ICL00 at the NTCIR-12 STC Task: Semantic-based Retrieval Method of Short Texts

Weikang Li.ICL00
Key Laboratory of Computational Linguistics (Peking University), Ministry of Education
wavejkd@163.com

Yixiu Wang.ICL00
Key Laboratory of Computational Linguistics (Peking University), Ministry of Education
labyrinth@pku.edu.cn

†Yunfang Wu.ICL00
Key Laboratory of Computational Linguistics (Peking University), Ministry of Education
wuyf@pku.edu.cn

ABSTRACT

We take part in the short text conversation task at NTCIR-12. We employ a semantic-based retrieval method to tackle this problem, by calculating textual similarity between posts and comments. Our method applies a rich-feature model to match post-comment pairs, by using semantic, grammar, n-gram and string features to extract high-level semantic meanings of text.

Team Name

ICL00

Subtasks

STC (Chinese)

Keywords

textual similarity, short text, semantic meaning computation

1. INTRODUCTION

We participate in the Chinese subtask at NTCIR12, which aims to solve the problem called Short Text Conversation (STC). Our registered name is "ICL00".

Our task aims at finding corresponding comments of a given post from a huge corpus of Chinese Weibo. We model the post-comment pairs mapping problem to a calculating of textual similarity, due to the following reasons: posts and corresponding comments usually share the similar topic; the corresponding comments are more similar with the post than other comments.

The research on textual similarity is a life-long topic, in which researchers have proposed many methods [1][2][3][4][5][6][7][8][9][10][11]. In this paper, we use word embedding to represent words of text. In order to apply syntactic information, we employ features of word order [12], information of dependencies [13] [14] [15] and semantic role [16] which are used in many other systems. Besides, we also make use of features like n-gram [17] and string, which are common in other systems.

This paper is organized as follows. We give detailed description of various features in Section 2, show our experiment and results in Section 3, and make a conclusion in Section 4.

2. FEATURES

Our method transfers the calculation of textual similarity to a classification problem. We train a SVM model based on various features between two sentences, and the output of classification probability (0.0 – 1.0) is regarded as the similarity score.

Table 2.1 Classification Features

No.	Category	Feature
1	Lexical	word embedding
2	Structure	dependency
3	N-gram	character unigram
4		character bigram
5		word unigram
6		word bigram
7		word trigram
8	String	Levenshtein distance
9		common word order

2.1 Lexical Features

Word embedding is widely applied in many task of NLP. Word and phrase embedding, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing and sentiment analysis [19].

We use word2vec to train word embedding based on the corpus “repos” given by the organizers, and we set the dimension of vector to 100.

For sentence *S-A* and sentence *S-B*, we would calculate the semantic similarity of each word in sentence *S-A* with sentence *S-B* by Equation 1. Words’ similarity is computed using cosine.

$$sim(w,S) = \arg\max_{w' \in S} sim(w,w') \tag{1}$$

We would take the average value gotten by the similarity value between words of one sentence and another respectively as semantic similarity of word embedding.

$$\mathrm{score}_{we}(S\text{-A}, S\text{-B}) = \frac{\sum_{w \in S-A}(sim(w, S-B) * idf(w))}{\sum_{w \in S-A} idf(w)} \tag{2}$$

$$\mathrm{sim}_{we}(S\text{-}A, S\text{-}B) = \frac{1}{2}(score_{we}(S\text{-}A, S\text{-}B) + score_{we}(S\text{-B}, S\text{-}A)) \tag{3}$$

The corpus to train word embedding has a direct influence on the performance of syntactic features. Besides, segmentation tool is an important impact element of word embedding, because the inputs of word2vec are words segmented by segmenter. We use Stanford’s segmenter to deal with task of segmentation in the paper.

2.2 Structure Features

Some significant information may be lost if we only use words' lexical knowledge, so we would take full advantage of depedndency knowledge to dig more features.

Stanford Parser is used to analyze dependencies of Chinese corpus to get structure of dependencies. We want to calculate syntactic information with the help of subject-predicate relationship and predicate-object relationship. The way referred to the paper [15] [16] is that extracting relationship of predicate-subject and predicate-object from the information of nsubj (nominal subject), nsubjpass (passive nominal subject), and dobj (direct object) produced by Stanford Parser.

For sentence *S-1* sentence *S-2*, we can get many pairs of *PS* and *PO* for that one sentence can be consist of several small sentences in Chinese corpus. We assume that *S-1* has PS_1 and PO_1, *S-2* has PS_2 and PO_2.

$$\mathrm{PS}_1 = \{ps_{11}, ps_{12} \ldots ps_{1m}\} \tag{4}$$

$$\mathrm{PO}_1 = \{po_{11}, po_{12} \ldots po_{1m'}\} \tag{5}$$

$$\mathrm{PS}_2 = \{ps_{21}, ps_{22} \ldots ps_{2n}\} \tag{6}$$

$$\mathrm{PO}_2 = \{po_{21}, po_{22} \ldots po_{2n'}\} \tag{7}$$

For PS_1 and PS_2, the similarity based on word embedding is the product of predicate and subject. And the situation is the same to PO_1 and PO_2.

$$\mathrm{sim}((ps)_{1i}, (ps)_{2j}) = sim(p_{1i}, p_{2j}) * sim(s_{1i}, s_{2j}) \tag{8}$$

$$\mathrm{sim}((po)_{1i}, (po)_{2j}) = sim(p_{1i}, p_{2j}) * sim(o_{1i}, o_{2j}) \tag{9}$$

We take the most similar pair in PS_1 and PS_2, which is also applicable to PO_1 and PO_2.

$$\mathrm{sc}ore_{ps} = \arg \max_{i \le m', j \le n'} sim((ps)_{1i}, (ps)_{2j}) \tag{10}$$

$$\mathrm{sc}ore_{po} = \arg \max_{i \le m, j \le n} sim((po)_{1i}, (po)_{2j}) \tag{11}$$

The average value of *PS* similarity score and *PO* similarity score is taken as the dependencies similarity.

$$\mathrm{sim}_{dep} = \frac{1}{2}(score_{ps} + score_{po}) \tag{12}$$

2.3 Features of N-gram

We have tried features of unigram, bigram, trigram in our experiment.

Table 2.2 Feature of N-gram

Name	Unit	n
word unigram	word	1
word bigram	word	2
word trigram	word	3
char unigram	character	1
char bigram	character	2
char trigram	character	3

2.4 Features of String

2.4.1 Levenshtein Distance

In many NLP's tasks, corpus from website has noise interference to deal with more or less. Inspired by the paper [1] [15], we take Levenshtein distance as the standard of evaluating the difference between strings. The Levenshtein distance between two sentences is the minimum number of single-word edits (i.e. insertions, deletions or substitutions) required to change one sentence into the other.

2.4.2 Common Word Order

We take features of common word order to our method with reference to the paper [1]. We assume that sentence *S-1* and *S-2* has word's order of $\{w_1, w_2, \ldots, w_m\}$ and $\{w_1, w_2, \ldots, w_n\}$ respectively. Then we would find same words to form words' order. If there are δ same words, words' order can be represented as $X = \{x_1, x_2, \ldots, x_\delta\}$ and $Y = \{y_1, y_2, \ldots, y_\delta\}$. x_i and y_i mean one same word's location in $S-1$ and $S-2$. The similarity of common word order can be calculated according to Equation 13.

$$sim_{word-order} = 1 - \frac{|x_1 - y_1| + |x_2 - y_2| + \ldots + |x_\delta - y_\delta|}{|x_1 - y_\delta| + |x_2 - y_{\delta-1}| + \ldots + |x_\delta - y_1|} \tag{13}$$

3. EXPERIMENT

3.1 Data Preprocessing

The given corpus contains redundant information, and informal using of punctuations. We used following steps of preprocessing to achieve better performance:

(1) Convert Punctuation

(2) Delete Redundant Punctuation

(3) Delete Stop Words

(4) Generalize Place Names(NS) : we use a NS entity recognizer developed in our previous work.

(5) Generalize URL

(6) Convert Tradition Chinese to Simplified Chinese

3.2 Experiment Settings

We use the tool LIBSVM to train our SVM model. In the given training corpus, those post-comment pairs with scores 1 and 2 are regarded as positive cases and others as negative cases.

Given a new post, we adopt two steps to find the appropriate comments:

(1) Exploit Lucene to search for comments in the given comment archive, and keep the top 100 ones as the candidate comments;

(2) Compute the textual similarity between post and candidate comments, and rank them to find the most appropriate comment.

We adopt different experiments to find the best combination of features, as shown in Table 3.1.

Table 3.1 Feature Setting

Name	Features
Baseline	cosine similarity weighted by tf-idf
Lab0: Lexical+N-gram	sentence's length, word embedding, word's unigram/bigram/trigram, character's bigram
Lab1: Lexical+N-gram+Struc	sentence's length, word embedding, dependencies, word's unigram/bigram/trigram, character's bigram
Lab2: Seman+N-gram+Struc+String	sentence's length, word embedding, dependencies, word's unigram/bigram/trigram, character's unigram/bigram, levenshtein distance, common word order

3.3 Evaluation Result

We use P@K method to evaluate our results on the given training data. The results are as follows:

Table 3.2 Experiment results

Name	**P@1**	**P@5**
Baseline	0.076	0.236
Lab 0	**0.107**	**0.287**
Lab 1	0.080	0.234
Lab 2	0.077	0.213

As shown in Table 3.2, the features combination of Lab 0 gets the best result in the training data.

There are three metrics in the overview paper to evaluate submitted results. Averagely, we have a middle rank in all results submitted [18].

Table 3.3 Evaluated results by the organizer

Name	**nDDG@1**	**P+**	**nERR@10**
Baseline	0.2633	0.4359	0.4066

3.4 Analysis of Results

From the above results, the combination of features in Lab 0 is useful to the task. We will give analysis of different features.

Lexical feature: word embedding has a significant influence on improving the accuracy of classifier.

Structure feature: structure features do not work well in this task. One main reason is that the performance of dependency parsing drops largely on the short text.

N-gram feature: Generally, this kind of features has a better performance on this task. N-gram features will capture some valid phrases, which might be left out by Stanford segmenter. The performance of segmentation also drops a lot on short texts like Weibo.

String Feature: This feature is not efficient enough. Although Levenshtein distance and common word order usually are efficient features in calculating similarity, they do not work well in computing the relation of post-comment pairs.

Our experimental results show that the combination features of word embedding and n-gram are most suitable for finding corresponding relationships between posts and comments. Dependent tree and string features are usually efficient in calculating similarities, but are not very valid for post-comment-pairs retrieval.

Although our method is simple and straightforward, it gets promising result in the official test. The experimental results show that there exist semantic similarities between some post-comment pairs. Of course, the limitation of our method is obvious. When the post-comment pair shows no similarity in surface form, our method doesn't work.

4. CONCLUSION

We adopt a semantic-based retrieval method to tackle the post-comment mapping problem at NTCIR-12 STC task. We train a SVM model to compute the similarities between posts and comments, integrating multi-level text information including word embedding, dependency, n-gram and string information. In the future, we will put up new ways to cope with those post-comment pairs that have no similarity in surface form.

Acknowledgement

We thank Tianyu Zhao for his previous work on computing textual similarity. This work is supported by National Natural Science Foundation of China (61371129), National High Technology Research and Development Program of China (2015AA015403) and Key Program of Social Science foundation of China (12&ZD227).

REFERENCES

[1] Islam A, Inkpen D. Semantic text similarity using corpus-based word similarity and string similarity[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2008, 2(2): 10.

[2] Landauer T K, Foltz P W, Laham D. An introduction to latent semantic analysis[J]. Discourse processes, 1998, 25(2-3): 259-284.

[3] Lund K, Burgess C. Producing high-dimensional semantic spaces from lexical co-occurrence[J]. Behavior Research Methods, Instruments, & Computers, 1996, 28(2): 203-208.

[4] Mihalcea R, Corley C, Strapparava C. Corpus-based and knowledge-based measures of text semantic similarity[C]//AAAI. 2006, 6: 775-780.

[5] Turney P. Mining the web for synonyms: PMI-IR versus LSA on TOEFL[J]. 2001.

[6] Leacock C, Chodorow M. Combining local context and WordNet sense similarity for word sense identification. WordNet, An Electronic Lexical Database[J]. 1998.

[7] Lesk M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone[C]//Proceedings of the 5th annual international conference on Systems documentation. ACM, 1986: 24-26.

[8] Wu Z, Palmer M. Verbs semantics and lexical selection[C]//Proceedings of the 32nd annual meeting on

Association for Computational Linguistics. Association for Computational Linguistics, 1994: 133-138.

[9] Resnik P. Using information content to evaluate semantic similarity in a taxonomy[J]. arXiv preprint cmp-lg/9511007, 1995.

[10] Lin D. An information-theoretic definition of similarity[C]//ICML. 1998, 98: 296-304.

[11] Jiang J J, Conrath D W. Semantic similarity based on corpus statistics and lexical taxonomy[J]. arXiv preprint cmp-lg/9709008, 1997.

[12] Li Y, McLean D, Bandar Z A, et al. Sentence similarity based on semantic nets and corpus statistics[J]. Knowledge and Data Engineering, IEEE Transactions on, 2006, 18(8): 1138-1150.

[13] Mohler M, Bunescu R, Mihalcea R. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments[C]//Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011: 752-762.

[14] Šarić F, Glavaš G, Karan M, et al. Takelab: Systems for measuring semantic text similarity[C]//Proceedings of the First Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics, 2012: 441-448.

[15] Zhu T T, Lan M. Measuring short Text Semantic Similarity using multiple measurements[C]//Machine Learning and Cybernetics (ICMLC), 2013 International Conference on. IEEE, 2013, 2: 808-813.

[16] Oliva J, Serrano J I, del Castillo M D, et al. SyMSS: A syntax-based measure for short-text semantic similarity[J]. Data & Knowledge Engineering, 2011, 70(4): 390-405.

[17] Malandrakis N, Iosif E, Potamianos A. DeepPurple: Estimating sentence semantic similarity using n-gram regression models and web snippets[C]//Proceedings of the First Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics, 2012: 565-570.

[18] Lifeng Shang, Tetsuya Sakai, Zhengdong Lu, Hang Li, Ryuichiro Higashinaka, Yusuke Miyao. Overview of the NTCIR-12 Short Text Conversation Task. NTCIR-12. 2015.

[19] Socher, Richard; Perelygin, Alex; Wu, Jean; Chuang, Jason; Manning, Chris; Ng, Andrew; Potts, Chris (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. EMNLP.