UWNLP at the NTCIR-12 Short Text Conversation Task

Anqi Cui
University of Waterloo, Canada
caq@uwaterloo.ca

Guangyu Feng
University of Waterloo, Canada
gfeng@uwaterloo.ca

Borui Ye
University of Waterloo, Canada
b7ye@uwaterloo.ca

Kun Xiong
University of Waterloo, Canada
xiongkun04@gmail.com

Xing Yi Liu
University of Waterloo, Canada
liuxingyi99@gmail.com

Ming Li
University of Waterloo, Canada
mli@uwaterloo.ca

ABSTRACT

In this paper, we describe our submission to the NTCIR-12 Short Text Conversation task. We consider short text conversation as a community Question-Answering problem, hence we solve this task in three steps: First, we retrieve a set of candidate posts from a pre-built indexing service. Second, these candidate posts are ranked according to their similarity with the original input post. Finally, we rank the comments to the top-ranked posts and output these comments as answers. Two ranking models and three comment selection strategies have been introduced to generate five runs. Among them, our best approach receives performances of mean nDCG@1 0.2767, mean P+ 0.4284 and mean nERR@10 0.4095.

Team Name

uwnlp

Subtasks

Short Text Conversation (Chinese)

Keywords

community question answering, semantic distance, information retrieval

1. INTRODUCTION

The UWNLP team participated in the Short Text Conversation (STC) pilot task in NTCIR-12 [8]. The team comprises members from University of Waterloo and RSVP Technologies Inc., a Waterloo-based start-up company. We analyzed and studied the task and evaluation methods carefully, and submitted five runs based on our best understanding.

Conversation is one of the challenging problems in the field of natural language processing. In this STC task, it treats conversation as an information retrieval (IR) problem, by responding the input from existing repository. Hence we adopt the community Question-Answering (cQA) framework to solve this task.

In a cQA problem, the existing repository consists of a huge number of question-answer pairs. The cQA algorithm answers the input question by retrieving a similar question in the repository and use the corresponding answer as the output. As long as the repository is big enough, we are then able to answer every possible question.

Following this philosophy, we build a repository of post-comment pairs from microblogging sites. Then given an input sentence (new post), we retrieve some similar posts from this repository, and use their corresponding comments as the responses.

We have investigated several ranking models, both supervised and unsupervised, involving character-based and word-based features. We have also trained a Word2Vec model to generate more features. Different strategies are applied to select the best comment. We aim at discovering the most similar sentences (questions), i.e. shortest semantic distance between the input and the post candidates [3].

The rest of the paper is organized as follows: Detailed algorithm, features and models are introduced in Section 2. Submitted results are shown and discussed in Section 3. We conclude our paper in Section 4.

2. THE CQA-BASED ALGORITHM

2.1 Algorithm Architecture

We consider this short text conversation as a cQA task, hence our algorithm follows the retrieval-based cQA framework: It retrieves posts in the post-comment repository as "questions", and then uses the corresponding comments to reply as "answers". Under this philosophy, we believe that comments are the best responses to the original sentence, hence as long as we find out the most similar "question" to the user's input, we would discover the best "answer" from the comments repository.

2.2 Post-comment Indexing

We index all the posts to easily retrieve relevant posts from the input sentence. In this way we do not need to scan all posts for each input, but instead retrieve a limit number of posts efficiently.

We apply Apache Solr [1], the Lucene-based indexing service to index the post-comment pairs. In practice, we use the default settings of Solr version 5.3.1. Since the retrieval algorithm only retrieves relevant posts, we index only the post texts in the service. We store all data of the pairs, including post id, comments (text and their id's) within the Solr service as well. Note we store all the comments of a post together in one record, i.e. the number of the records is equal to the number of different posts.

Tokenization of the input posts is achieved by Chinese word segmentation, hence we could retrieve the relevant posts from a given sentence. We use Ansj [7] version 2.0.7 for Chinese word segmentation. This algorithm is based on the Google semantics model and conditional random fields (CRFs).

The process of indexing significantly increases the time-efficiency of retrieving relevant posts. However the ranking of "relevant" posts may not be suitable for our goal, since we aim to discover the best reply to the original sentence; the posts are acting as a bridge between the new post and the old comments. Hence we develop our own ranking algorithm to find the most proper posts which contain the best comments.

2.3 Posts Ranking

After retrieving some relevant posts in the repository, we rank these posts by the similarity between the input and the posts.

2.3.1 Features

The similarity methods are based on some textual features, including character-based and word-based features:

1. Character-based features:
   (a) Length of the longest common substring (LCS) between A and B.
   (b) Overlapping of any character in the two sentences, 1 if at least one same character, 0 if not.
2. Word-based features:
   (a) Cosine similarity [9]:

$$\cos(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

   where A_i and B_i are components (words) of sentence vector A and B respectively.
   (b) Overlap similarity [5]:

$$\text{overlap}(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)} \quad (2)$$

   (c) Word order similarity [4]:

$$\text{order}(r_1,r_2) = 1 - \frac{||r_1 - r_2||}{||r_1 + r_2||} \quad (3)$$

   where r_1 and r_2 are the word order vectors of the sentences.
   (d) Inverse document frequency (IDF) scores: The sum of the IDF of the common words in the sentence, based on a large set of Chinese sentences.
   (e) Latent semantic analysis (LSA) similarity [2]: The sentence vectors are decomposed by singular value decomposition (SVD), and the similarity is then computed by the cosine similarity of the two matrices.
   (f) Word2Vec similarity [6]: The Word2Vec similarity is the cosine similarity between two word vectors, generated from the sentence word vectors with the additive synthesize method.

For the word-based features, we firstly segment the sentences with the Ansj word segmentation algorithm, and then generate the sentence vector with the bag-of-words (BoW) model. In addition, for the features 2a and 2b, we select some *important words* in each sentence to form a new sentence vector, hence generate two new features.

The *important words* are determined by their Part-Of-Speech (POS) tags, including: Nouns, verbs, adjectives, conjunctives, distinguishing words, numbers, and words denoting time, places etc. These words usually carry more semantic information hence we increase their weights.

2.3.2 Ranking Models

Based on the features mentioned above, each retrieved post, as a candidate, is assigned a score for ranking. The score is generated from two similarity methods:

- Linear combination: The score is combined from the features 1a, 1b, 2a (all-words and important-words) and 2b (all-words and important-words). Each of the features is given a weight of 0.5, except 2a (all-words) and 2b (all-words), with a weight of 1.
- Random forest: The score is generated from a random forest model, using all the features mentioned above.

The candidate posts are ranked in an order, that the posts with higher scores are more relevant or similar to the input. The comments to these posts are more suitable to reply the original input.

2.3.3 Comment Selection

In addition to the similarity between the input and the post, we also compute the similarity between the input and the comments. We then rank the comments by different strategies, by assigning scores to the comments:

1. Length: The score of the comment is its length, i.e. number of characters of the comment.
2. Max comment similarity: For each post p, we retrieve all its comments $\{c|c \in p\}$, and compute the similarity between the input q and these comments. Then we use the comment with the maximum similarity score as a representative, and add up this score to the post similarity score. I.e.,

$$score'(p) = sim(q,p) + \max_{c \in p}\{sim(q,c)\} \quad (4)$$

   After that, we use the post with the highest $score'$, and output all its comments as the submitted results.
3. Combined similarity: In this strategy we consider each comment separately. The score is assigned to the comment c itself, i.e.

$$score'(c) = sim_{c \in p}(q,p) + sim(q,c) \quad (5)$$

   Then rank all the comments by this score, and output the comments with the highest scores.

2.4 Experiment Settings

We have collected 200 million question-answer pairs from two of the biggest Chinese cQA websites: Baidu Zhidao[1]

[1] http://zhidao.baidu.com/

and Sogou Wenwen[2]. These pairs are used both as the IDF repository and the training data of the Word2Vec model. In practice the word vectors consist of 200 dimensions.

We have gathered 3,809 sentence pairs from search query logs of Baidu[3], the biggest search engine in China, annotated with labels *same meaning*, *similar*, and *dissimilar*. These pairs serve as the training data for the random forest model. In practice the random forest model is set up with 97 trees.

3. SUBMITTED RESULTS

In NTCIR-12 STC task, we have submitted five runs, corresponding to different combinations of ranking models and comment selection methods, as shown in Table 1.

Table 1: Ranking Models and Comment Selection Methods of the Submitted Runs

Run ID	Ranking model	Comment selection
uwnlp-C-R1	Linear combination	Combined sim.
uwnlp-C-R2		Max comment sim.
uwnlp-C-R3		Length
uwnlp-C-R4	Random forest	Combined sim.
uwnlp-C-R5		Max comment sim.

The evaluated results provided by the task organizers are shown in Table 2 [8]. The evaluation measures are also introduced in the reference [8].

Table 2: Ranking Models and Comment Selection Methods of the Submitted Runs

Run ID	Mean nDCG@1	Mean P+	Mean nERR@10
uwnlp-C-R1	0.2767	0.4284	0.4095
uwnlp-C-R2	0.2767	0.3977	0.3740
uwnlp-C-R3	0.1733	0.2564	0.2255
uwnlp-C-R4	0.1033	0.2085	0.1867
uwnlp-C-R5	0.1067	0.1862	0.1732

From the results we discover that the first three runs have better results than the rest two. We can conclude that the linear combination model ranks the posts better than the random forest model. This is mainly because of the different distribution of our training data against the post-comment repository. Especially, the search engine queries do not carry all the semantic information as in conversational sentences.

Moreover, the combined similarity method is better than the maximum comment similarity method, showing that the comments also provide useful semantic information with the posts. Sometimes the comment containing some repeating words in the post, showing that it is more relevant towards the topic of the post. This is also proved by the results that runs R1 and R2 are better than R4 and R5, since they retrieve better posts.

Within the runs of the linear combination ranking model, the run R3 is the worst among the three runs. This run chooses longer comments as better ones. We can conclude that although they are longer, the information they contain may be irrelevant. They may even about a different topic (spams or advertisements). However, the fact that R3 is even better than R4 and R5 proves that the retrieving a relevant post is more important, as comments to the irrelevant posts are totally non-related to the original input.

In addition to the results provided by the organizers, we have also summarized the number of topics which we provide good comments. Table 3 shows that out of the 100 test topics, we could comment (reply) to 17 topics perfectly (labeled as L2) with our first candidate (@1), and similarly, we reply to 93 topics with acceptable comments (labeled as L1 or L2) within the ten candidates (@10). We can conclude from these results that, in a practical conversation system with only one reply per input, our method may only reply with satisfiable answers one fifth of the time. However if we could furthermore choose one of the ten candidates wisely, we may achieve a satisfaction rate of more than 90%.

Table 3: Number of Test Topics Containing Good Comments. #L2@n stands for the number of posts we reply with at least one L2 comment within the top-n candidates, and similarly L1+ for L1 or L2.

Run ID	#L2@1	#L1+@1	#L2@10	#L1+@10
uwnlp-C-R1	17	25	64	93
uwnlp-C-R2	17	25	49	80
uwnlp-C-R3	8	12	32	59
uwnlp-C-R4	8	7	35	61
uwnlp-C-R5	8	8	30	53

As revealed from these results, we are interested in combining the comment candidates from different strategies together and vote to rank them:

1. For each post (test topic), we first discover at most ten candidate comments from each of the five strategies, together with their rank (one to ten). There may be duplicate comments generated by different strategies.

2. Then we merge the candidates together, and rank the merged list again according to the score. The score to each candidate is assigned with a sum of five scores from all strategies, where the top candidate in a strategy has a score of ten, the second candidate has a score of nine, etc.

3. Finally we examine the top-one (@1) or top-ten (@10) candidate comments.

The performance of this merge-and-vote strategy is:

- #L2@1: 19
- #L1+@1: 42
- #L2@10: 47
- #L1+@10: 79

Therefore we may conclude that the top candidate becomes better than a single strategy – the majority rule helps discover the best comment. However, the top-ten results become worse probably because the better choice is influenced by worse strategies. Nevertheless, we may choose different strategies (single or combined) according to different scenarios and different applications.

[2]http://wenwen.sogou.com/
[3]http://www.baidu.com/

4. CONCLUSIONS

In this paper, we describe our efforts in the participation of the NTCIR-12 STC task. We have explored some textual features to capture the similarity between two sentences. Our results show that these features help us rank the relevant posts. Moreover, we discover that longer comments may be better to respond the original post as they carry more information.

For the future work, we will continue digging out better features and models to measure the semantic distance between sentences.

5. ACKNOWLEDGEMENTS

We would like to thank the organizers for organizing this task. This work has been supported by an NSERC grant OGP0046506, the Canada Research Chair program, an ORF grant 115354, and CFI.

6. REFERENCES

[1] Apache Solr. Apache solr. http://lucene.apache.org/solr/. Accessed: 2016-02-22.

[2] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[3] G. Feng, K. Xiong, Y. Tang, A. Cui, J. Bai, H. Li, Q. Yang, and M. Li. Question classification by approximating semantics. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 407–417. International World Wide Web Conferences Steering Committee, 2015.

[4] T. K. Landauer, D. Laham, B. Rehder, and M. E. Schreiner. How well can passage meaning be derived without using word order? a comparison of latent semantic analysis and humans. In *Proceedings of the 19th annual meeting of the Cognitive Science Society*, pages 412–417, 1997.

[5] D. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 517–524. ACM, 2005.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] NLPchina. Ansj chinese word segmentation. https://github.com/NLPchina/ansj_seg. Accessed: 2016-02-22.

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

[9] Wikipedia. Cosine similarity. https://en.wikipedia.org/wiki/Cosine_similarity. Accessed: 2016-02-23.