UDEL/SMU at TREC 2009 Entity Track

Wei Zheng[1], Swapna Gottipati[2], Jing Jiang[2], Hui Fang[1]

[1] Department of Electrical and Computer Engineering
University of Delaware

[2] School of Information Systems
Singapore Management University

Abstract

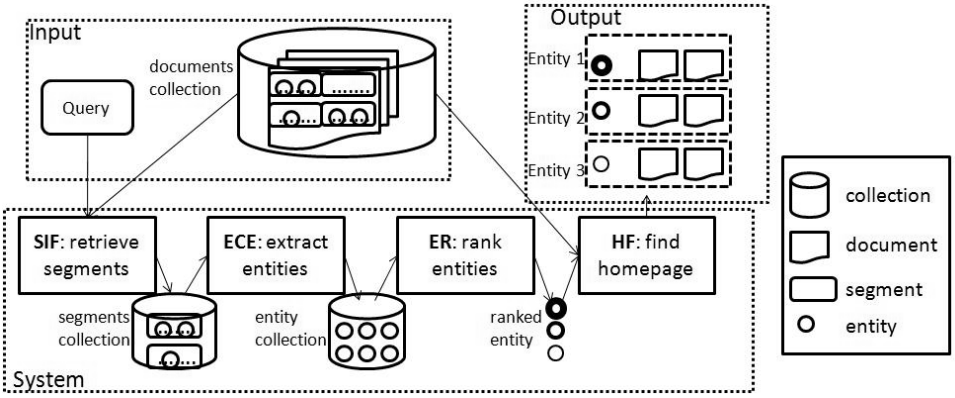
We report our methods and experiment results from the collaborative participation of the InfoLab group from University of Delaware and the school of Information Systems from Singapore Management University in the TREC 2009 Entity track. Our general goal is to study how we may apply language modeling approaches and natural language processing techniques to the task. Specifically, we proposed to find supporting information based on segment retrieval, to extract entities using Stanford NER tagger, and to rank entities based on a previously proposed probabilistic framework for expert finding.

1 Introduction

The InfoLab from University of Delaware and Singapore Management University collaborated in the TREC 2009 Entity track. The task is to find a list of target entities and their homepages, given a query entity, its relations with the target entities, and the specific entity type of the target entities [1]. Our general goal is to study how we may apply language modeling approaches and natural language processing techniques to the task. Specifically, we focus on the following four challenges: (1) how to find supporting information that is useful to extract a target entity according to a query; (2) how to extract entities based on the type specified in the query; (3) how to rank entities based on the supporting information; and (4) how to locate the homepages of the target entities.

First, to capture the intuition that the supporting information for relations between two entities often spans a short distance, we proposed to segment the documents into fixed-length paragraphs, and ranked the paragraphs based on the relevance to the query using KL divergence retrieval method. Second, to extract the target entities, we used an off-the-shelf NER tagger (the Stanford NER tagger) to find person and organization entities. For product entities, we

Figure 1: design of our Entity track system



used a set of heuristic rules to extract phrases that are likely to be target entities. We extracted the candidates of target entities from the relevant paragraphs that contain the topic entity and the relation narrative keywords, as retrieved in the previous step. Third, to rank the target entity candidates, we adapted the general language modeling framework for expert finding proposed in a previous study [4]. It has been shown that the proposed methods perform well on TREC Enterprise track. Thus, it is interesting to evaluate the proposed framework in the context of the entity track. Finally, we perform standard retrieval to locate the homepages of the target entities.

The paper is organized as follows. We propose a general strategy for entity retrieval in Section 2, and describe implementation details in Section 3. We then report the experiment results in Section 4 and conclude in Section 5.

2 A General Strategy for Entity Track

Queries used in the entity track are structured. The queries specify the type of target entities, a query entity, URL of the query entity, the type of target entities and the relation between the query entity and target entities. To retrieve the target entities and their homepages based on the queries, we propose a general strategy for entity retrieval aiming to solve four challenges: (1) how to find supporting information that is useful to extract a target entity according to a query; (2) how to extract entities based on the type specified in the query; (3) how to rank entities based on the supporting information; and (4) how to locate the homepages of the target entities.

Figure 1 shows the system architecture of the proposed general strategy. It can be divided into four steps:

1. **Retrieving supporting information:** Given a query entity and a rela-

tion specified in a query, it is necessary to retrieve the supporting information that can be used to extract target entity. One possible solution is to retrieve information relevant to both the query entity and the specified relation. The underlying assumption is that if the supporting documents contain are related to both the query entity and the relation, they would be likely to contain target entity candidates, i.e., the entities related to the query entity in the specified way.

2. **Extracting candidate entities:** Given the supporting information, the next step is to extract candidate entities based on the type specified in the queries. Named Entity Recognition (NER) classifier can be applied to extract entity candidates based on the specified type.

3. **Ranking candidate entities:** After extracting the candidate entities from supporting information, we need to rank the candidates based on how likely they are target entities based on the query. It is necessary to use the supporting information as a bridge to connect the target entities and the information specified in the queries. Note that the problem of ranking entities is very similar to the problem of expert finding [2], whose goal is to find persons with expertise specified in a query. Thus, we could adapt the methods proposed for expert finding problem to solve the entity ranking problem.

4. **Finding homepages of the entities:** For every candidate, we need to find its corresponding homepage. This step is essentially similar to the named page finding task in the previous Web tracks.

3 Our Proposed Methods

We proposed a general strategy for entity track in the previous section. We now discuss our proposed methods for each of the four steps in details.

3.1 Retrieving Supporting Information

The first step is to retrieve supporting information that contains the target entities based on the information specified in the queries. We propose to construct keyword queries based on the narrative fields of the original queries, and then use the constructed queries to retrieve supporting information. The underlying assumption is that the supporting information would be likely to contain the target entities if it contains both the query entity and the relations between the query entity and target entities,

One decision we need to make in the process is which text unit we should retrieve as the supporting information. On the one hand, the text unit should be long enough to include both the query entity and target entity as well as their relations. On the other hand, the text unit should not be too long because longer text units may contain more noisy entities that are not the ones relevant

to the queries. For example, query 6 in the entity track is "organizations that award Nobel prizes." The Wikipedia page of Nobel Prize discusses not only organizations that award the prize but also the organizations that mint the prize medal. The supporting information would include both the organizations awarding Nobel Prizes and the organizations minting the medal if the text unit is too long. To balance the tradeoff, we use fixed length segments, i.e., 50 words or 100 words, as the units for the supporting information in this work. To improve the efficiency, we first use the narrative field of the query to retrieve 1000 documents for each query. Each retrieved document is split into segments. Finally, the segments are ranked by the query and the top-ranked segments are selected as the supporting information.

The retrieval function used in this step is the Dirichlet prior retrieval method [3]:

$$S(q,d) = \sum_{w \in q \cap d} c(w,q) \times ln(1 + \frac{c(w,d)}{\mu \times p(w|C)}) + |q| \times ln(\frac{\mu}{|d| + \mu}) \qquad (1)$$

where q is the query, d is the document or segment, $S(q,d)$ is the score of d in q, w is the term that occurs both in q and d, $c(w,q)$ is the term frequency of w in the query, C is the collection of the d, $p(w|C)$ is the probability of the term given the collection language model, $|d|$ is the length of d, $|q|$ is the length of q, μ is the parameter and set to 1500 in our experiment.

3.2 Entity Extraction

This step extracts the entity candidates from the supporting segments. Entity Track focuses on three types of target entities, namely, person, organization, and product. For person and organization, we directly apply the Stanford NER tagger to extract named entities of the specified type from the top-ranked segments retrieved for each topic. Stanford NER Classifier [5] is a named entity recognizer that can label the following types of entities: PERSON, ORGANIZATION, and LOCATION. It uses the Conditional Random Field (CRF) sequential model to identify named entity tokens in sentences. The CRF classifier is trained on data from CoNLL, MUC6, MUC7, and ACE. For product entities, however, Stanford NER tagger is not able to recognize them, because the annotated training corpora (CoNLL, MUC and ACE) do not contain product entities. In fact, we are not aware of any corpus that has product entities annotated. To solve this problem, initially we planned to treat proper nouns as candidate product entities and then exclude proper nouns that are tagged as persons, organizations and locations by the Stanford NER tagger. However, after some preliminary experiments, we found that true product entities could also be tagged as person, organization or location by the Stanford NER tagger, while treating all proper nouns as candidate product entities produced too much noise. In the end, we decided to use the named entities tagged by the Stanford NER tagger as either person, organization or location as candidate product entities.

After observing the extracted entities, we find that some different names of the query entity are recognized as the possible target entities and their existence

will make a negative impact to the next step. Therefore, we define the following heuristics to reduce the noise: the supersets of the query entity terms are removed from the candidate list. Our assumption is that the names of the query entity may have different variations and the entity candidates that contain all the terms of the query entity are probably alternative names of the query entity. For example, entity candidate "new blackberry" is not a target entity, i.e., "carriers that blackberry makes phones for". Instead, it is clearly more related to the query entity "blackberry" than the target entity. This heuristic can delete the irrelevant candidates. However, it may also delete some relevant entities. For example, for query "Campuses of Indiana University," the entity candidate "Indiana University East" is deleted because it contains both terms in the query entity "Indiana University," but it is indeed one of the target entities. Clearly, it remains unclear what is a good strategy to filter out noisy candidates, and we leave this as our future work.

3.3 Entity Ranking

This step ranks the candidate entities based on the query and supporting information. The main challenge is how to utilize the supporting information to rank the candidate entities based on how likely they are relevant to the query. The problem of entity ranking is very similar to expert finding [2], since both problems need to rank entities or experts based on the supporting information. Previous work [4] proposed two generative models for expert finding, i.e., candidate generation model and topic generation model. In this work, we focus on apply these two models to rank entities based on the supporting information. We now briefly review these two models.

3.3.1 Candidate generation model

The candidate generation model assumes that the candidate entity is generated by the model of its relevant topics. Formally, it computes the ranking scores of candidate entities as follows [4]:

$$f(e,q) = \sum_{s \in S} p(e|s, R=1) \times p(q|s, R=1) \tag{2}$$

where e is a candidate entity, q is a query, R is a binary variable denoting the relevance (0 denotes irrelevant and 1 denotes relevant), s is a supporting segment and S is the collection of all the segments. The probability $p(e|s, R=1)$ and $p(q|s, R=1)$ can be computed with the Dirichlet prior retrieval function as shown in Equation (1). The candidate generation model utilize the co-occurrences of entity e and query q in the supporting segments to compute the relevance score of entity e. Each supporting segment contributes to the score of e by the probabilities that it is relevant to q and that it mentions e. The more relevant the segment is to both t and e, the more it can contribute to the score of e.

3.3.2 Topic generation model

The topic generation model assumes that the topic is generated by the probabilistic model of the relevant entities. The candidate entities can be ranked based on the function shown as follows [4]:

$$f(e,q) = P(e|S) \times \sum_{s \in S} (p(q|s, R=1) \times \frac{p(e|s, R=1)}{\sum_{s' \in S} p(e|s', R=1)}) \quad (3)$$

where $P(e|S)$ is the prior probability of candidate entity e in the segment collection S and β is a parameter whose value is set to be 6 according to the values reported in the previous study [4]. The entity prior could be either uniform or computed based on the entity occurrences, such as $\frac{count(e,S)}{count(e,S)+\beta}$, where $count(e,S)$ is the count of candidate entity e in the segment collection S.

There are two main differences between the candidate generation and topic generation models in Equation (2) and (3). The first one is that topic generation model can integrate the entity prior, i.e., $P(e|S)$ into the formula. The other one is that topic generation model can normalize the weight of each entity candidate by the sum of its probability in all segments, which is similar with the idea of idf.

3.4 Homepage Finding

The last step of Entity track is to find the homepages of the result entities. In this work, we simply use every entity as a keyword query and retrieve the documents using Dirichlet prior methods as shown in Equation (1). The Entity track requires that one document can only be returned as the homepage of one entity in the same topic. Therefore, we keep the document as the homepage of the entity where it has highest relevance score if the document has been returned as homepage of multiple entities.

The Entity track also requires the system to return the supporting documents for each entity. For each entity, we take the sum of the scores of all segments in one document as the score of that document and return the documents with highest scores as the supporting documents.

4 Experiment Results

4.1 Official runs

We submitted the following four official runs in the entity track. They mainly differ in three aspects: (1) whether candidate or topic generation model is used for entity ranking; (2) whether the supporting segments are based on 50 or 100 words; (3) whether a non-uniform prior is used in the topic generation model. The description of the official runs are summarized as follows.

1. **UdSmuCM50:** This run uses 50 words as supporting segments and uses the candidate generation model to rank candidate entities.

Table 1: Results of our submitted runs

	UdSmuCM50	**UdSmuCM**	**UdSmuTU**	**UdSmuTP**
#rel	191	192	189	186
#pri	8	10	11	9
NDCG@R	0.0809	0.0698	0.0611	0.0729

2. **UdSmuCM:** This run uses 100 words as supporting segments and uses the candidate generation model to rank candidate entities.

3. **UdSmuTP:** This run uses 100 words as supporting segments and uses the topic generation model with non-uniform prior to rank candidate entities.

4. **UdSmuTU:** This run uses 100 words as supporting segments and uses the topic generation model with uniform prior to rank candidate entities.

4.2 Results

In this section, we show the results of our systems and analyze the effectiveness of each step. There was a bug in our submitted runs. In the submitted runs, we retrieved 1000 documents by each result entity. The same document may be returned in different entities. We kept the document in the entity where it had the highest relevance score and deleted the duplication from the returned documents of all other entities in the same topic. However, each entity only needs to return three normal homepages and one wikipedia homepage. We only need to avoid the same document to occur in the top 3 normal documents and top 1 wikipedia document, instead of all returned documents, of different entities. After fixing the bug, we only delete duplication from all top 3 normal documents and top 1 wikipedia document of all the entities. That is why the results in this paper are different from the official reported results.

The normalized discounted cumulative gain at rank R (NDCG@R) is the main measure of the official results. Table 1 listed the NDCG@R, the number of relevant retrieved entities and the number of primary retrieved entities in our four runs. In segments with 100 words, the topic generation model with non-uniform prior performed best. The reason is that it uses the occurrences of entities in relevant information to measure the quality of the entities and that can help to select target entities from the entity candidates. For the segments in different length, the segments of length 50 performed better than those of length 100 in candidate generation model. The reason is that the small length will add more strict proximity constraints into the model and the entities co-occuring with the query in the same segment are more likely to be discussing the same topic.

In the following experiments, we analyze the effectiveness of each part by evaluating the entity names selected in each step. The method used in the following experiments is the topic generation model with non-uniform prior because it performed best in all the generative models.

Table 2: NDCG@R of our runs using different collections for entity ranking

	UdSmuCM	**UdSmuTU**	**UdSmuTP**
Seg	0.0506	0.051	**0.0643**
Doc_ret	0.0727	0.0778	**0.0894**
Doc_collection	0.0920	**0.1118**	0.1049

First of all, we evaluate how many percentage of target entity names were selected in each step. The recall values of target entities in the supporting information, entity candidates and result entities were 85.5%, 67.5% and 37.5% respectively. It showed that the quality of supporting information was good since less than 15 % of target entities were missed. But there were some problems in entity extraction step and entity ranking step because the recall dramatically decreased in these two steps.

After analyzing the results of entity extraction, we found that there were mainly three kinds of mistakes in the extracted entities. First, some entities were identified in the wrong types. For example, the organization *Bowmore* was extracted as the location. Second, the NER classifier made mistakes in identifying the boundary of the entities. For example, the *Philadelphia Athletics Philadelphia White stockings Philadelphia Centennials* was extracted as one entity. Third, some entities were entirely missed. The main reason is that their first letter is non-capitalized.

In the entity ranking step, the main problem was that we used the same segments collection to extract entity candidates and rank entities. The entity extraction and entity ranking steps were both based on the relevant segments. Therefore, all the extracted entities would have high scores in entity ranking and it was difficult to select the target entities. Table 2 showed the NDCG@R values of our systems when using different collections in entity ranking. *Seg* are the systems using segments collection in entity ranking, which are our submitted runs. *Doc_ret* uses the documents in entity ranking and the documents are retrieved by the entity candidates from the document collection of the documents returned by all the original queries. *Doc_collection* uses the documents retrieved by the entity candidates from the Category B collection. The results showed that the *Doc_collection* performed best because the documents are only related to the entities but are not neccessarily related to the original query. The score of the noisy entities would decrease since their relevant documents would have lower relevance score with the original query. Therefore, it can distinguish the target entities from the noisy entities. Another interesting observation is that the topic generation model with non-uniform prior performed worse than the topic generation model with uniform prior in *Doc_collection*. The reason is that the returned documents by entity candidates may be non-relevant to the original query and we cannot measure the quality of the entities by their occurrences in these documents.

5 Conclusion

This is the first year of the entity track. It requires the system to find the names, homepages and supporting document of the related entities. To solve this problem, we propose a general strategy with four steps, i.e., finding supporting information, extracting entities, ranking entities and finding homepages. Specifically, we use segment retrieval to find supporting information, apply Stanford NER classifier to extract candidate entities from the supporting information, adapt two generative models (i.e., candidate generation model and topic generation model) for expert finding to rank the candidate entities, and then use the language modeling approaches to find the homepages. The results show that the topic generation model performs better than the candidate generation model in the entity ranking step and short segments can perform better than long segments because they can combine more strict proximity constraints into the models.

References

[1] Guideline of TREC 2009 Entity Track, http://ilps.science.uva.nl/trec-entity/guidelines/

[2] I. Soboroff, A. P. de Vries, and N. Craswell. Overview of the trec 2006 enterprise track. In *Proceedings of TREC-06*, 2007.

[3] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR-01*, 2001.

[4] H. Fang and C. Zhai. Probabilistic models for expert finding. In *Proceedings of the 29th European Conference on Information Retrieval*, 2007.

[5] Vijay Krishnan and Christopher D. Manning. An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006.