Beihang at the NTCIR-13 STC-2 Task

Dejian Yang
Beihang University
dejianyang@buaa.edu.cn

Yu Wu
Beihang University
wuyu@buaa.edu.cn

Zhoujun Li
Beihang University
lizj@buaa.edu.cn

Wei Wu
Microsoft Research Asia
wuwei@microsoft.com

Can Xu
Microsoft Research Asia
can.xu@microsoft.com

ABSTRACT

This paper presents our system participated in the NTCIR-13 Short Text Conversation Task. Previous researches focus on how to rerank response candidates but pay little attention on how to generate high quality candidates. In this work, we try different approaches for the retrieval part, such as neural network features and symbolic features. The evaluation results show that symbolic features can significantly boost the performance of the retrieval-based chatbot. According to the official STC evaluation, we are in the second best group among teams who participated in the Chinese retrieval-based chatbots.

Team Name

Beihang

Subtasks

Short Text Conversation (Chinese, Retrieval)

Keywords

retrieval-based chatbots, deep learning, symbolic knowledge

1. INTRODUCTION

Beihang team participates in the Chinese retrieval-based Short Text Conversation (STC) task at NTCIR-13 [10]. This report describes our approach on response selection and discusses the experimental results.

STC, a simple version of conversation, is defined as a one round conversation comprising two short texts, where the former one is the message given by a human and the latter one is a response given by a computer. There are two dominated approaches for this task, including the retrieval-based approach [12, 13] and the generation based approach [11, 14, 6]. Generation-based methods generate responses with natural language generation models learnt from the conversation data, while retrieval-based methods re-use the existing responses by selecting proper ones from an index of the conversation data. In this report, we show our preliminary results on the Chinese retrieval-based chatbots [10].

The architecture of a retrieval-based chatbot [4] is shown in the Figure 1. Given a message, it is normalized into a clean text with pre-processing algorithms. After that, several response candidates are retrieved from a pre-defined index. Finally, matching models are employed to compute similarities between the responses and the message. Previous research focus on the matching models but pay little

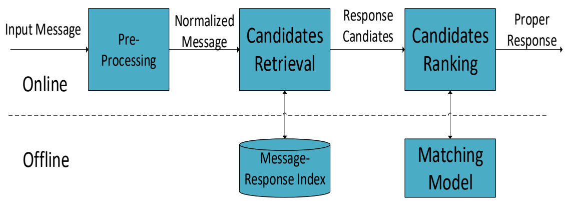

Figure 1: The architecture of a retrieval-based chatbot.

attention on the candidate retrieval. Candidate retrieval, however, is a crucial step for a chatbot, because an appropriate response have no chance to be selected if it cannot be retrieved back in this step. Current work uses keyword based approaches to retrieve candidates from the index, and keyword based retrieval tools such as Lucene [1] and Elasticsearch [2] are integrated into their systems. However, some appropriate responses cannot be retrieved back with the methods, because they do not share many common words with the given message. To mitigate this issue, we propose two alternative retrieval approaches and give their preliminary results on the Chinese STC data set.

The first approach is that we use sentence embedding and approximated k nearest neighbor algorithms to directly find semantically related candidates from the index. The second approach is that we use symbolic knowledge to help the keyword based retrieval, in which important entities are taken into account. After the candidate retrieval, we train a matching model with several similarity features, including: Tf-idf cosine, LCS (Longest Common Substring), Word overlap, Tree kernel, Language model, Translation probability, Word Embedding cosine, GRU (Gated Recurrent Units) and CNNs (Convolution Neural Networks). All the features are combined as a ranking model by a gradient boosted regression tree which is implemented by Xgboost [3].

2. SYSTEM DESCRIPTION

In this section, we elaborate our system in details.

2.1 Pre-Processing

[1]https://lucene.apache.org/
[2]https://www.elastic.co/
[3]https://github.com/dmlc/xgboost

We choose a popular Chinese word segmentation tool named jieba [4] to segment a Chinese sentence into several words. We used a dictionary based named entity recognition algorithm to extract entities in a text. The dictionary consists of Baidu Baike entries [5], which have more than 1 Chinese characters. To filter noise in the dictionary, we remove entires whose inverse document frequency (IDF) are below 5.

2.2 Candidate Retrieval

Suppose that the entire data set is defined as $\mathcal{D} = \{p_i, c_i\}_{i=1}^{N}$, where p_i is a post, and c_i is its associated comment. Given a query q, the goal of the candidate retrieval component is to form a subset $\mathcal{D}' \in \mathcal{D}$, in order to reduce the time cost of the candidate ranking. Here we use three approaches to form $\mathcal{D}'$. We randomly sample a c_i^- for each of post p_i for the further parameter learning.

2.2.1 Keyword based approach

The basic idea of candidate comment retrieval is to use the invert index to obtain comments that share some words with the input q. Since the number of retrieved comments is too large to apply complex rerank algorithms, top 30 comments are preserved with the Okapi BM25 [7] algorithm that is formulated as

$$BM25(q, c_i) = \sum_{j=1}^{M} IDF(w_j) \frac{f(w_j, c_i) \cdot (k+1)}{f(w_j, c_i) + k \cdot (1 - b + b \cdot \frac{|q|}{avgcl})}, \tag{1}$$

where $f(w_j, c_i)$ is the j-th word frequency in the comment c_i, $|q|$ is the length of the query q in words, and $avgcl$ is the average comment length in the index. $k = 1.5$ and $b = 0.75$ in our implementation and $IDF(w_j)$ is the inverse document frequency weight [8] of the term w_j.

In the following section, we denote the approach as R_{word}.

2.2.2 Sentence embedding based approach

As the invert index may ignore comments that do not share same terms with the input query, we use a embedding based method to form $\mathcal{D}'$. At first we train a recurrent neural network with gated units (GRU) [3] to compute the sentence representation vector of a post or a comment. The parameters of GRU is optimized by maximizing

$$\sum_{i=1}^{N} \left[Cos\Big(\mathbf{GRU}(p_i), \mathbf{GRU}(c_i)\Big) - Cos\Big(\mathbf{GRU}(p_i), \mathbf{GRU}(c_i^-)\Big) \right] \tag{2}$$

where c_i^- is a randomly negative sampled comment for p_i, and Cos is the cosine similarity function.

By this mean, each text can be represented as a dense vector by $\mathbf{GRU}(\cdot)$, and the Euclidean distance of two vectors describes their gap in the semantic space. Given a new input query and its sentence embedding, we find its neighbors in the vector space as candidates. We adopt the Annoy [6], a C++ library with Python bindings, to do the approximate nearest neighbor search for the efficiency. Note that we can also compute the cosine distance between the input query and a comment one by one, but it increases the system overload significantly. The sentence embedding based approach for retrieval is denoted as R_{embed}.

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

[5]https://baike.baidu.com/

[6]https://github.com/spotify/annoy

2.2.3 Symbolic knowledge enhanced approach

We leverage symbolic knowledge, named entity information, to enhance the key word based retrieval. We modify Equation 1 that only uses the frequency information to compute the score, and propose a new ranking function

$$\sum_{j=1}^{M} IDF(w_j) \frac{f(w_j, c_i) \cdot (k+1)}{f(w_j, c_i) + k \cdot (1 - b + b \cdot \frac{|q|}{avgcl})} + \lambda \sum_{l=1}^{O} \frac{f(e_l, c_i)}{|c_i|}, \tag{3}$$

where e_l represents an entity in the query q, $|c_i|$ is the comment length, λ is a trade-off between entity matching score and BM25 score which is chosen as 5 in our system. We denote this approach as R_{sym}.

We can obtain 30 candidate comments by these approaches respectively in our different submissions.

2.3 Candidate Ranking

Candidate ranking targets at reranking comments $C = \{c_i | (p_i, c_i) \in \mathcal{D}'\}$ according to their similarities with the input query q. To this end, we first extract similarity features between query and comments and then combine these features to predict the final similarity score.

2.3.1 Similarity Features

We propose several features to compute the similarity.

Tf-idf cosine: The query q and comments C are converted to one hot representations weighted by tf-idf values, where tf is the term frequency in the text, and idf is calculated using the entire data in the index. The cosine of representations is used as a feature.

Longest common subsequence: we measure the lexical similarity of each c_i and q with the term-level longest common subsequence (LCS) [1]. The length of LCS is normalized by dividing the maximum length of the two texts.

Word overlap: we calculate the normalized count of common ngrams (n=1,2,3) and nouns.

Tree kernel: tree kernels are similarity functions used to measure the syntactic similarity of a text pair. Here we employ the subtree kernel (ST) [9] to compute their syntactic distance.

Translation probability: we learn word-to-word translation probabilities using GIZA++ [7] with the index data. In training, we regard posts as source language and their comments as target language. Following [4], we use the translation probability $p(\text{comment}|\text{query})$ as a similarity feature.

Word embedding cosine: we employ a pre-trained word embedding on the index data, where the dimensionality of word vectors is 200. We average the embedding of words in a piece of text as its representation, and compute the cosine of the representations of two pieces of text as a feature.

GRU: The recurrent neural network with gated units (GRU) [3], that is able to model the sequential information of a text, is employed to compute the sentence embeddings of q and c_i, denoted as $\vec{q}$ and $\vec{c_i}$ respectively. Then we use a bilinear function to compute the similarity between q and c_i, which is written as

$$sim(q, c_i) = \vec{q}^{\top} W \vec{c_i} + b, \tag{4}$$

where W and b are two parameters.

CNNs: Convolutional Neural Network (CNN) models the n-gram information in a text, that has proven effective on

[7]http://www.statmt.org/moses/giza/GIZA++.html

Table 1: Official Statistics of dataset for Chinese subtask

Repository	#posts	219,174
	#comments	4,305,706
	#original pairs	4,433,949
Labeled Data	#posts	769
	#comments	11,535
	#labeled pairs	11,535
Test Data	#query posts	100

various NLP tasks such as text classification [5], question answering [15] and etc,. Here we use both of term level CNN and char level CNN to represent texts, and then use a bilinear function to compute text similarity score.

We learn the GRU and CNN network by minimizing cross entropy on training data. Let Θ denote the parameters, then the objective function can be formulated as

$$-\sum_{i=1}^{2N}\Big[l_i log(f(p_i, c_i^{(\cdot)})) + (1-l_i)log(1-f(p_i, c_i^{(\cdot)}))\Big], \quad (5)$$

where $l_i \in \{0,1\}$ is a label, $c_i^{(\cdot)}$ means it may be c_i^+ or c_i^-.

2.3.2 Feature Combination

Since our objective can be formulated as a ranking problem, we learn a gradient boosted regression tree using XgBoost [2] as a ranking model to combine all features. The ranking model is learned by minimizing pairwise loss on training instances provided by the annotated training data. We denote the model as $f(q, c_i)$.

2.3.3 Ranking comments

We consider both of query-comment similarity and query-post similarity in our reranking algorithm, which is written as

$$Score(q, c_i) = g(q, p_i) + \alpha \cdot f(q, c_i), \quad (6)$$

where α is 0.8 in our implementation. $g(q, p_i)$ is the semantic distance between the query and the post, which is computed by $Cos\Big(\mathbf{GRU}(p_i), \mathbf{GRU}(q)\Big)$.

3. EXPERIMENTS

Table 2: XgBoost Settings

eta	0.01
num rounds	200
max depth	6
gamma	6
min child weight	1
colsample bytree	0.8
subsample	0.6

The STC Chinese subtask uses post-comment pairs crawling from Weibo to construct a million-scale repository. Table 1 shows the details of the index data. Besides, 769 query posts and 11,535 candidate comments are manually labeled as the annotated training data. Note that 100 query posts are carefully selected as the test data.

3.1 Parameter Settings

3.1.1 Feature Computation Settings

To train the GRU and CNNs, the post-comment pairs in the repository is considered as the positive samples. And for each post, we randomly select the same number of comment from other pairs as the negative samples. The word embedding size of GRU and CNN is fixed as 200, and the pre-trained word embeddings is used to initialize. The hidden unit size of GRU is 200. And the convolution kernel size in CNNs is set to 3*200, 4*200 and 5*200 respectively. The GRU and CNNs are both optimized by Adam to minimize the cross-entropy loss. The learning rate is fixed to 0.0001.

Xgboost is used to combine all the similarity features. We split the training data into 5 folds and adopt the cross validation to avoid overfitting. We aim to minimize the pairwise loss in XgBoost and use the nD@1 to evaluate the performance. After training and validation, the final settings is shown in Table 2.

3.1.2 5 Run Settings

We submit 5 retrieval-based runs in the task. The settings of each run are shown as follows:

- R1: We first use the keyword-based approach R_{word} to generate candidate post-comment pairs. After that, we only consider the query post similarity (i.e., $\alpha = 0$) in the reranking strategy.
- R2: R_{word} is also used for candidate generation. We consider query-post and query-comment similarities in the reranking strategy following Equation 6.
- R3: R3 utilizes the sentence embedding based approach R_{embed} for candidate retrieval. The ranking method is the same with R2.
- R4: R4 utilizes the symbolic knowledge enhanced approach to retrieve the candidates and reranks these candidates by Equation 6.
- R5: We merge candidates comments given by R_{word} and R_{embed} and employ the reranking strategy in Equation 6.

3.2 Results and Analysis

In the STC Chinese subtask, three different metrics are employed for the evaluation including nG@1 (normalised gain at cutoff 1), P+, and nERR@10 (normalised expected reciprocal rank at cutoff 10) [10].

64 retrieval-based and 56 generation-based runs are submitted for the Chinese subtask. Table 3 shows the results for top 15 runs. One can see that SG1 achieves the best result both in the retrieval-based and the generation-based subtask. Our submission achieves the second best performance in the retrieval-based subtask in terms of the nG@1 metric.

The results of our 5 runs are shown in Table 5. Comparing to the baseline R1, the improvement of R2 shows the advantage of the reranking. Note that R4 outperforms other 4 runs, demonstrating that symbolic information is useful in a retrieval based chatbot. R3 and R5 do not achieve the performance as we expect, since the the quality of the sentence embedding is not good enough.

To further investigate the characteristics of 5 runs, we conduct a qualitative analysis which is shown in Table 4. Run 1 and Run 2 are able to handle a post-comment pair

Table 3: Official STC Results for the top 15 runs

Run	Mean nG@1	Run	Mean P+	Run	Mean nERR@10
SG01-C-G1	0.5867	SG01-C-G1	0.6670	SG01-C-G1	0.7095
SG01-C-G3	0.5633	SG01-C-G3	0.6567	SG01-C-G3	0.6947
SG01-C-G2	0.5483	SG01-C-G2	0.6335	SG01-C-G2	0.6783
SG01-C-R1	0.5355	SG01-C-R3	0.6200	SG01-C-R3	0.6663
SG01-C-R2	0.5168	SG01-C-R1	0.6084	SG01-C-R1	0.6579
splab-C-G4	0.5080	splab-C-G4	0.6080	splab-C-G4	0.6492
SG01-C-R3	0.5048	SG01-C-R2	0.5944	SG01-C-R2	0.6461
Beihang-C-R4	0.4980	Beihang-C-R4	0.5818	splab-C-G1	0.6282
splab-C-G1	0.4848	splab-C-G1	0.5768	splab-C-G5	0.6175
Nders-C-R4	0.4780	splab-C-G5	0.5657	SG01-C-G4	0.6129
Nders-C-R2	0.4743	DeepIntell-C-R1	0.5564	Beihang-C-R4	0.6105
Nders-C-R3	0.4647	SG01-C-G4	0.5545	DeepIntell-C-R1	0.5994
Nders-C-R1	0.4593	Beihang-C-R2	0.5510	splab-C-G3	0.5966
Nders-C-R5	0.4550	Nders-C-R2	0.5497	Nders-C-R2	0.5882
Beihang-C-R2	0.4510	Nders-C-R5	0.5495	Nders-C-R5	0.5868

Table 4: Case Examples of 5 Runs

Run Name	Case Examples
Beihang-C-R1	post:母亲节快乐，愿天下的母亲健康漂亮 cmnt:愿天下的母亲。母亲节快乐啊。
Beihang-C-R2	post:[泪]很喜欢几米每天分享的漫画围脖，大爱这个围脖！ cmnt:大爱几米漫画呀，不经意的文字却直触心里。
Beihang-C-R3	post:比特币洗白了，大利好啊，支持 cmnt:如果不出利好，市场会怎么说?
Beihang-C-R4	post:美国股市集体下跌，连续第三个交易日走低 cmnt:美国股市和国内比的好处是能上能下。比如：东南融通
Beihang-C-R5	post:木棉。最后一张是今早院门口。 cmnt:最后一张，象是油画，不错。

Table 5: Comparison of Performance on 5 Runs

Run Name	Mean nG@1	Mean P+	Mean nERR@10
Beihang-C-R1	0.4343	0.5441	0.5643
Beihang-C-R2	0.4510	0.5818	0.5716
Beihang-C-R3	0.4080	0.5441	0.5623
Beihang-C-R4	**0.4980**	**0.5818**	**0.6105**
Beihang-C-R5	0.3937	0.5215	0.5544

that shares several terms, since the retrieval is conducted in a term overlap way. Run 3 requires the entity overlap between a post and a comment, such as "利好" in the example. Run 4 and Run 5 are good at dealing with post-comment pairs have high similarity scores in the semantic space.

4. CONCLUSIONS

We focus on the candidate retrieval problem in the retrieval based chatbots, and propose two novel approaches for this problem. The preliminary results show that symbolic information can significantly boost the performance of a retrieval based chatbot.

5. REFERENCES

[1] L. Allison and T. I. Dix. A bit-string longest-common-subsequence algorithm. *Information Processing Letters*, 23(5):305–310, 1986.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.

[4] Z. Ji, Z. Lu, and H. Li. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*, 2014.

[5] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[6] 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. *arXiv preprint arXiv:1607.00970*, 2016.

[7] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[8] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[9] B. Schlkopf, K. Tsuda, and J. Vert. Fast kernels for string and tree matching. *NIPS*, 15(1):296, 2003.

[10] 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 *Proceedings of NTCIR-13*, 2017.

[11] L. Shang, T. Sakai, Z. Lu, H. Li, R. Higashinaka, and Y. Miyao. Overview of the ntcir-12 short text conversation task. *Proceedings of NTCIR-12*, pages 473–484, 2016.

[12] M. Wang, Z. Lu, H. Li, and Q. Liu. Syntax-based deep matching of short texts. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[13] Y. Wu, W. Wu, X. Chen, M. Zhou, and Z. Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *ACL 2017*, 2017.

[14] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, and W.-Y. Ma. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340*, 2016.

[15] W. Yin and H. Schütze. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *ACL*, 2015.