Microsoft Research Asia at NTCIR-12 STC Task

Zhongxia Chen
University of Science and Technology,Hefei,P.R.C
czx87@mail.ustc.edu.cn

Ruihua Song
Microsoft Research Asia,Beijing,P.R.C
song.ruihua@microsoft.com

Xing Xie
Microsoft Research Asia,Beijing,P.R.C
xing.xie@microsoft.com

ABSTRACT

This paper describes our approaches at NTCIR-12 short text conversation (STC) task (Chinese). For a new post, instead of considering post-comment similarity, our system focus on finding similar posts in the repository and retrieve their corresponding comments. Meanwhile, we choose frequency property of comments to adjust ranking models. Our best run achieves 0.4854 for mean P^+, 0.3367 for mean nDCG@1 and 0.4592 for mean nERR@10, which reaches the top tier in official STC results.

Team Name

MSRSC

Subtasks

Short Text Conversation (Chinese)

Keywords

Short Text Conversation, Word Segmentation, Word to Vector, Post-Post Similarity, Frequency

1. INTRODUCTION

We participated in the NTCIR-12 Short Text Conversation (STC) subtask. Given a new post, this task aims to retrieve an appropriate comment from a large post-comment repository. The retrieved comment is judged from four facets: **Coherent**，**Topically relevant**, **Non-repetitive** and **Context independent** [2].

The principle of a suitable comment is that this comment keeps talking about the same topic with the given post. In a previous work of retrieval-based STC, Ji et al. [1] directly focus on the similarity between this new post and each comments to pursue the principle. That is, when the new post and a candidate comment share same words or phrases, we can strongly infer that they may be on the same topic.

However, this is not a prerequisite of being a proper response of the post. For example, the post talks about the travel plan to Shanghai (a city in south China). An appropriate comment could be "envying you" (See Table 1). In this case, post and comment is different in words but coherent. Such kind of comment is also non-repetitive.

Different from Ji et al.'s work, we try to explore other characteristics of posts and comments for solving the short text conversation problem. Suppose there is a new post that is same as an old post in repository. That is, like an Information Retrieval problem, comments corresponding to the old post will be perfect responses. Therefore, to be more general, our assumption is that **similar posts has similar corresponding comments**.

This hypothesis looks reasonable, but not all similar comments are appropriate responses. For example, for the test post again, a similar post and its two corresponding comments are shown in Table 1. Both posts express the willing to some city for sightseeing. Comment 2 shares the feeling of envy that means I want it too. It fits the test post as well, whereas, Comment 1 is not appropriate because it recommends the Summer Palace, which is a tourist interest in Beijing (not Shanghai).

Test Post	看这几天天气挺好的，到上海去玩一圈 Since the weather is good these days, I'd like to go to Shanghai for a visit
Similar Post	看这几天天气挺好的，到北京去玩一圈 Since the weather is good these days, I'd like to go to Beijing for a visit
Comment 1	推荐去颐和园 I recommend you to go to the Summer Palace
Comment 2	羡慕 Envying you

Table 1: Example of posts and candidate comments

This example inspires us to take a second look on those two different kinds of comments. Comment 2 stands for common comments which are appropriate for not only one post. This kind of comment might express happiness, shock, gratefulness, encouragement or other common attitudes toward a micro-blog. We list top ten comments that are most commonly used in the repository (Table 2). The total number of post-comment pairs is 5,648,128. The comment of "envying you" is used 412 times as responses.

Conversely, a particular comment like Comment 1 above may informative and completely fit the corresponding post in the repository. But it may not easy to meet the context of the new post and may compromise the criteria of **Coherent** or **Context independent** when responding new post. So to some extent, the popularity of a comment in the repository represent how likely the comment is appropriate to respond posts in general.

Finally, we construct a similarity-based method to rank

Comment	Frequency
哈哈 (laugh)	14830
哈哈哈 (laugh)	8096
呵呵 (no comment)	8075
嗯 (fine)	4443
哈哈哈哈 (laugh)	4297
不错 (not bad)	4222
好 (good)	3676
是的 (yes)	3472
喜欢 (I like it)	3216
赞 (great)	3209

Table 2: Top 10 popular comments in repository

comments in repository, regarding post-post similarity as a major feature. Furthermore, we take comment frequency as a modulatory feature to make sure that the comments are appropriate.

2. SYSTEM DESCRIPTION

Our system consists of preprocessing, feature generating and ranking (see in Figure 1). For all post data, we calculate similarity feature; for comment data in repository, we calculate frequency feature. For every test post, we set every comment in the repository as a candidate. Then we use feature vectors of training data to generate feature weights. Finally we rank comments as result by training model.

2.1 Preprocessing

Unlike English words in a sentence are separated by spaces, Chinese short texts are written without any symbol between characters. So the word segmentation becomes necessary.

We choose C# implement package of Stanford Chinese Word Segmenter in Nuget[1]. After segmentation, our system filters meaningless words and symbols according to Chinese stop words list.

The following example in Table 3 shows the origin text, segmentation result and filtering result.

Short Text ID	repos-cmnt-1000578260
Origin Text	西班牙踢球多黏啊总看传球了’意大利必胜 (Spain always passes and I will support Italy)
Segmentation Result	西班牙 踢球 多 黏 啊 总 看 传球 了 ’ 意大利 必胜
Filtering Result	西班牙 踢球 黏 总 传球 意大利 必胜

Table 3: Example of Preprocessing

2.2 Feature Generation

[1]http://sergey-tihon.github.io/Stanford.NLP.NET/StanfordWordSegmenter.html

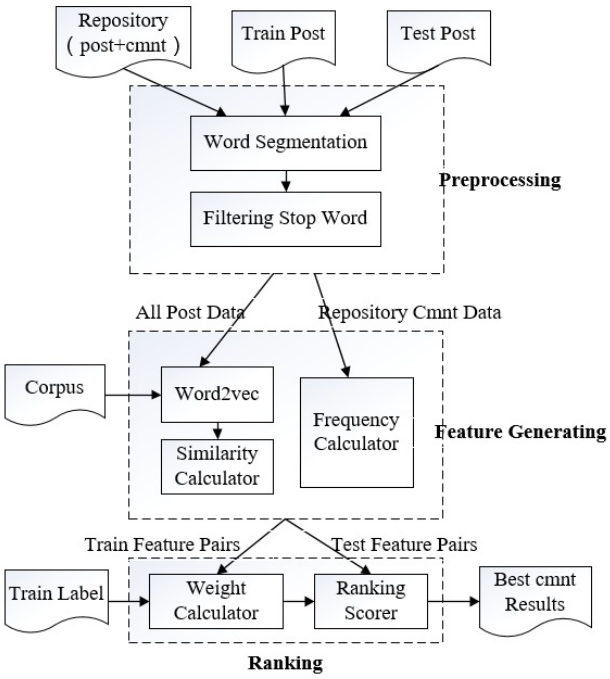

Figure 1: System Architecture

In this module, our system extracts similarity feature for post texts and frequency feature for comment texts in repository.

2.2.1 *Similarity feature*

This fundamental feature is hoped to represent semantic similarity of short texts. A primitive thought is turning these short text sentences into feature vectors and calculate.

1. **Google Word2Vec**[2]

   Word to vector is an efficient tool for computing continuous distributed representations of words. Since every short text post or comment consists of several words, it is available to combine vectors of words in these short texts to gain a representation of the whole text. In our method, we combine word vectors with summation. It is simple to implement and is acceptable for further calculation because these texts are short after preprocessing and will not cause a large amount of texts with different meanings but have same summation vectors.

2. **Chinese Corpus**

   Word2vec learns vector representations from training text Corpus with skip-gram architecture. Since short conversations have shortage for lack of contextual information, we choose an appropriate external Chinese corpus[3] released by Institute of Automation& Chinese Academy of Sciences instead of repository data.

3. **Cosine Similarity**

   The cosine similarity is a widely used approach to quantify similarity of summation vectors of all post

[2]https://code.google.com/archive/p/word2vec/
[3]http://www.datatang.com/data/13484

data. Suppose there are short text posts p_1 and p_2, corresponding to word vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ respectively, cosine similarity is defined as follow:

$$S(p_1, p_2) = \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\|\|\overrightarrow{v_2}\|}$$

That is, given a new post p, for every candidate comments C_i in the repository, the **similarity feature** of C_i responding p is:

$$F_{sim}(p, C_i) = S(p, P_i)$$

Here (C_i, P_i) is the corresponding post-comment pair in the repository.

2.2.2 *Frequency feature*

This feature measures how likely a comment be a general response. Considering the Power Law, we simply define the frequency feature as follow:

$$F_{fre}(p, C_i) = \log\left((C_i)_{frequency}\right)$$

Here $(C_i)_{frequency}$ stands for population of comment C_i in the repository.

2.3 Ranking

Given a new post, our method is to calculate scores of candidate comments according to their feature vectors and apply ranking.

Since the feature vectors just contain similarity feature and frequency feature, a primitive method is using similarity to retrieving top k comments with closed corresponding posts and using frequency feature to rerank comments in a small range.

Further, we choose a linear model as follow:

$$Score(p, C_i) = F_{sim}(p, C_i) + w * F_{fre}(p, C_i)$$

We use grid search to optimize the parameter w , maximizing gain (which is labeled by train-label file:+2 for L2-relevant,+1 for L1-relevant and 0 for not relevant) summation of ten recommended comments at top:

$$w = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{j=1}^{10} Gain(P_i, \overline{C_{ij}})$$

Here n is the number of training posts, $\overline{C_{ij}}$ is the jth comment (ranked by score) in all labeled comments in repository for training post i, $Gain(P_i, \overline{C_{ij}}) \in \{0, 1, 2\}$.

For test file, we also keep ten comments at top after ranking by scores as submit result.

3. EXPERIMENTS AND DISCUSSION

We submitted three runs for comparison and analysis:

- MSRSC-C-R1: Use full features described above for modeling and ranking, hope to gain better performance than other two runs.
- MSRSC-C-R2: Only similarity-based model for ranking, then rerank top fifty comments by their populations in descending order (which is described above). Here we choose top fifty comments in the first step.
- MSRSC-C-R3: Only similarity-based model for ranking. Specifically, we rank the comments by their IDs if they have the same corresponding post. The result is regarded as baseline for comparison.

We used the same setting in applying word2vec for every run: a skip-gram model that window size is 10 and vector length is 100. There are totally 131340 vocabularies and 33133315 words in the corpus.

STC task use three different measures for evaluation: $nDCG$@1, $nERR$@10 (Expected Reciprocal Rank) and P^+ (the bigger the better) [2]. The experiment results of three runs are shown in following Table 4:

Run	Mean nDCG@1	Mean P^+	Mean nERR@10
MSRSC-C-R1	**0.3367**	**0.4854**	**0.4592**
MSRSC-C-R2	0.2733	0.4208	0.3857
MSRSC-C-R3	0.0933	0.2420	0.2236

Table 4: Official STC results for team MSRSC

Comparing to baseline MSRSC-C-R3, the results of MSRSC-C-R1 and MSRSC-C-R2 both improves visibly as we expect. That is to say, the population of comments seems to be surprisingly helpful in this task.

Moreover, the result of MSRSC-C-R2 infers that the strategy of retrieving with similarity and ranking with frequency contributes to a certain extent but still has shortage comparing with modeling method is MSRSC-C-R1. A credible explanation is that the range of best fifty similar posts is uncertain. If the rank fiftieth post, for instance, differs from the test post apparently, some improper common comments might be ranked at top because of their high frequencies as long as their corresponding posts are in top fifty.

Besides, results in Table 4 stand for general expectation. There are some exceptions as well in particular test posts. (See in Table 5)

Test Post ID	MSRSC-C-R1	MSRSC-C-R2	MSRSC-C-R3
test-post-10240	0.2679	0.9697	0.2173
test-post-10250	0.344	0.1633	0.9571

Table 5: The exceptional nERR@10 results of some test cases

In these two cases, MSRSC-C-R2 and MSRSC-C-R3 perform better on nERR@10 respectively (similar results by comparing other measures).

Meanwhile, ten best results of overall runs in Official STC results are shown in Table 6.

Our run MSRSC-C-R1 is competitive for every measure, especially the 3rd place evaluating with P^+ measure.

4. CONCLUSIONS

We proposed a system to rank candidate comments in the repository and find appropriate responses of a new post. First we segment data and generate vectors of posts. After we extract similarity feature and frequency feature, we propose three ranking models and eventually applied them to the test post for ranking and retrieving proper comments. The mean P^+ of our best run was 0.4854, the 3rd place in all official STC results.

Since human evaluation data of test pairs has released, the labeled dataset is enlarged, we could use these official

Run	Mean nDCG@1	Run	Mean P^+	Run	Mean nERR@10
BUPTTeam-C-R4	**0.3567**	BUPTTeam-C-R4	**0.5082**	BUPTTeam-C-R4	**0.4945**
BUPTTeam-C-R3	0.3533	BUPTTeam-C-R2	0.4933	BUPTTeam-C-R2	0.4830
BUPTTeam-C-R2	0.3533	BUPTTeam-C-R1	0.4883	BUPTTeam-C-R3	0.4805
BUPTTeam-C-R5	0.3467	**MSRSC-C-R1**	**0.4854**	BUPTTeam-C-R5	0.4800
BUPTTeam-C-R1	0.3400	BUPTTeam-C-R3	0.4853	BUPTTeam-C-R1	0.4770
MSRSC-C-R1	**0.3367**	BUPTTeam-C-R5	0.4840	**MSRSC-C-R1**	**0.4592**
OKSAT-C-R1	0.3267	splab-C-R1	0.4735	splab-C-R1	0.4449
ITNLP-C-R3	0.3067	OKSAT-C-R1	0.4691	Nders-C-R1	0.4196
splab-C-R1	0.2933	USTC-C-R5	0.4509	ITNLP-C-R3	0.4186
ITNLP-C-R2	0.2900	USTC-C-R1	0.4499	USTC-C-R4	0.4181

Table 6: Official STC results for best ten runs (three measures respectively)

results as comparison and work on new dataset to seek some new thought for future improvements.

5. REFERENCES

[1] Z. Ji, Z. Lu, and H. Li. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988, 2014.

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