is the total number of documents and the denominator is the number of documents which contain the i-th word. We use the large repository contains all the post-comment pairs as the training dataset to obtain the word vectors and IDF table.

When a user-issued utterance is input into the system, it will be first used as a query to Solr which build on the whole repository to obtain a number of candidate post-comment pairs. Then these pairs are preprocessed into a list of word phrases without stop words. Then, we gain the sentence vector of the input post and the candidate pairs by two ways: simply mean value and weighted sum value of each word vector in the sentence. Assume a sentence s contains n word phrases $(w_1, \cdots, w_n)$:

$$Vector\_Mean(s) = \frac{\sum_{i=1}^{n} f(w_i)}{n}$$

$$Vector\_WeightedSum(s) = \sum_{i=1}^{n} IDF_i * f(w_i)$$

where $f(w_i)$ is the vector of i-th word. The reciprocal of distance is used as the similarity between the input post and the candidate posts (post-posts similarity) and the input post and the candidate comments (post-comments similarity). We use Cosine and Euclidean function to calculate the distance of two vectors.

$$Cosine(\overrightarrow{v_1}, \overrightarrow{v_2}) = \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \cdot \|\overrightarrow{v_2}\|}$$

$$Euclidean(\overrightarrow{v_1}, \overrightarrow{v_2}) = \frac{1}{\sqrt{(\overrightarrow{v_1} - \overrightarrow{v_2})(\overrightarrow{v_1} - \overrightarrow{v_2})^T}}$$

Since we consider that the sentence vector obtained by weighted sum may have more potential, we calculate the similarity under this kind of representation by both Cosine function and Euclidean function. On the contrary, the sentence vectors constructed by mean value is used only once for Cosine distance. The post-posts similarity and post-comments similarity are combined linearly:

$$similarity(p, candidates) = 0.75 * post\text{-}posts\_similarity$$

$$+0.25 * post\text{-}comments\_similarity$$

Finally we get three different kinds of similarity which measure the relevance between the input post and candidate post-comment pairs.

The result is an ordered list of candidates reranking by the similarity scores.

2.2 Generation-based System

Figure 2 depicts the overall architecture of our generation-based approach, which comprises two main steps:

Step 1: use PMI and TF-IDF to predict 3 global keywords and 3 local keywords for the reply separately.

Step 2: generate a reply conditioned on the keywords as well as the input post based on a Seq2Seq model with attention mechanism.

2.2.1 PMI and global keywords

PMI indicates the relevance between two words through considering the co-occurrence of them. Here we compute the PMI of a query word w_q and a reply word w_r using a large corpus by:

$$PMI(w_q, w_r) = log\frac{p(w_q, w_r)}{p(w_q)p(w_r)} = log\frac{p(w_q|w_r)}{p(w_q)}$$

Then, we define the PMI of a reply word w_r and a query sentence $q = (w_{q_1}, w_{q_2}, \cdots, w_{q_n})$ as follows:

$$PMI(q, w_r) = PMI(w_{q_1}, w_{q_2}, \cdots, w_{q_n}, w_r)$$

$$= log\frac{p(w_{q_1}, w_{q_2}, \cdots, w_{q_n}|w_r)}{p(w_{q_1}, w_{q_2}, \cdots, w_{q_n})} \approx log\frac{\prod_{i=1}^{n} p(w_{q_i}|w_r)}{\prod_{i=1}^{n} p(w_{q_i})}$$

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

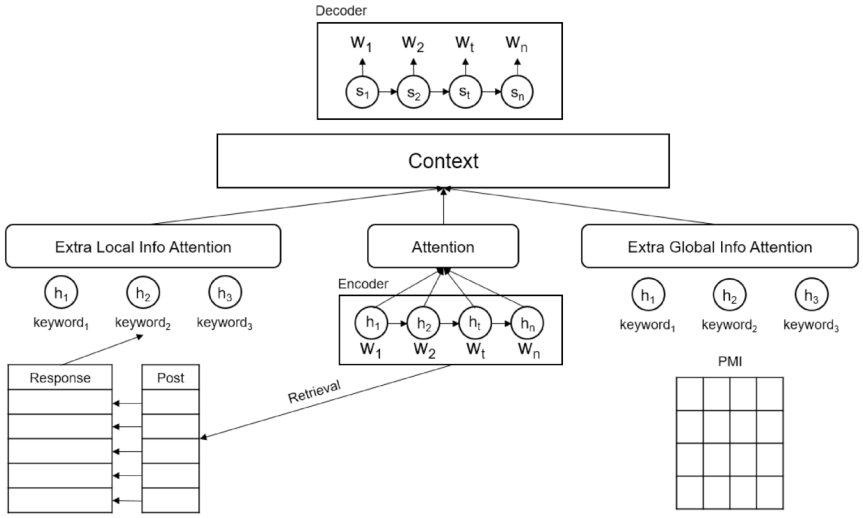

Figure 2: The structure of generation-based system

$$= \sum_{i=1}^{n} log \frac{p(w_{q_i}|w_r)}{p(w_{q_i})} = \sum_{i=1}^{n} PMI(w_{q_i}, w_r)$$

The approximation is due to the independence assumptions of both the prior distribution $p(w_{q_i})$ and posterior distribution $p(w_{q_i}|w_r)$.

Unlike simply choosing the word with highest occurrence, PMI penalizes a common word by dividing its prior probability, hence, PMI prefers a word that is most "mutually informative" with the query. Moreover, the PMI is computed over the whole corpus thus we denote the words chosen by this way as global keywords.

In our system, we use 2500 nonus as candidate keywords. And compute the PMI of each word with the top 40000 frequent words in the post to obtain a PMI matrix. When we input a query into the system, it will check the PMI of every candidate keyword with the words in the query and return the top 3 as the global keywords.

2.2.2 TF-IDF and local keywords

On the other hand, we also consider the most important words in the user-issued utterance. TF-IDF is involved to measure the importance of each word in one input post. IDF is described in the former section, and here TF-IDF is computed by:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$TF\text{-}IDF_{i,j} = TF_{i,j} * IDF_i$$

where $n_{i,j}$ is the occurrence times of w_i in document d_j which contains k words.

The words we obtain by this measure may contain more information and reflect the most important things in the user input post. And TF-IDF of each word in different document is different accordingly. We therefore use these words as local keywords.

In our system, we first use the input post as the query to Solr and get 20 candidate comments. Then, we compute the TF-IDF of every word in the candidate comments and choose the top 3 as the local keywords.

2.2.3 Seq2Seq model with attention mechanism

The Seq2Seq model is one of the RNN-based model which suited to solve tasks in symmetrical structure like machine translation, speech recognition, etc. All of these problems are in similar form, for example, machine translation transforms one language to another language and speech recognition transforms text to speech. Intuitively, response generation belongs to this kind of tasks that it transforms posts to comments. Both of the two sides are around the same topic abstractly.

The Seq2seq model comprises two RNNs which denoted as encoder and decoder. The input post is encoded into a fixed-size vector by the encoder and then the decoder generates reply from the vector. Formally, given a source sequence (post) $x = (x_1, x_2, \cdots, x_n)$ and a target sequence (comment) $y = (y_1, y_2, \cdots, y_{n'})$, the model maximizes the generation probability of y conditioned on x: $p(y_1, \cdots, y_{n'}|x_1, \cdots, x_n)$. We denote the vector output by decoder as context vector c. And the objective function of Seq2Seq model can be written as:

$$p(y_1, \cdots, y_{n'}|x_1, \cdots, x_n) = p(y_1|c) \prod_{t=2}^{n'} p(y_t|c, y_1, \cdots, y_{t-1})$$

and the context vector c is calculated by the RNN:

$$h_t = f(x_t, h_{t-1}); c = h_T$$

where h_t is the hidden state at time t and f is a non-linear transformation which can be either an long-short term memory unit (LSTM)[4] or a gated recurrent unit (GRU)[2, 3]. In this work, we use GRU which is parameterized as

$$z = \sigma(W^z x_t + U^z h_{t-1})$$

$$r = \sigma(W^r x_t + U^r h_{t-1})$$

$$s = tanh(W^s x_t + U^s(h_{t-1} \circ r))$$

$$h_t = (1 - z) \circ s + z \circ h_{t-1}$$

The decoder is a standard RNN language model except conditioned on the context vector c. The probability distribu-

tion p_t of candidate words at every time t is computed as:

$$s_t = f(y_{t-1}, s_{t-1}, c); p_t = softmax(s_t, y_{t-1})$$

where s_t is the hidden state of the decoder RNN at time t and y_{t-1} is the word at time $t-1$ in the reply sequence.

The attention mechanism was proposed to enhance the model on learning patterns from data[]. In Seq2Seq with attention, when decoder generates a word, it not simply depends on the final state of encoder (the context vector c) as we described before. On the contrary, each word y_i corresponds a context vector c_i, and c_i is a weighted sum of all hidden states $(h_1, \cdots, h_n)$ of the encoder. Formally, c_i is defined as:

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j$$

where α_{ij} is given by:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})}; e_{ij} = \eta(s_{i-1}, h_j)$$

η is usually implemented as a multi-layer perceptron (MLP) with $tanh$ as the activation function.

In our methods, we have two kinds of keywords as extra information, and we therefore introduce two extra attentions to involve them. As it shows in Figure 2, we use an "extra local info attention" and an "extra global info attention" which function on local keywords and global keywords separately. Then we combine them with the encoder hidden states by a joint attention to construct the context vector. As for the decoder, it runs like the traditional Seq2Seq model with attention mechanism. Besides, beam search is employed to generate multiple sentences.

2.2.4 Reranking

After these aforementioned steps, we gain a list of replies. This result is one of our submitted runs. Moreover, we design some exquisite functions to rerank the replies.

The first one is the Cosine similarity between the generated reply and corresponding post. We have already described the Cosine similarity in the former section. And the replies are raranked by the similarity score.

Another one is that we put these responses into Solr again and get the top 1 post. And use the same Cosine function to measure the similarity between these posts and the input post. Again, rerank replies according to these scores.

3. EXPERIMENT AND ANALYSIS

In this section, we will introduce the dataset in this task, the evaluation metrics and the implementation details in our different runs.

3.1 Dataset

Table 1 shows the statistics of the repository and the small set of data with label.

The repository contains 219,174 Weibo posts and corresponding 4,305,706 comments which comprise 4,433,949 pairs. Therefore each post have more than 20 different comments on average, and one comment may be used to respond to multiple different posts.

Besides, this task provides another small set of data with labels. They are 768 posts and each of them correspond to 15 comments, thus the total number of pairs is 11,535.

Table 1: The Dataset of Sina Weibo

Repository	#posts	219,174
	#comments	4,305,706
	#original pairs	4,433,949
Labeled Data	#posts	768
	#comments	11,535
	#labeled pairs	11,535

These pairs are labeled with "suitable(+2)", "neural(+1)" and "unsuitable(0)". The label "suitable" means the comment matches clearly with the post, while "neural" indicates the comment can be a reply to the post in some specific scenarios and "unsuitable" means they are unrelated.

3.2 Evaluation Metrics

The evaluation metrics are nG@1, $P+$, and $nERR$@10 which following the NTCIR-12 STC Chinese subtask[].

nG@1 shows the quantity of effective result (like result with label "suitable" or "neural") in the results. It will take three values: 1, 1/3 or 0 in this task.

$P+$ reflects the position of the best result in the ranking list of candidates.

$nERR$@10 considers the correctness of the ranking, which means the more effective result should be ranked more top in the candidates.

Table 2: Part of Official Results (Top 90 only)

Run	nG @1	Run	P+	Run	nERR @10
SG01-C-G1	0.5867	SG01-C-G1	0.6670	SG01-C-G1	0.7095
splab-C-G4	0.5080	splab-C-G4	0.6080	SG01-C-G1	0.7095
Beihang-C-R4	0.4980	Beihang-C-R4	0.5818	splab-C-G4	0.6492
Nders-C-R4	0.4780	DeepIntell-C-R1	0.5564	Beihang-C-R4	0.6105
srcb-C-R5	0.4500	Nders-C-R2	0.5497	DeepIntell-C-R1	0.5994
rucir-C-R2	0.3453	rucir-C-R2	0.4675	rucir-C-R2	0.5064
rucir-C-R3	0.2617	rucir-C-R3	0.3924	rucir-C-R3	0.4272
rucir-C-R1	0.2567	rucir-C-R1	0.3802	rucir-C-R1	0.4079
rucir-C-G3	0.2543	rucir-C-G3	0.3461	rucir-C-G3	0.3828
		rucir-C-G1	0.1712		

3.3 Implementation Details

We use Jieba[3] to split the posts and comments into sequences of words. Albeit both posts and comments are Chinese short text collected from Weibo, the distribution of the words are quite different. The number of unique words in the posts is 131,658 while that of responses is 574,916. We build two separate dictionary for posts and responses

[3]https://github.com/fxsjy/jieba

by using 40,000 most frequent words on each side, covering 97.01% usage of words for post and 95.65% for reply respectively. The words which not in the dictionary are replaced by a special token "_UNK". The dimensions of the hidden state of encoder and decoder are both 1,000 and the dimensions of the word-embedding for post an d response are both 620. All the model parameters are initialized by random sampling from a uniform distribution in $[-0.1, 0.1]$. All our models were trained on a NVIDIA Tesla K40 GPU using stochastic gradient descent algorithm with mini-batch.

3.4 Experimental Results and Analysis

The best 5 teams with their best run results and our results are shown in Table 2. They are sorted by evaluation metrics respectively.

Following the results, our methods can provide meaningful responses to some extent. However, the performance is not really good compared with other teams. On the other hand, we found that the "rucir-C-G3" model outperforms our other generation models. This is the model without reranking module. We guess that our reranking method is too naive and not suitable for the generation results. Besides, the results generated by the model with only one kind of keywords are even worse and do not appear at the top 90 list. We can infer that the two different keywords may both affect and improve the Seq2Seq model.

We also do some vertical analysis on each team's results. We count the number of team whose best model is either generation-based system or retrieval-based system. The result is interesting, under the mean $nG@1$ metrics, only 5 teams' generation-based models outperform their retrieval-based models while 12 teams on the contrary. We conclude that although the retrieval-based approaches could not return totally new response, they guarantee the fluency and grammar correctness at least. However, the best one among all the models is generation-based. It proves that generation models contain numerous potential to be explored.

4. CONCLUSION

In this paper, we propose two different kinds of approaches based on retrieval and generation for STC-2 task of NTCIR-13. We use the similarity as features to rank the retrieval results. And introduce an advanced Seq2Seq model with local and global keywords to generate responses. The experimental results show the effectiveness of our methods.

In the future, we will modify the reranking module for our generation-based system, and adjust the encoder to involve more information for generating process.

5. REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Computer Science*, 2013.

[2] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*, 2014.

[3] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Eprint Arxiv*, 2014.

[4] A. Graves. *Long Short-Term Memory*. Springer Berlin Heidelberg, 2012.

[5] Z. Ji, Z. Lu, and H. Li. An information retrieval approach to short text conversation. In *Computer Science*, 2014.

[6] M. T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Computer Science*, 2015.

[7] L. Mou, Y. Song, R. Yan, G. Li, L. Zhang, and Z. Jin. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*, 2016.

[8] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *ACL*, 2015.

[9] L. Shang, T. Sakai, H. Li, R. Higashinaka, Y. Miyao, Y. Arase, and M. Nomoto. Overview of the ntcir-13 short text conversation task. In *The NTCIR-13 Conference*, 2017.

[10] H. Wang, Z. Lu, H. Li, and E. Chen. A dataset for research on short-text conversations. In *EMNLP*, pages 935–945, 2013.