TUA1 at NTCIR-13 Short Text Conversation 2 Task

Yunong Wu
Faculty of Engineering
Tokushima University
Tokushima, Japan
wuyunong@tokushima.ac.jp

Xin Kang, Kenji Kita, Fuji Ren
Faculty of Engineering
Tokushima University
Tokushima, Japan
{kang-xin, kita, ren}@is.tokushima-u.ac.jp

ABSTRACT
In this paper, we describe the overview of our work in Short Text Conversation 2 task at NTCIR-13. We propose two different methods including retrieval-based method and generation-based method. Our retrieval-based method contains index part and re-ranking part. Rep-post is used as query to search comments from rep-cmnt, and indexed candidate comments are re-ranked by three models respectively. Our generation-based method constructs a sequence-to-sequence neural network model with attention mechanism, to sequentially read a post sentence word by word, calculate an attention weight over the input words, and output a comment sentence with the normal search and the beam search strategies. We propose an RNN model to reorder the generated comment sentences from 26 parallel sequence-to-sequence models by evaluating the fitness between post-comment pairs, and employ a cosine similarity between the post-comment pair to assist the reordering. The evaluation over our groups of Formal Run submissions results suggest that our method is effective for re-ranking and generating a list of meaningful comment sentences for short text conversation.

Keywords
Elasticsearch, LSI, RNN, GRU

Team Name
• TUA1

Subtasks
• Short Text Conversation 2 (Chinese)

1. INTRODUCTION
Short Text Conversation 2 (STC2) is a task [1] organized by NTCIR-13 which focuses on comment retrieval from a large repository of post-comment pairs from weibo in Chinese. In this year, STC2 task considers two methods from different aspects including retrieval-based method which was proposed in the first year, and generation-based method to generated the new comments. In this task, we submit 4 runs with retrieval-based method and 5 runs with generation-based method respectively. In retrieval-based method, we build the index part by open source named elasticsearch and re-ranking part composed by three different machine learning models: Latent Semantic Indexing (LSI) [2], Paragraph Vector [3] and Recurrent Neural Network (RNN) [4]. In generation part, we build a sequence-to-sequence neural network [5] model with the attention mechanism [6] to encode a post sentence into a sequence of 128-dimensional vectors, and decode them into a sequence of comment words with a weighted attention.

2. RETRIEVAL-BASED METHOD
Two parts including index and re-ranking are proposed in this retrieval-based method. Figure 1 demonstrate the whole architecture of this method.

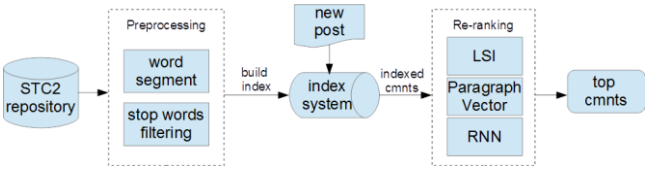

Figure 1. Architecture of retrieval-based method.

2.1 Data Preprocessing
The STC2 repository contains 4 million rep-post sentences and 4 million rep-cmnt sentences in pairs. Because all the STC2 rep-post and rep-cmnt sentences are raw sentences from Sina-weibo, we have to do careful preprocessing to these sentences, in order to retain the original information. We segment the sentences by using the Jieba segmentation algorithm. We count the word frequency and remove words with the highest and the lowest word frequencies. Stop words are also filtered out.

2.2 Building Index
We build the index part by Elasticsearch[1]. All the rep-post and rep-cmnt sentences are preprocessed, and imported to the index model. We use the sentence of testing set as query to search the rep-cmnt sentences, only top 30 candidate sentences with high scores are returned.

2.3 Re-ranking models
Three different re-ranking models are used in this part, including Latent Semantic Indexing, Paragraph Vector and Recurrent Neural Network.

2.3.1 Latent Semantic Indexing model
We have an intuition that the query and comment sentences share same words or words of same meanings, we try to find the semantic relations between them. Instead of word matrix, we construct the tfidf matrix by computing the tfidf of each word using equation 1:

$$C(tfidf) = \begin{bmatrix} [(w_{11}, tfidf_{11}),...,(w_{1j}, tfidf_{1j})] \\ \dots \\ [(w_{i1}, tfidf_{i1}),...,(w_{ij}, tfidf_{ij})] \end{bmatrix} \quad (1)$$

[1] Elasticsearch is an open source software which can search data quickly and easily. It can be accessible at https://www.elastic.co

where, (w_{ij}, $tfidf_{ij}$) indicates tfidf value of the i_{th} word of j_{th} sentence calculated from the STC2 repository.

We derive the LSI space representation using Singular Value Decomposition with matrix C(tfidf) by equation 2:

$$\hat{C} = W_{I\times k}\Sigma_{k\times k}\left(S_{J\times k}\right)^T \quad (2)$$

where, I is the number of distinct words, k denotes dimension of LSI space, and J is the number of sentences. W and S represent word matrix and sentence matrix respectively, Σ is diagonal matrix containing singular value in descending order.

We map the query and indexed candidate comments from index part into the LSI space to obtain the corresponding vector and compute the cosine similarity between them using equation (3):

$$sim(q, c_i) = \frac{\hat{q}\cdot\hat{c}_i}{\|q\|\|c_i\|} \quad (3)$$

2.3.2 *Paragraph Vector model*

We think that LSI is good application of vector representation, but it constructs matrix without word order so that it loses the relevance information. We propose the paragraph vector which can consider more information carried by the sentence.

Given a sequence of training words (w_1, w_2, .., w_T), the paragraph vector model learns the word vector representations and generate a log probability of the missing word w_t given surrounding words as

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p\left(w_t \mid w_{t-k},...,w_{t+k}\right) \quad (4)$$

We predict the word using the multi-class classifier like softmax by equation (5)

$$p\left(w_t \mid w_{t-k},...,w_{t+k}\right) = \frac{\exp\left(y_{w_t}\right)}{\sum_i \exp\left(y_i\right)} \quad (5)$$

where, y_i is un-normalized log probability of each output word i, calculated by equation (6)

$$y = b + Uh\left(w_{i-k},...,w_{i+k};W,S\right) \quad (6)$$

in which, U and b indicates softmax parameters, h is constructed by the average word vectors extracted from word matrix W and paragraph matrix S.

2.3.3 *Recurrent Neural Network model*

The neural network model[7] takes a pair of post sentence and comment sentence as input and generate the probabilities (p_0, p_1, p_2) for 3 fitness levels in y, i.e. $y \in (0, 1, 2)$ like Figure 2.

Specifically, each word in the post sentence wpi is embedded into a 128-dimensional vector, and the vector sequence from the post sentence is fed into a GRU network to generate a neural summarization of the post. The same procedure is performed on the comment sentence, which generates a neural summarization of the comment, with shared neural network parameters. The post summarization and the comment summarization are then concatenated into a long vector with Dropout, and then fed into a fully-connected layer to predict their fitness levels (p0, p1, p2) with a softmax activation function. We use the expected value of the fitness level E(y) to evaluate the fitness between a post and a comment, which is calculated by equation (7) as follows:

$$E(y) = \sum_{y=0}^{2} y \times p_y \quad (7)$$

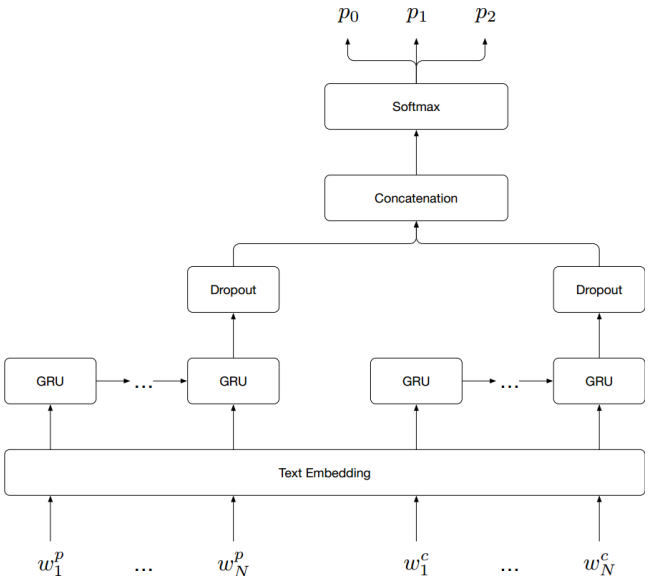

Figure 2. Recurrent Neural Network Model.

2.4 Submitted Runs

For training, the STC2 Dry Run set provides 786 rep-post sentences as the query sentences, and 11536 rep-cmnt sentences with manually labeled relative scores as the reply sentences. In testing set, 100 sentences extracted from rep-post are used to examine the performance of three different models in our system.

We submitted four runs by three different models and one without re-ranking considered as the benchmark in this task.

TUA1-C-R1: index + Paragraph Vector

TUA1-C-R2: index

TUA1-C-R3: index + RNN

TUA1-C-R4: index + LSI

Table 1. Official Results of TUA1 Runs By Retrieval-Based Method

	Mean nG@1	**Mean nERR@10**	**P+**
TUA1-C-R1	0.2653	0.4479	0.406
TUA1-C-R2	0.3697	0.5298	0.4913
TUA1-C-R3	0.3177	0.4963	0.4662
TUA1-C-R4	**0.421**	**0.5524**	**0.4952**

Table 1 shows the official result of our runs. Run4 with LSI models get the best performance in three evaluation methods. Run1 with paragraph vector get the worst performance, while Run 2 and Run3 get the median scores.

3. GENERATION-BASED METHOD

3.1 Data Preparation

We employ the post-comment pairs in the STC2 repository to train a sequence-to-sequence neural network model with attention mechanism, and employ post-comment pairs with "L2" labels in the STC2 Dry Run training set for evaluating the training process. For each sentence, we simply split it into a list of words, with all capital words transformed to the lower case. To keep the model simple, we also restrict the vocabulary for the most common 10,000 words.

3.2 Model Construction

We build a sequence-to-sequence neural network with attention mechanism to encode a post sentence $x_1, x_2, \cdots, x_T$into a sequence of hidden vectors $\bar{h}_1, \bar{h}_2, \cdots, \bar{h}_T$, and decode them into a sequence of comment words $y_1, y_2, \cdots, y_{T'}$,with a weighted attention a. The detailed model is depicted in Fig. 3.

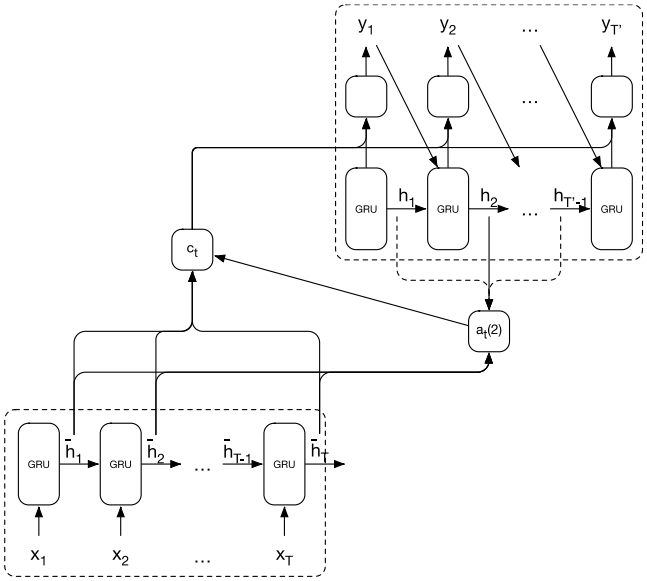

Figure 3. Sequence-to-Sequence network with attention.

In Fig. 3, we firstly encode each word x_t in a post sentence into a 128-dimensional hidden vector $\bar{h}_t$ with a GRU network [7]. The GRU network could learn to remember the previous information in the post sentence in its hidden vector $\bar{h}_{t-1}$ at position $t-1$and integrate it with the new post word x_t to generate a new hidden vector $\bar{h}_t$ as its output. The encoding GRU takes in a post sentence $x_1, x_2, \cdots, x_T$ and generates a sequence of encoding vectors $\bar{h}_1, \bar{h}_2, \cdots, \bar{h}_T$.

Then we decode the sequence of hidden vectors $\bar{h}_1, \bar{h}_2, \cdots, \bar{h}_T$ into a vector of comment words in $y_1, y_2, \cdots, y_{T'}$, with another GRU network. The decoding GRU network also leans to remember the previous information of its output in h_{t-1} and integrate it with its previous output y_{t-1} to generate a 128-dimensional vector h_t as the output. This new output h_{t-1} is combined with an attentional input vector c_t as

$$c_t = \sum_{s=1}^{T} a_t(s)\bar{h}_s,$$

which is the weighted sum of encoding vector sequence $\bar{h}_1, \bar{h}_2, \cdots, \bar{h}_T$ with attention a_t, which is given by

$$a_t(s) = \frac{exp(h_t \cdot \bar{h}_s)}{\sum_{s=1}^{T} exp(h_t \cdot \bar{h}_s)},$$

and fed into a fully-connected layer with softmax activation to predict the output word y_t

$$y_t = \text{softmax}(W_y \times [c_t, h_t]).$$

A beam-search is performed on the comment word prediction, to generate comment sentences with a large joint probability.

To reduce overfitting, we add Dropout layers on the outputs of the encoder GRU network and the decoder GRU network, with a probability of 0.2 for randomly replacing an output unit to 0. By looking into the repository data, we specify the maximum number of words in the post sentences and the comment sentences to 50, in order to keep the model simple while fit most of the input and output sentences.

3.3 Submitted Runs

We submit 5 groups of Formal Run results, which are generated by 26 parallel the sequence-to-sequence networks with different random initializations. The submitted runs are generated by different searching and ranking specifications as follows:

TUA1-C-G1: normal search, RNN ranking

TUA1-C-G2: beam search, RNN ranking

TUA1-C-G3: normal search + beam search, RNN ranking

TUA1-C-G4: normal search, RNN + cosine ranking

TUA1-C-G5: beam search, RNN + cosine ranking

In the above, normal search indicates that each comment word is generated by maximum a posteriori estimation, and beam search indicates that the comment sentence as a whole is generated by maximum a posteriori estimation. RNN ranking corresponds to our recurrent neural network ranking approach as illustrated in section 2.3.3, and the cosine ranking corresponds to the cosine similarity between the post and comment sentences with respect to the tf-idf features.

Table 2 shows our results with the generation-based method. By comparing the results, we find that beam search does not render promising results as the normal search, which is probability because our beam width (5) is not large enough to consider the joint word probability in longer sentences. We also find that the cosine ranking helps reordering the comment sentences. This implies that given a post sentence, the generated comment sentence with a similar word distribution as the post sentence, i.e. repeating the post sentence in some extent, could be more preferable than those with very different word distributions.

Table 2. Official Results of TUA1 Runs By Generation-Based Method

	Mean nG@1	**Mean nERR@10**	**P+**
TUA1-C-G1	0.3440	0.4339	0.4882
TUA1-C-G2	0.2553	0.3731	0.4313
TUA1-C-G3	0.3276	0.4232	0.4711
TUA1-C-G4	**0.3994**	**0.4914**	**0.5395**
TUA1-C-G5	0.3110	0.4253	0.4809

4. CONCLUSION

In this paper, we report our work on the Short Text Conversation 2 (Chinese) task at NTCIR-13. We propose two different methods including retrieval-based method and generation-based method. Our retrieval-based method contains index part and re-ranking part. In index part, we use the rep-post as query to search comments from rep-post, and in re-ranking part, we utilize LSI, paragraph vector, RNN respectively to construct re-ranking models and use the models to re-rank the indexed candidate comments. Our generation-based method constructs a sequence-to-sequence neural network with attention mechanism, which takes the post sentence as the input sequence and generates the comment sentence as the output sequence. The attention mechanism allows our method to calculate a weighted attention covering every input word in the encoder network and to feed proper information to the decoder network for generating each output word. The retrieval-based method and generated-based method render 4 and 5 groups of Formal Run submissions respectively, with different searching and ranking strategies. We find that in retrieval-based method LSI used re-ranking models obtain the best scores, and in generation-based method the output comment sentences under the normal search render better results than those under the beam search, and that a cosine similarity with the RNN fitness over the candidate comment sentences score could improve result ranking.

Our future work will focus on the development of better neural language models for sentence generation.

5. ACKNOWLEDGMENTS

This research was partially supported by JSPS KAKENHI Grant Number 15K00425.

6. REFERENCES

[1] Lifeng Shang, Tetsuya Sakai, Hang Li, Ryuichiro Higashinaka, Yusuke Miyao, Yuki Arase and Masako Nomoto: "Overview of the {NTCIR}-13 Short Text Conversation Task", Proceedings of NTCIR-13, 2017

[2] Deerwester S, Dumais S T, Furnas G W, et al. "Indexing by latent semantic analysis[J] ". Journal of the American society for information science, 1990, 41(6): 391.

[3] Le Q, Mikolov T. "Distributed representations of sentences and documents[C] "//Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014: 1188-1196.

[4] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, September). "Recurrent neural network based language model". In Interspeech (Vol. 2, p. 3).

[5] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[6] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[7] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).