CUIS Team for TREC 2017 CAR Track

Xinshi Lin
The Chinese University of Hong Kong
Hong Kong
xslin@se.cuhk.edu.hk

Wai Lam
The Chinese University of Hong Kong
Hong Kong
wlam@se.cuhk.edu.hk

Deepanway Ghosal
Indian Institute of Technology Patna
Patna, India
deepanway.me14@iitp.ac.in

ABSTRACT
We participated in the Complex Answer Retrieval (CAR) track at TREC 2017. Our ranking system scores each candidate passage via a slightly modified sequential dependence model that restricts the generation of ordered and unordered bigram queries. Our best submission achieves map of 0.1716 and recip_rank of 0.3930 in the passage ranking task.

Keywords
TREC, Sequential Dependence Model, Passage Ranking, Entity Ranking, Dirichlet Smoothing

Team Name
CUIS

Track
Complex Answer Retrieval

1. INTRODUCTION
The objective of TREC CAR 2017 Complex Answer Retrieval Track[1] is to encourages research for answering more complex information needs with longer answers. Its focus is on developing systems that can answer complex information needs by collating relevant information from an entire corpus.

Our CUIS team participated in both passage ranking task and entity ranking task. In the passage ranking task, participants are required to retrieve for each of its sections given an article stub, a ranking of relevant entity-passage tuples (E,P). The passage P is taken from a provided passage corpus. The entity E refers to an entry in the provided knowledge base. In the entity ranking task, the entity must be given, and optionally, complemented with a passage that serves as provenance for why the entity is relevant. If the passage is omitted, it will be replaced with the first paragraph from the entity's article. A passage or entity is defined as relevant if the passage content or entity should be mentioned in the knowledge article.

One challenge for TREC CAR is that the query has a structure of sections while current retrieval systems can only provide solutions to phrase-level retrieval for simple fact and entity-centric needs. To model section dependencies, we employ sequential dependence model [1] as a starting point with restrictions on its generation of bigram queries concerning sections. We propose a method that transforms passage ranking results into entity ranking results through establishing a mapping between paragraph identifier and article identifier with the help of a modified version of TREC CAR Tools.

2. Our Approach
Our ranking system is composed of three main components. The first component preprocess a query. The second component indexes corpus provided by the organizer for training and testing. The third component choose top K candidate paragraphs in the corpus via BM25 and rank each candidate with sequential dependence model. And finally, we transform passage ranking results into entity ranking results.

2.1 Query Preprocessing
For each query stub q with n sections $H_1/H_2/\ .../H_m$. For each section, a list of query terms L_i are generated after stop word removal and stemming. Generation of unigram queries $T(q)$, ordered bigram queries $O(q)$ and unordered bigram queries $U(q)$ for sequential dependence model can be expressed by following formulas:

$$T(q) = \bigcup_{j\in\{1..m\}} \{t_i : t_i \in L_j\}$$

$$O(q) = \bigcup_{j\in\{1..m\}} \{t_i t_{i+1} : t_i t_{i+1} \in L_j\}$$

$$U(q) = \bigcup_{i,j\in\{1..m\}, i\neq j, |i-j|<3} \{t_x t_y : t_x \in L_i, t_y \in L_j\}$$

As shown in the above formulas, we only consider ordered bigram queries formed of query terms from the same section and unordered bigram queries formed of query terms from different sections whose distance is less than 3. It aims to reduce the computational complexity of our retrieval model since a query for complex answer retrieval task may be very long.

2.2 Indexing
As main corpus for training and testing, a preprocessed English Wikipedia dump provided by the organizer was automatically indexed for passage ranking and entity ranking. The official corpus consisits of a paragraph corpus and an article corpus from 50% of Wikipedia that match the official selection criterion[2].

The corpus stores data in CBOR format. We can automatically extract paragraphs from paragraph corpus with the help of TREC CAR Tools[3], the official support tool for TREC CAR track. The index contains two fields, "para_id" and "content", which stores the identifier of a paragraph and its textual content respectively.

2.3 Ranking Passages and Entities
Passage Ranking
Given a query stub q, we score each paragraph d via a two-phase strategy. First, we use q as a query string to retrieve top K

[1] http://trec-car.cs.unh.edu

[2] http://trec-car.cs.unh.edu/process/dataselection.html

[3] https://github.com/TREMA-UNH/trec-car-tools

candidate paragraphs via BM25. Then we use sequential dependence model to each candidate paragraph. The relevance score is defined by the following formulas:

$$score(q,d) = \lambda_T \sum_{t \in T(q)} \log(p(t|d)) + \lambda_O \sum_{t \in O(q)} \log(p(t|d)) + \lambda_U \sum_{t \in U(q)} \log(p(t|d))$$

$$p(t|d) = \frac{TF(t,d) + \mu \cdot p(t|C)}{|d| + \mu}$$

$$p(t|C) = \frac{TF(t,C)}{|C|}$$

where C is the collection of paragraphs, $TF(t,d)$ is the frequency of the term t in the paragraph d, μ is the average document length in the collection, and $|C|$ is the sum of paragraph length for each paragraph in C. λ_T, λ_O and λ_U are parameters of the model. Compared to original sequential dependence model, our model differs on the generation of bigram queries set $O(q)$ and $U(q)$ as mentioned in section 2.1.

Transformation from Passage Ranking Results into Entity Ranking Results

Once we obtain results for the passage ranking task. We can transform it into entity ranking results via the mapping between paragraph identifiers and article identifiers. For each query, we simply replace paragraph identifiers by its mapped article identifiers in the ranked list of paragraphs and filter out identifier articles.

Our ranking system also need to find which article that a certain paragraph come from. To establish mapping between a paragraph identifier and an article identifier, we need to make use of train.fold(0-4).cbor provided in the article corpus, which stores structure information of each Wikipedia article including paragraphs it contains. We modify read_data.py in TREC CAR Tools. When each article p is being parsed by calling function *iter_annotations*, we make the method *from_cbor* of class *PageSkeleton* prints para_id, an attribute of *Paragraph.from_cbor*(*cbor*[1]) under condition "tag==1" to a new file. And our program also prints article identifier (*p.page_id*) to this file after the article is parsed. Finally, we can analyze the file and find a mapping.

3. Experiments

3.1 Experiment Setup

We use Pylucene to index the corpus. For each query, we use Lucene's default BM25 search engine to retrieve top 1000 most relevant paragraphs as candidate paragraphs for the second phase scoring. The model was trained by performing a grid search for parameters λ_T, λ_O and λ_U to optimize the overall map on training dataset.

As stated by the organizers, both an automatic binary and a fine-grained manual evaluation procedure are enacted on 40 test stubs. The annotators on manual evaluation had to choose among the following options:

3: must be mentioned;

2: should be mentioned;

1: can be mentioned

0: roughly on topic;

-1: non-relevant;

-2: trash

Mean average precision (map), R-precision (Rprec) and reciprocal rank (recip_rank) are used as the evaluation metrics.

3.2 Experiment Results

We submitted a passage ranking run and an entity ranking run. We receive results for the passage ranking run. As shown in Table 1, the column "Method" shows the name of each evaluation method. The column "map", "Rprec" and "recip_rank" report Mean average precision, R-precision and reciprocal rank for each run respectively.

4. Conclusions

We have presented the participation of our passage ranking and entity ranking system in the TREC 2017 CAR track. We develop a ranking system that scores each passage via a slightly modified sequential dependence model. We make use of official support tool to establish a mapping between paragraph identifiers and article identifiers to obtain entity ranking results. Experiments show that our system obtains reasonable results for the formal run topics.

One of the future work is to exploit more structure information in the Wikipedia for the relevance measure.

Table 1. Evaluation Results for Passage Ranking Run

Method	map	Rprec	recip_rank
Automatic	0.1300	0.1008	0.1963
Lenient	0.1949	0.2711	0.4895
Manual	0.1716	0.1861	0.3930

5. REFERENCES

[1] Metzler, D. and Croft, W.B., 2005, August. A Markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 472-479). ACM