ITNLP: Pattern-based Short Text Conversation System at NTCIR-12

Yang Liu
Harbin Institute of Technology, China
yliu@insun.hit.edu.cn

Chengjie Sun
Harbin Institute of Technology, China
cjsun@insun.hit.edu.cn

Lei Lin
Harbin Institute of Technology, China
linl@insun.hit.edu.cn

Xiaolong Wang
Harbin Institute of Technology, China
wangxl@insun.hit.edu.cn

ABSTRACT

This paper describes the ITNLP system participated in the Short Text Conversation (STC Chinese subtask) of the NTCIR-12. We employed a Logistic Regression Model combined with pattern-based matching features to solve the STC problem. Deep learning methods were also tried in our experiment. Out of the 44 submitted runs our best performance run ***ITNLP-C-R3*** ranked 8th(Mean nDCG@1),12th(Mean P+) and 9th(Mean nERR@10) respectively [4].

Team Name

ITNLP

Subtasks

Short Text Conversation System (Chinese)

Keywords

artificial intelligence, dialogue systems, natural language processing

1. INTRODUCTION

Building an open-domain system that can communicate with humans using natural language has always been a dream of scientists and researchers. And it is still a difficult problem to solve in AI area. Although much efforts have been devoted in this problem, little progress was made. As large volume conversation data become possible to obtain on social media, which makes it possible to explore new solutions. In NTCIR-12, given a large repository of short text conversation data, the objective of Short Text Conversation (STC) is to build an IR system that can effectively reuses past comments to respond to a post [3].

As the task description mentioned, we took the idea of taking STC as an IR problem. Our method has two stages: first, we use Lucence system to retrieve 50 candidate answers for each given question. Then we rank the answers according to the matching score given by our models, keeping the top ten answers as final results. We mainly refer to paper [5], our work could be regarded as a simplified version of their original work with only some minor changes. As we can see, the official results again proved the effectiveness of this method.

It is worth mentioning that our efforts mainly focused on the second stage. That is to say, the information retrieval part automatically done by the Lucene system, we only care about whether the retrieved answers match the given question. In other words, we treated this task as a classification problem.

2. SYSTEMOVERVIEW

Figure 1 shows an overview of our system. First, we extract appropriate patterns based on the conversation repository. At the same time, Lucene retrieve candidate answers for each question. We then generate features for training set and test set according to patterns. Logistic Regression model and Multi Layer Perceptron were both tried in our experiments. We will discuss them in details in the following sections.

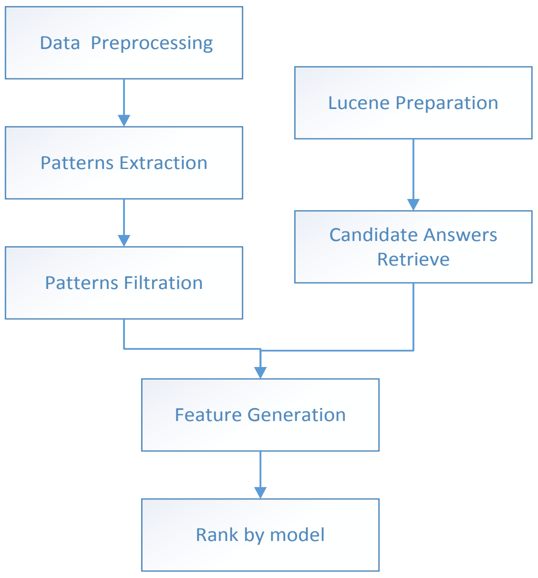

Figure 1: System Overview

3. LUCENE SYSTEM

The method that searching answers for each given test question in the repository of short text conversation data one

Table 1: Pattern examples

Pattern	Counts
like_China	2156
climb_Great Wall	1245
winter_snow	5412

by one would be most labor intensive and time consuming. So, we used the information retrieval system Apache Lucene to help us. As to similarity, we used the BM25Similarity score function to rank. For each question, we retrieved the most similar 50 questions in repository and collect their corresponding answer as candidate answers. Then we rank the answers according to the matching score given by our models, only keep the top ten answers in final estimation.

Actually, we also tried other number of candidate answers as well (20 and 100 etc.). After comparison on randomly selected sentence pairs according to human judgements, we decided to choose the number 50. When we judge the appropriateness of the retrieved responses, we mainly considered the four facets provided by the organizers: Logic Consistency, Semantic Relevance, Scenario Dependence and Repeating Opinions.

4. DATA AND EVALUATION METRICS

The data set used in this task comes from Weibo Corpus. The organizers provided a large repository of conversation data which include more than 5 million pairs and a additional training set contains thousands of pairs. Actually, as the size of the given training set is to small, it was not used in our experiments, we build training set based on the repository. In this task, the official metrics include: Mean nDCG@1, Mean P+, Mean nERR@10 More details about the data and metrics please refer to [4].

5. FEATURE ENGINEERING

In our approach, features are the generated *patterns*, examples are shown in Table 1 (Already translated into English).In the following, we will explain these patterns in detail.

5.1 Pattern Extraction

We removed English words, messy codes and punctuation in sentences as preprocessing.In order to generate these patterns, word segmentation is needed. In our experiment, we used the popular Chinese word segmentation tool JIEBA[1] to segment the sentences. . After tokenization, for each pair of matched sentences, we connect each token appeared in question and every token in corresponding answer with an underline. Though this, patterns were generated. This way, 369337081 patterns in total were generated.

5.2 Pattern Filtration

Apparently, not all of these patterns are appropriate, some patterns appeared only once, which makes it nearly impossible to be hit. Spelling mistakes, duplicates or some other reasons also make some patterns redundant and indiscriminate. So we want to get rid of these patterns, only turn suitable patterns to be features. But due to the limitation of the time, some simple operations were applied to filter patterns. We removed the patterns which contain Chinese stop words, this operation makes the number of patterns fell a great deal. Then, we removed the patterns appears only once. In the end, the total number of kept patterns is 13493613.

[1]http://lucene.apache.org/

[2]https://github.com/fxsjy/jieba/tree/jieba3k

5.3 Feature Generation

Once the filtered patterns were given, features are easy to generate. We generate the features according to the following steps:

1. Build a look-up dictionary.
2. For each pair of sentences, extract patterns contained in it.
3. For every extracted pattern, check the look-up dictionary, set one in the position if the pattern in it (same pattern appeared several times also regarded as one), otherwise zero.

After these steps, a very sparse feature vector was formed. Based on these feature vectors, another set of features were generated to feed Neural networks.

Due to the limitation of the computing resources, the number of features is so huge that our computer's memory can not load all the training examples (The configuration of our computer is: Intel Core i5 Processor, 16GB RAM, Microsoft Windows 7 Professional x64 SP1). So, we have a compromise proposal which can be regarded as a tradeoff between the number of features and the correctness of the hit patterns. To be specific, the strategy is that for every consecutive 50000 features, we form a new feature to replace them, which is the sum of these features' values. As to the number 50000, we also tried 10000, 20000, and 100000 respectively, and choosing the number in terms of the new features' size and performance. Intuitively, a non-zero new feature means this 'part' of old features are hit. On the other side, the larger the summation, the more possible the sentence is appropriate.

6. EXPERIMENTS

In the final estimation, we submitted three runs: ITNLP-C-R1, ITNLP-C-R2 and ITNLP-C-R3 respectively. The first two runs we used a logistic regression model to combine these features, while the ITNLP-C-R3 we employed a multiple layer perceptron model instead. We listed the results on Table 2.

Table 2: Results of our three submitted runs

RunID	nDCG@1	P+	nERR@10
R3	0.3067	0.4445	0.4186
R2	0.2900	0.4320	0.4123
R1	0.1033	0.2495	0.2354

6.1 Negative Example Generation

The only difference between the first two runs is the training data used to train the models is different while the features and model used in third model was different from the first two runs. Precisely, when creating training sets for each model, the proportions of negative examples and positive examples are different, as shown in Table 3.

To generate negative examples, we adopted the same methods proposed in [5]. We randomly select several responses

Table 3: Positive/Negtive Ratio

RUN-ID	**Positive-ratio**	**Negative-ratio**
ITNLP-C-R3	10.0%	90.0%
ITNLP-C-R2	16.7%	83.3%
ITNLP-C-R1	33.3%	66.7%

as negative examples. Since the repository is so large that there is even an outside chance that a randomly selected response is suitable. Considering the way we created them, there is another way to explain the percentages: in the data sets, each positive example was followed by 9/5/2 negative examples.

6.2 Logistic Regression Model

We use the Logistic Regression model with L2 regularization, and the regularization makes a big difference maybe for the reason that the features were so sparse. Actually, we compared the performance of L1 regularization and L2, with human evaluation on some randomly selected system output.Finally, we decided to use L2 regularization in our submission.

In terms of implementation, we used LIBLINEAR to do the prediction which is an open source library for large-scale linear classification. [2]

As mentioned before, we treated this task as classification problem, and then rank candidate answers according to their confidence of match.

6.3 Multiple Layer Perceptron model

We also tried one deep learning method, but the result is not very good. we used open-source deep learning framework theano to realize the function. [1] Because of the lack of experience, many parameters in the network were are randomly set without fine tuning. We implement the model based on multiple layer perceptron code on tutorial. The architecture of the network is shown in Figure 2

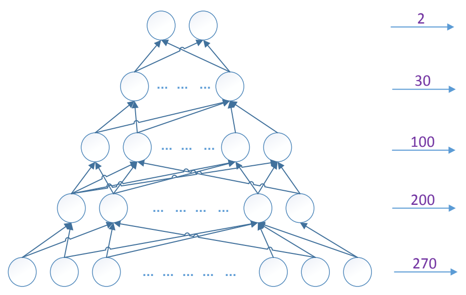

Figure 2: DNN Architecture

As the figure shows, we have 270 input nodes,200 units in the first hidden layer (tanh active function), 100 units in the second hidden layer, 30 in the third hidden layer, and two in the output layer.

7. CONCLUSIONS AND FUTURE WORK

Table 4: Parameter setting in MLP

Parameters	**Learning_rate**	**batch_size**
Values	0.01	100
Parameters	**n_epoches**	**patience**
Values	1000	400000
Parameters	**patience_increase**	**L2_reg**
Values	40	0.0001

In our work of NTCIR-12, we employed a simple logistic regression model to match past comments to respond to a post. The official results demonstrate the effectiveness of the solution. The patterns are the core of this method, extracting effective patterns would be the most important part in this solution. In addition, we can know from the experiment results that the positive/negative ratio in training set also matters a lot in final estimation. In failure analysis part, we found that our patterns can not accurately represent some sentences whose meaning depends on long-term dependency. We planed to handle this problem using results of dependency parser later. Due to the limitation of time, there were still many work to be done and many ideas needed to try in future. Our future work include: try some other more sophisticated methods to filter patterns. Append more complicated patterns (such as syntax-based patterns). Change the similarity metrics used in Lucene system. Directly represent the sentences in semantic space and combine these representations with deep learning.

8. ACKNOWLEDGMENT

The authors would like to thank the NTCIR-12 task organizers for their hard work. We also thank Yizhong Hu for his contribution on pattern extraction and data preparation.

9. ADDITIONAL AUTHORS

10. REFERENCES

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[3] L. Shang, Z. Lu, H. Li, and T. Sakai. Ntcir-12 pilot task: Short text conversation (stc). *Call for Participation to the NTCIR-12 Kick-Off Event*, 2015.

[4] 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*, pages 1–10, 2016.

[5] M. Wang, Z. Lu, H. Li, and Q. Liu. Syntax-based deep matching of short texts. *arXiv preprint arXiv:1503.02427*, 2015.

[3]https://www.csie.ntu.edu.tw/ cjlin/liblinear/
[4]http://deeplearning.net/software/theano/
[5]http://lucene.apache.org/