Introduction

We participate in NTCIR-13 Short Text Conversation (STC) Chinese subtask. In our system, we use the retrieval-based method and the generation-based method respectively. We have achieved top performance in both methods with 8 submissions.

Retrieval-based Method

In this part, we treat STC as an IR problem. We separate the process into stages, as it goes, we reduce the candidate set and introduce more complex features. In the end, we use learning to rank to get the final result list. Figure 1. describes the process of our retrieval-based method.

Generation-based Method

In our generation-based method, we first generate various candidate comments, then perform ranking on them to get a preferable top 10 results. Figure 2. shows our generation-based method.

Figure 1. Diagram of Retrieval-based Method

Stage1: Retrieval Stage

At the beginning, we do data pre-processing to remove some low-quality post-comment pairs, then we put the repository into a light-weighted search engine, treating the post like a title and the comment like content.

For a given query, we retrieve 500 post-comment pairs from the repository for further comment selection.

Traditional features in IR are used in this step, such as BM25, MRF for term dependency, Proximity, etc. These features will be used in the final stage.

Stage2: Ranking Stage I

In this stage, we employ features designed for STC task:

- cosine similarity of TF-IDF Vector between:
- negative Word Mover Distance [M. J. Kusner 2015]

where query = post
query + comment
query + post + comment

- Translation based language model [Z. I. 2014]

We treat each feature as a ranker, simply add the sequence number to get a final rank, we keep the top 50 candidates.

Stage3: Ranking Stage II

We employ some DNN features to better capture rich structure in STC problem:

- Scoreemb
- ScoreBiLSTM-CN( ) [R. Yan 2016]

Trained with a ranking-based objective, using given repository plus extra 12 million crawled post-comment pairs, noted as RepBase.

ScoreBiLSTM-CN = Defined in Generation-based Method

At last, we use LambdaMART to perform learning to rank, all the features aforementioned will be used. The training data are 40 thousands, labeled pairs. For each given query, we keep top 10 pairs’ comments as the final result.

Generator-based Method

We design 4 generative models to generate candidate comments, models are trained with RepBase, corpus is pre-processed by rules before training.

- S2SAttn
  - addmem ( )
  - Add dynamic memory to the attention
- VAEAttn
  - Use Variational Auto-Encoder
  - addmem

Rank the Candidates

We define likelihood and posterior to rank the candidates. For a post X and a generated post Y, we define ScoreLogP, as a prediction of logarithmic P(X|Y), known as likelihood. We sum up likelihood scores from different models and implementations, noted as Li. As for posterior, we make the prediction P(X|Y); so we have ScoreLogP and Po. We combine them in the following way to get the final ranking score:

\[
\text{score} = \frac{1}{1 + e^{-X}} \times P_{\text{Likelihood}}(X\mid Y)
\]

where P_{\text{Likelihood}}(X\mid Y) \geq v_{\text{Likelihood}}(Y, Wu 2016).

Before ranking, we also process the comments by rules to make them more fluent and to remove improper comments.

Case Study

We show some from our generation-based-method submissions in Table 3. and Table 4. to reveal how improvements on baseline models benefit candidate generation and ranking.

Analysis & Conclusions

On average, VAE does worse than traditional seq2seq, but it can bring in interesting candidates. The feature Po works, giving higher rank to more informative candidates. Fusion of models do better than single model, because the ranking will bring preferable candidates to top 10.

According to the evaluation results, the generation-based method does better, however, it still prunes to generate “safe” responses. Meanwhile, the retrieval-based method tends to get in-coherent comments. We also find that larger size of training data will help a lot.

References