BUPTTeam at the NTCIR-13 STC-2 Task

Songbo Han
Beijing University of Posts and Telecommunications, China
hansongbo@bupt.edu.cn

Hao Zhu
Beijing University of Posts and Telecommunications, China
hzhu@bupt.edu.cn

Yongmei Tan
Beijing University of Posts and Telecommunications, China
ymtan@bupt.edu.cn

Abstract

This paper provides an overview of BUPTTeam's system participated in the Short Text Conversation (STC) task of Chinese at NTICR-13. STC is a new NTCIR challenging task which is defined as an information retrieval (IR) or natural language generation problem. In this paper, we propose a novel method to generate appropriate comments based on the following four steps: 1) preprocessing, 2) model building, 3) candidate comments generation, 4) candidate comments ranking. The evaluation results show that our methods finish the task successfully and have positive effect on improving the evaluation measurement.

Team name

BUPTTeam

Subtasks

Short Text Conversation (Chinese)

Keywords

Generation, Beam Search, Random Walk

1. Introduction

STC (Short Text Conversation) is a new NTCIR core task in NICIR-13, which aims at building a STC system. This system can generate new comments for each post based on the provided post-comment repository (Shang et al., 2017).

Compared with NTCIR-12 STC task, NTCIR-13 STC considers not only retrieval-based method but also generation-based method. For the retrieval-based method, the basic idea is maintaining a large repository of short text conversation data (i.e. post-comment pairs), and finding a clever way to retrieve related comments from the repository and return the most appropriate one (Shang et al., 2017). For the generation-based method, it can build a suitable model based on the large repository so that it can generate more fluent and reasonable new comments for each post. And it is also helpful for researchers to find new approaches to build human-computer conversation.

Generation-based methods for STC fall into two categories, 1) the statistical machine translation(SMT) method and 2) the RNN-based neural models (Shang et al., 2017). RNN-based neural models are neural network models, which no longer needs artificial features. Besides, it can map the source language into target language directly.

In this paper, we propose a novel method to generate new comments for the post based on the following specific steps: preprocessing, building the model, candidate comments generation and candidate comments ranking. We then show the effectiveness of the method in generating short texts.

Our contribution is twofold: 1) we propose a novel method to make the generated sentences more fluent by applying n-gram to Seq2Seq model (Bahdanau et al., 2014). 2) we put forward a new way to construct graph for candidates ranking to find the most appropriate comments for a new post.

2. System Architecture

The architecture of our STC system is described as Figure 1, which includes the following four components.

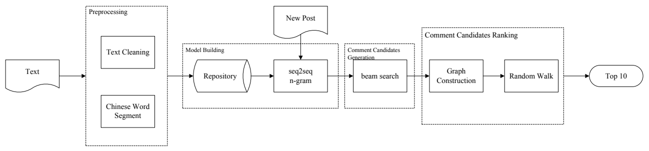

Figure 1: System Architecture

2.1 Preprocessing

There are some traditional Chinese characters, specific symbols, excess punctuations in the raw text. We convert traditional Chinese characters to simplified Chinese characters and remove the specific symbols and excess punctuations to clean the text.

Word is the smallest meaningful linguistic element which is capable of independent activity. As there is no clear distinction between the word marks (Zheng et al., 2013). Therefore, the segment for the Chinese words plays the key role in analyzing Chinese text. We use Stanford Word Segment[1] to split Chinese text into a sequence of words (Chang et al., 2008).

2.2 Model Building

The training data is crawled from the Sina WeiBo. It is rich in content and varied in style, so the sentence generated by the traditional Seq2Seq model is not fluent. To solve the problem, we propose a novel model to make the generated sentences more fluent by applying n-gram to Seq2Seq model. The model is described as Figure 2.

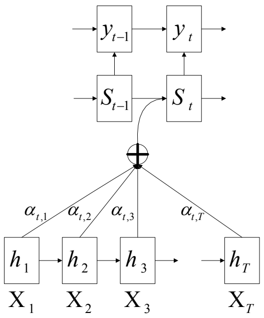

Figure 2: Model Architecture

In the Encoder-Decoder framework, an encoder reads the input sentence $x = (x_1, \ldots, x_{T_x})$, into a sequence of vector c. The most common approach is to use an RNN such that

$$h_t = \mathrm{f}(x_t, h_{t-1}) \quad (1)$$

[1] http://nlp.stanford.edu/software/segmenter.html

$$c = q(\{h_1, \dots, h_{T_x}\}) \qquad (2)$$

where $h_t \in R^n$ is a hidden state at time *t,* and *c* is a vector generated from the sequence of hidden states. f and q are some nonlinear functions. System uses a Long short-time memory(LSTM) as f and $q(\{h_1, \dots, h_{T_x}\}) = h_T$.

The decoder is often trained to predict the next word y_t given the context vector c and all the previously predicted words $\{y_1, \dots, y_{t-1}\}$. In other words, the decoder defines a probability over the translation y by decomposing the joint probability into the ordered conditionals:

$$P(y) = \prod_{t=1}^{T} P(y_t | \{y_1, \dots, y_{t-1}\}, c) \qquad (3)$$

where $y = (y_1, \dots, y_{T_y})$. With an RNN, each conditional probability is modeled as

$$P(y_i | \{y_1, \dots, y_{i-1}, x) = k(y_{i-1}, y_i) * g(y_{i-1}, s_i, c_i) \qquad (4)$$

where x is the input sentence, g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t,and s_t is the hidden state of the RNN. The k function introduces 2-gram, which can make the generated result more fluent while maintain the generation capability of the model.

$$k(w_i, w_j) = \frac{count_{w_j}}{count_{w_i w_j}} \qquad (5)$$

where $count_{w_j}$ is the frequency count of word w_j in the repository and $count_{w_i w_j}$ is the frequency count of phrase $w_i w_j$ in the repository.

The context vector c_t depends on a sequence of annotations $(h_1, \dots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sentence with a strong focus on parts surrounding the i-th word of the input post.

The context vector c_i is, then computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \qquad (6)$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{e^{e_{ij}}}{\sum_{k=1}^{T_x} e^{e_{ik}}} \qquad (7)$$

where

$$e_{ij} = \alpha(s_{i-1}, h_j) \qquad (8)$$

as the system use a feedforward neural network as α function.

2.3 Candidate Comments Generation

For a given new post, the system generates candidate comments using the Beam Search algorithm. The Beam Search algorithm is a heuristic algorithm based on the branch and bound method. It heuristically estimates the k best paths, and only searches down from the estimated k paths. Only satisfied nodes will be retained in each layer while other nodes were permanently abandoned, In this way, the running time will be reduced greatly.

The Beam Search algorithm is applied to generate the comment according to the following three steps. Here, we set beam size as *k*.

1) Generate the probability of the word at the time step *t*.

2) Calculate the probability of the generated sentences from the time step 1 to *t,* and the system chooses the last word in the top *k* maximum probability sentences as the input of the time step *t* + 1.

3) Repeat step 1.

2.4 Candidate Comments Ranking

Referent graph is a strongly connected graph represented by G= (V, E), where V is the set of all candidate comments of the new post and E is the set of all edges in the referent graph (Han et al., 2011).

To find the most appropriate comments for a new post, we use a referent graph-based approach for candidates ranking instead of directly getting the result based on the probability of candidate comment for a new post.

• Referent Graph Construction

Given a new post, the number of candidate comment is 20. Each edge is between these candidate comments or between the new post and the candidate comment, so there are two types of edges in Referent Graph. The weight of the edge between one candidate comment c_i and another c_j is the semantic similarity $SR(c_i, c_j)$ between candidates comment c_i and c_j defined as:

$$SR(c_i, c_j) = \frac{v(i).v(j)}{||(v(i)||*||v(j)||} \qquad (9)$$

where $v(c_i)$ and $v(c_j)$ is semantic vector of c_i and c_j respectively, which can be calculated as follows:

$$v(c_i) = \frac{1}{m} \sum_{k=1}^{m} w_{ik} \qquad (10)$$

$$v(c_j) = \frac{1}{n} \sum_{k=1}^{n} w_{jk} \qquad (11)$$

where w_{ik}, w_{jk} is pre-trained word2vector embedding.

The weight of the edge between one candidate comment c_i and the its post is defined as:

$$score(p, c_i) = P_{model}(c_i | p)$$

The transition probability matrix *T* on the graph *G* can be calculated as:

$$P(p \to c_i) = \frac{score(p,c_i)}{\sum_{c_i \in N_p} score(p,c_i)} \qquad (12)$$

$$P(c_i \to c_j) = \frac{SR(c_i,c_j)}{\sum_{c_j \in N_{c_i}} SR(c_i,c_j)} \qquad (13)$$

where N_p refers to a candidate comments set of a post p and N_{c_i} refers to the set of the new post which are adjacent with the candidate c_i.

• Ranking

The random walk original vector *α* on *G* is the vector of $|\boldsymbol{V}| \times 1$. After completing of the initialization of vector $\boldsymbol{\alpha}$, the normalization of vector $\boldsymbol{\alpha}$ can make sure that vector $\boldsymbol{\alpha}$ is a correct initialization vector.

Formula (14) and (15) illustrate the process of random walk with restart:

$$r^0 = \alpha \qquad (14)$$

$$r^{t+1} = (1 - \lambda) \times T \times r^t + \lambda \times \alpha \qquad (15)$$

where r^t refers to the intermediate result of random walk with restart, t refers to times of iteration, and λ refers to a parameter. Making $r^{t+1} = r^t$, eventual stationary distribution can be calculated as shown in formula (16):

$$r = \lambda (I - mT)^{-1} \alpha, m = 1 - \lambda \qquad (16)$$

For a post *p*, *E(c),* the effectiveness measure of a candidate comment *c* is defined as follows:

$$E(c) = score(p, c) \cdot r(c) \qquad (17)$$

Finally, candidate comments are ranked by *E(c)* and a ranking list of ten comments for a new post is acquired.

3. Experiments

3.1 Data Set

There are 219174 Weibo posts and the corresponding 4305706 comments in the repository.

There are 769 query posts and each has nearly 15 comment candidates in the training data. There are 11,535 post-comment pairs with "suitable", "neural", and "unsuitable" labels. "Suitable" means that the comment is clearly a suitable comment to the post and "neutral" means that the comment can be a comment to the post in a specific scenario, while otherwise the comment shall be labeled as "unsuitable".

There are 225 query posts and each of them has about 30 comment candidates in the training data. There are 6,017 post-comment pairs with "suitable", "neural", and "unsuitable" labels. "Suitable" means that the comment is clearly a suitable comment to the post, "neutral" means that the comment can be a comment to the post in a specific scenario, while "unsuitable" means it is not the two former cases.

100 posts are used for test. We are permitted to submit up to five runs for the task. In each run, a ranking list of ten comments for each test query is required

3.2 Evaluation Metrics

The evaluation metrics are nG@1, nERR@10 and P+ (SaKai et al., 2015).

nG@1 shows the quantity of effective result (such as L1, L2 result) in the retrieved candidates. L2 result will score as 1, the L1 result will score as 1/3, and the L0 result will score as 0.

The score of nERR@10 shows the rank correctness of the candidates ranking, which means that the more effective result should be ranked higher in the ranking list of retrieved candidates.

The score of P+ mostly depends on the position of the best effective result in the ranking list of retrieved candidates. It gives the top ranked result the most ratio.

3.3 Experimental Results

The best four teams with their best run results are shown in Table 1. The runs have been sorted by Mean nG@1, P+ and nERR@10, respectively.

Table 1 shows that our proposed methods are not ideal in the STC task. And there is still a lot of room for improvement in the nG@1 indicator.

Table 1: Part of Official STC results

Run	nG @1	Run	P+	Run	nERR @10
SG01-C-G1	0.58 41	SG01-C-G1	0.65 80	SG01-C-G1	0.7130
splab-C-G4	0.50 80	splab-C-G4	0.60 80	splab-C-G4	0.6492
srcb-C-G2	0.40 80	srcb-C-G2	0.51 88	srcb-C-G2	0.5781
TUA1-C-G4	0.38 93	TUA1-C-G4	0.49 09	TUA1-C-G4	0.5277
BUPTTe am-C-G1	0.18 23	BUPTTe am-C-G1	0.27 55	BUPTTe am-C-G1	0.2746

Table 2 shows the performance of our different runs in the task.

1) G1 generates sentence by applying n-gram based on Seq2Seq model. And the Beam Search algorithm is applied to the comment generated. And the random walk with restart is used to rank the candidate comments.

2) G2 generate sentence based on traditional Seq2Seq model. And the Beam Search algorithm is applied to the comment generated. And the random walk with restart is used to rank the candidate comments.

Table 2: Comparison of Performance on 2 Runs

Run name	nG@1	P+	nERR@10
BUPTTeam-C-G1	0.1823	0.2755	0.2746
BUPTTeam-C-G2	0.0933	0.1895	0.2001

As we can see from the table, G1, using n-gram based on Seq2Seq model, can improve the evaluation measurement significantly.

4. Conclusions

In this paper, we propose an approach for STC task of NTCIR-13.

We apply n-gram to improve the fluency of the generated generate results based on the Seq2Seq model, where the beam search is used to generate candidate comments and the random walk which is a graph-based method is used to rank candidate comments. The evaluation results show that our method finish the task successfully and have positive effect on improving the evaluation measurement.

5. References

[1] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer Science, 2014.

[2] Daisuke Ishikawa, ASURA: A Best-Answer Estimation System for NTCIR-8 CQA Pilot Task, 2010.

[3] Daisuke Ishikawa, Tetsuya Sakai, Noriko Kando. Overview of the NTCIR-8 Community QA Pilot Task: The Test Collection and the Task. NTCIR-8 Workshop Meeting, 2010.

[4] Kazuko Kuriyama. Best-Answer Selection Using a Machine Learning Tool at NTCIR-8 CQA Pilot Task, 2010.

[5] Karl Pearson. The Problem of the Random Walk. Nature, 1905, 268: 2113–2122.

[6] Leo Grady. Random walks for image segmentation. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2006, 28(11): 1768–1783.

[7] Lifeng Shang, Tetsuya Sakai, Zhengdong Lu, Hang Li, Ryuichiro Higashinaka, and Yusuke Miyao. 2016. Overview of the NTCIR-12 Short Text Conversation Task. In Proceedings of NTCIR-12. 473–484.

[8] Lifeng Shang, Tetsuya Sakai, Zhengdong Lu, Hang Li, Ryuichiro Higashinaka, Yusuke Miyao, Yuki Arase. 2017. Overview of the NTCIR-13 Short Text Conversation Task. In Proceedings of NTCIR-13.

[9] Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. Optimizing Chinese word segmentation for machine translation performance, ACL, 2008.

[10] Tetsuya Sakai, Lifeng Shang, Zhengdong Lu, and Hang Li. Topic Set Size Design with the Evaluation Measures for Short Text Conversation, 2015.

[11] Xianpei Han, Le Sun and Jun Zhao. Collective entity linking in Web Text: a graph-based method. Proceedings of International Conference on Research & Development in Information Retrieval. Beijing, 2011.

[12] Xiaoqing Zheng, Hanyang Chen and Tianyu Xu. Deep Learning for Chinese Word Segmentation and POS Tagging. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013.

[13] Young-In Song, Jing Liu, Tetsuya Sakai, Xin-Jing Wang, Guwen Feng, Yunbo Cao, Hisami Suzuki and Chin-Yew Lin. Microsoft Research Asia with Redmond at the NTCIR-8 Community QA Pilot Task, 2010.

[14] Yongmei Tan, Mingda Wang and Songbo Han. BUPTTeam Participation in NTCIR-12 Short Text Conversation Task, 2016.

[15] Zhaochen Guo, Denilson Barbosa. Robust Entity Linking via Random Walks. Proc. CIKM, 2014.