Information Processing and Management 49 (2013) 995–1007

Contents lists available at SciVerse ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

An exploration of ranking models and feedback method for related entity finding

Xitong Liu *, Wei Zheng, Hui Fang

Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA

A R T I C L E   I N F O

Article history:
Received 14 October 2011
Received in revised form 15 March 2013
Accepted 21 March 2013
Available online 23 May 2013

Keywords:
Entity retrieval
Feedback model

A B S T R A C T

Most existing search engines focus on document retrieval. However, information needs are certainly not limited to finding relevant documents. Instead, a user may want to find relevant entities such as persons and organizations. In this paper, we study the problem of *related entity finding*. Our goal is to rank entities based on their relevance to a structured query, which specifies an input entity, the type of related entities and the relation between the input and related entities. We first discuss a general probabilistic framework, derive six possible retrieval models to rank the related entities, and then compare these models both analytically and empirically. To further improve performance, we study the problem of feedback in the context of related entity finding. Specifically, we propose a mixture model based feedback method that can utilize the pseudo feedback entities to estimate an enriched model for the relation between the input and related entities. Experimental results over two standard TREC collections show that the derived relation generation model combined with a relation feedback method performs better than other models.

© 2013 Elsevier Ltd. All rights reserved.

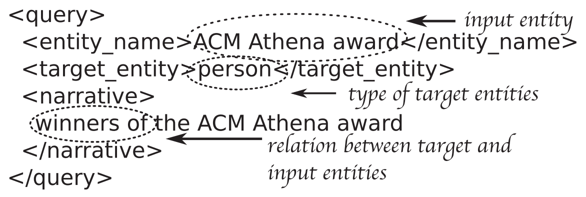
1. Introduction

Traditional information retrieval models have mainly focused on finding documents relevant to an information need. However, information needs are certainly not limited to relevant documents. For example, a consumer may want to find cell phone carriers selling iPhones, or a conference PC chair may want to find potential reviewers who are knowledgeable in certain research areas. Although users could use existing IR systems to find relevant entities by going through every returned document and manually identifying the entities, the whole process would be labor-intensive and time-consuming. Thus, it is necessary to study how to automatically retrieve entities, such as persons and organizations, that are relevant to a given query.

The problem of *related entity finding* is finding entities related to the information specified in a query. Unlike keyword queries used in document retrieval, an entity-related query needs to specify both the type and the relevance criteria of the target entities. Following the problem setup in the TREC Entity Track (Balog, de Vries, Serdyukov, Thomas, & Westerveld, 2010), a query for the related entity finding task is formulated as a structured query, which specifies an input entity, the type of the related entities to be found, and the relation between the input and the related entities. The goal is to find the related entities, which have the specified type and relation with the input entity, from a supporting document collection. For example, a user who wants to find ACM Athena award winners may formulate a query as shown in Fig. 1, and this query specifies that the input entity is "ACM Athena award", the type of related entities (i.e., target entities) is "person" and the relation is "winners of" (note that the narrative field is a *mixture* of input entity and relation), in a supporting document collection. The

* Corresponding author.
E-mail addresses: xtliu@udel.edu (X. Liu), zwei@udel.edu (W. Zheng), hfang@udel.edu (H. Fang).

0306-4573/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ipm.2013.03.004



```
<query>
  <entity_name>ACM Athena award</entity_name>
  <target_entity>person</target_entity>
  <narrative>
   winners of the ACM Athena award
  </narrative>
</query>
```

Fig. 1. An example structured query.

results are expected to include a list of the award winners such as "Deborah Estrin", "Shafi Goldwasser" and "Karen Spärck Jones".

The related entity finding problem can be naturally formulated as two subtasks: (1) candidate extraction, i.e., how to extract candidate entities based on the type specified in the query, and (2) candidate ranking, i.e., how to rank candidates according to their relevance to the query. While the first subtask can be solved using an off-the-shelf NER tagger or external resources such as DBPedia (McCreadie, Macdonald, Ounis, Peng, & Santos, 2010; Serdyukov & de Vries, 2010), the second subtask does not have any existing straightforward solution, and is thus more challenging.

In this paper, we focus on the second subtask, i.e., to rank candidate entities based on their relevance to a structured query. The main challenge is to leverage the structured query to estimate the probability that a candidate entity is relevant. Motivated by the general probabilistic frameworks proposed for traditional ad hoc retrieval (Lafferty & Zhai, 2003) and expert finding (Fang & Zhai, 2007), we derive six generative models for candidate entity ranking and then analytically and empirically compare these six functions. To further improve the performance, we study the problem of feedback in the context of related entity finding. In particular, we propose a novel mixture model based method that can utilize the pseudo feedback entities to estimate an enriched model for relation between the input and related entities. We conducted experiments on two standard TREC collections to evaluate the derived generative ranking models and the proposed relation-based feedback method.

We find that incorporating candidate priors is crucial to achieve reasonable retrieval performance. As a result, the retrieval models that can incorporate candidate priors are better choices than others since they are less sensitive to the quality of entity candidates. Furthermore, the proposed relation-based feedback methods are effective to further improve the retrieval performance.

The rest of the paper is organized as follows. We formulate the problem of related entity finding and present a general approach in Section 2. We discuss the proposed generative models in Section 3 and propose relation-based feedback method in Section 4. We discuss experiment results in Section 5, and discuss the related work in Section 6. Finally, we conclude in Section 7.

2. Problem formulation

The problem of *related entity finding* is to find the entities that are related to the information described in a query based on a supporting document collection. Unlike keyword queries used in the traditional ad hoc retrieval task, queries used in the related entity finding task are structured as shown in Fig. 1. In particular, a query specifies the type of related entities, an input entity and the relation between the input and related entities.

Formally, let $D=\{d_1, d_2, \ldots, d_n\}$ denote a supporting document collection, $E_t=\{e_{t_1}, e_{t_2}, \ldots, e_{t_m}\}$ denote a set of entities with entity type t, and $r(e_i,e_j) \in \{0,1\}$ denote whether there exists a relation r between two entities e_i and e_j. We define $q=\{t, e_{\text{in}}, r\}$ as a query with three components for the related entity finding task, where t is the type of related entities, e_{in} is the input entity, and r is the relation between the input and related entities. Note that if a query is formulated in the same format as the one discussed in Fig. 1, e_{in} corresponds to the value of the "entity_name" field, t corresponds to the value of the "target_entity" field, and r corresponds to the value of the "narrative" field excluding the value of the "entity_name" field. Given query q, the goal is to find a set of entities $E(q)$ that have type t and have relation r with e_{in}:

$$E(q) = \bigcup_{e_t \in E_t \wedge r(e_{\text{in}}, e_t)=1} \{e_t\}. \tag{1}$$

It often requires a two-step approach to solve the problem of related entity finding. The first step is *candidate extraction*, which is to find all the candidate entities with the specified type, i.e., $e_t \in E_t$. A simple solution is to directly apply an off-the-shelf NER tagger to extract candidate entities based on the specified type. The second step is *candidate ranking*, which is to further refine the candidate entities extracted in the first step and find those having the specified relation with the input entity, i.e,. $e_t \in E_t$, where $r(e_{\text{in}}, e_t) = 1$.

In this paper, we focus on the candidate ranking step. Following the spirit of the probabilistic ranking principle (Robertson, 1977), we rank the candidate entities according to the probabilities that they are relevant to a query. Thus, the key challenge is how to estimate the probability of a candidate entity being relevant, i.e., how likely it has the specified relations r

with the input entity e_{in}. Unlike keyword queries used in traditional retrieval tasks, we focus on deriving generative retrieval models and feedback method for the structured queries. The details are given in Section 3 and Section 4.

3. Generative models for candidate ranking

With a list of extracted candidate entities and the corresponding structured query, we now discuss how to assign a relevance score to each candidate based on the probability that the candidate has the specified relations with the input entity.

3.1. Basic idea

Given a query $q = \{t, e_{\text{in}}, r\}$ and a related entity candidate $e_t \in E_t$, we need to estimate the probability that e_t is relevant to q, i.e., $P(\mathcal{R}=1|q,e_t)$, where $\mathcal{R}$ is a binary random variable denoting the relevance. Following the previous studies (Fang & Zhai, 2007; Lafferty & Zhai, 2003) we may use the odds ratio to rank candidates and then apply Bayes' rule for the estimation:

$$P(\mathcal{R}=1|q,e_t) \overset{\text{rank}}{=} \frac{P(\mathcal{R}=1|q,e_t)}{P(\mathcal{R}=0|q,e_t)} = \frac{P(q,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(q,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} = \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)}, \tag{2}$$

where $\overset{\text{rank}}{=}$ means the two values are equivalent for ranking entity candidates e_t.

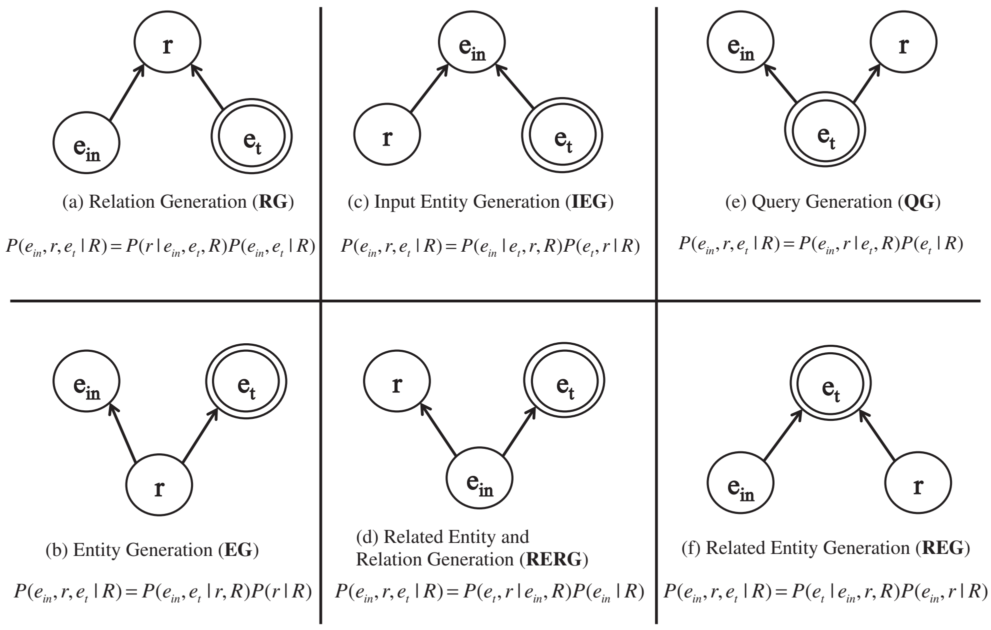
Now the challenge is how to factor the conditional probability $P(e_{\text{in}}, r, e_t|\mathcal{R})$. Since there are three variables involved, we may apply the chain rule in different ways based on their dependency relations shown in Fig. 2. Each dependency relation corresponds to a generative process among these variables. As an example, RG (relation generation) model assumes that relations are generated from the input and target entities.

By comparing these models, we can make the following observations:

- The top three models (i.e. *RG*, *IEG* and *QG*) share a similar property that e_t generates the other components, which enables us to incorporate a candidate-related prior for e_t, while the other models cannot utilize the candidate priors.
- Every pair of the models in the same column (i.e. *RG* and *EG*, *IEG* and *RERG*, *QG* and *REG*) would share some commonalities in the ranking functions because one component either generates the other two or is generated by the other two.

3.2. Derivation of generative models

We now discuss the derivation of the six generative models. Since the derivations are similar, we provide the details only for *RG* model and briefly explain the derivations for the other models.



r
e_in
e_t
(a) Relation Generation (RG)
P(e_in, r, e_t | R) = P(r | e_in, e_t, R)P(e_in, e_t | R)
(c) Input Entity Generation (IEG)
P(e_in, r, e_t | R) = P(e_in | e_t, r, R)P(e_t, r | R)
(e) Query Generation (QG)
P(e_in, r, e_t | R) = P(e_in, r | e_t, R)P(e_t | R)
(b) Entity Generation (EG)
P(e_in, r, e_t | R) = P(e_in, e_t | r, R)P(r | R)
(d) Related Entity and Relation Generation (RERG)
P(e_in, r, e_t | R) = P(e_t, r | e_in, R)P(e_in | R)
(f) Related Entity Generation (REG)
P(e_in, r, e_t | R) = P(e_t | e_in, r, R)P(e_in, r | R)

Fig. 2. Generative networks of different models.

3.2.1. Relation generation model (RG)

The RG model assumes that the relation between two entities is generated by a probabilistic model based on the entities (as shown in Fig. 2a). Thus, the conditional probability in Eq. (2) can be factored in the following ways:

$$\begin{aligned} P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &= \frac{P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot P(e_{\text{in}},e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(r|e_{\text{in}},e_t,\mathcal{R}=0)\cdot P(e_{\text{in}},e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &= \frac{P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(r|e_{\text{in}},e_t,\mathcal{R}=0)\cdot P(\mathcal{R}=0|e_{\text{in}},e_t)} \end{aligned}$$

It is reasonable to assume that conditioned on the event $\mathcal{R}=0$, entities and their relations (i.e., e_{in}, e_t and r) are independent. Under this assumption, we get

$$P(\mathcal{R}=1|q,e_t) \overset{\text{rank}}{=} \frac{P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(r|\mathcal{R}=0)\cdot P(\mathcal{R}=0|e_{\text{in}},e_t)} \overset{\text{rank}}{=} P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot\frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(\mathcal{R}=0|e_{\text{in}},e_t)} \tag{3}$$

Note that $P(r|\mathcal{R}=0)$ can be safely ignored for the purpose of ranking candidate entities since it is independent of e_t.

We now discuss how to estimate $P(r|e_{\text{in}},e_t,\mathcal{R}=1)$ in Eq. (3), which is the likelihood of e_{in} and e_t having the relation r given we know that two entities e_{in} and e_t are related, One possible approach is to directly estimate the probability based on an existing knowledge base about entities and their relations. However, one limitation is that the coverage of the knowledge base is not complete, which could lead to inaccurate estimation. In this paper, we instead propose to leverage supporting documents from the collection as a bridge to estimate the probability by exploiting the co-occurrences of these three variables (i.e., e_{in}, e_t and r) as follows:

$$\begin{aligned} P(r|e_{\text{in}},e_t,\mathcal{R}=1) &= \sum_{d\in D} P(r|d,e_{\text{in}},e_t,\mathcal{R}=1)P(d|e_{\text{in}},e_t,\mathcal{R}=1) \\ &= \sum_{d\in D} P(r|d,\mathcal{R}=1)P(d|e_{\text{in}},e_t,\mathcal{R}=1) \\ &\overset{\text{rank}}{=} \sum_{d\in D}\left(P(r|d,\mathcal{R}=1)\times\frac{P(e_{\text{in}},e_t|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_{\text{in}},e_t|d',\mathcal{R}=1)}\right) \end{aligned}$$

One assumption we have made in this estimation process is that r is independent of e_{in} and e_t given supporting document d. Fig. 3 shows a graphical representation of the relation generation model under the assumption.

There is another component in Eq. (3) that we need to estimate: $\frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(\mathcal{R}=0|e_{\text{in}},e_t)} = \frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{1-P(\mathcal{R}=1|e_{\text{in}},e_t)}$, in which $P(\mathcal{R}=1|e_{\text{in}},e_t)$ serves as the prior probability that e_t and e_{in} are related. We consider two methods: (1) *UniPrior*, which assumes that the prior probabilities are uniformly distributed; and (2) *OccPrior*, which estimates the prior probability of the two entities' being related based on their co-occurrence in the supporting documents, i.e., $\frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(\mathcal{R}=0|e_{\text{in}},e_t)} \propto \frac{c(e_{\text{in}},e_t,D)}{\sum_{e\in E_t}c(e_{\text{in}},e,D)}$, where $c(e_{\text{in}},e_t,D)$ is the number of documents that mention both entities in the supporting document collection D and E_t is a set of to-be-ranked entity candidates with the type t. The supporting document collection contains top ranked documents for the query.

3.2.2. Entity generation model (EG)

The EG model assumes that the input and candidate entities are generated by a probabilistic model based on their relation (as shown in Fig. 2b). Thus, the conditional probability in Eq. (2) can be factored as follows:

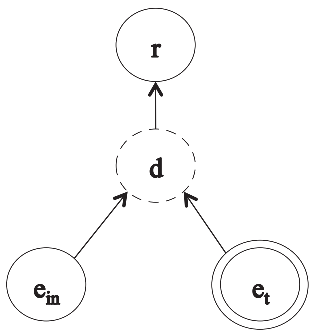


Fig. 3. Generative network of RG model based on supporting documents.

$$\begin{aligned}
P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)}\\
&= \frac{P(e_{\text{in}},e_t|r,\mathcal{R}=1)\cdot P(r|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},e_t|r,\mathcal{R}=0)\cdot P(r|\mathcal{R}=0)\cdot P(\mathcal{R}=0)}\\
&= \frac{P(e_{\text{in}},e_t|r,\mathcal{R}=1)\cdot P(\mathcal{R}=1|r)}{P(e_{\text{in}},e_t|r,\mathcal{R}=0)\cdot P(\mathcal{R}=0|r)}\\
&\overset{\text{rank}}{=} P(e_{\text{in}},e_t|r,\mathcal{R}=1)
\end{aligned} \tag{4}$$

Here we make the similar assumption as in the derivation of the RG model (i.e., e_t and e_{in} are independent of r conditioned on $\mathcal{R}=0$). And $P(\mathcal{R}|r)$ is safely ignored since it does not affect the ranking of the candidate entities. The component in Eq. (4) is estimated as:

$$\begin{aligned}
P(e_{\text{in}},e_t|r,\mathcal{R}=1) &= \sum_{d\in D} P(e_{\text{in}},e_t|d,r,\mathcal{R}=1)P(d|r,\mathcal{R}=1)\\
&= \sum_{d\in D} P(e_{\text{in}},e_t|d,\mathcal{R}=1)P(d|r,\mathcal{R}=1)\\
&\overset{\text{rank}}{=} \sum_{d\in D}\left(P(e_{\text{in}},e_t|d,\mathcal{R}=1)\times\frac{P(r|d,\mathcal{R}=1)}{\sum_{d'\in D}P(r|d',\mathcal{R}=1)}\right)\\
&\overset{\text{rank}}{=} \sum_{d\in D} P(r|d,\mathcal{R}=1)P(e_{\text{in}},e_t|d,\mathcal{R}=1)
\end{aligned}$$

Here we drop $\sum_{d'\in D}P(r|d',\mathcal{R}=1)$ as it does not affect the ranking of e_t.

3.2.3. Input entity generation model (IEG)

The IEG model assumes that an input entity is generated by a probabilistic model based on the candidate and their relation (as shown in Fig. 2c). Following the similar assumptions used for the derivation of other models, we can rank the candidates using:

$$P(\mathcal{R}=1|q,e_t) \overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \overset{\text{rank}}{=} P(e_{\text{in}}|e_t,r,\mathcal{R}=1)\cdot\frac{P(\mathcal{R}=1|e_t,r)}{P(\mathcal{R}=0|e_t,r)} \tag{5}$$

The two component functions can be estimated as:

$$P(e_{\text{in}}|e_t,r,\mathcal{R}=1) \overset{\text{rank}}{=} \sum_{d\in D}\left(P(e_{\text{in}}|d,\mathcal{R}=1)\times\frac{P(e_t,r|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t,r|d',\mathcal{R}=1)}\right)$$

$$\frac{P(\mathcal{R}=1|e_t,r)}{P(\mathcal{R}=0|e_t,r)} \propto \frac{c(e_t,r,D)}{\sum_{e\in E_t}c(e,r,D)}$$

3.2.4. Related entity and relation generation model (RERG)

The RERG model assumes that both relation and candidate entities are generated by a probabilistic model based on an input entity (as shown in Fig. 2d). Following the similar assumptions used for the derivation of other models, we can rank the candidates using:

$$P(\mathcal{R}=1|q,e_t) \overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \overset{\text{rank}}{=} P(e_t,r|e_{\text{in}},\mathcal{R}=1), \tag{6}$$

where $P(e_t,r|e_{\text{in}},\mathcal{R}=1)$ can be estimated as:

$$P(e_t,r|e_{\text{in}},\mathcal{R}=1) \overset{\text{rank}}{=} \sum_{d\in D}P(e_{\text{in}}|d,\mathcal{R}=1)P(e_t,r|d,\mathcal{R}=1)$$

3.2.5. Query generation model (QG)

The QG model assumes that a query with an input entity and a relation is generated by a probabilistic model based on a candidate entity (as shown in Fig. 2e). Applying similar assumptions, the model is shown as follows:

$$P(\mathcal{R}=1|q,e_t) \overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \overset{\text{rank}}{=} P(e_{\text{in}},r|e_t,\mathcal{R}=1)\cdot\frac{P(\mathcal{R}=1|e_t)}{P(\mathcal{R}=0|e_t)}, \tag{7}$$

where $P(e_{\text{in}},r|e_t,\mathcal{R}=1)$ can be estimated as:

$$P(e_{\text{in}},r|e_t,\mathcal{R}=1) \overset{\text{rank}}{=} \sum_{d\in D}\left(P(e_{\text{in}},r|d,\mathcal{R}=1)\times\frac{P(e_t|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t|d',\mathcal{R}=1)}\right)$$

Table 1
Derivations of six generative models.

RG	$P(\mathcal{R}=1\|q,e_t)\overset{\text{rank}}{=}\sum_{d\in D}\left(P(r\|d,\mathcal{R}=1)\times\frac{P(e_{\text{in}},e_t\|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_{\text{in}},e_t\|d',\mathcal{R}=1)}\right)\times\frac{P(\mathcal{R}=1\|e_{\text{in}},e_t)}{P(\mathcal{R}=0\|e_{\text{in}},e_t)}$
EG	$P(\mathcal{R}=1\|q,e_t)\overset{\text{rank}}{=}\sum_{d\in D}(P(e_{\text{in}},e_t\|d,\mathcal{R}=1)\times P(r\|d,\mathcal{R}=1))$
IEG	$P(\mathcal{R}=1\|q,e_t)\overset{\text{rank}}{=}\sum_{d\in D}\left(P(e_{\text{in}}\|d,\mathcal{R}=1)\times\frac{P(e_t,r\|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t,r\|d',\mathcal{R}=1)}\right)\times\frac{P(\mathcal{R}=1\|e_t,r)}{P(\mathcal{R}=0\|e_t,r)}$
RERG	$P(\mathcal{R}=1\|q,e_t)\overset{\text{rank}}{=}\sum_{d\in D}(P(e_t,r\|d,\mathcal{R}=1)\times P(e_{\text{in}}\|d,\mathcal{R}=1))$
QG	$P(\mathcal{R}=1\|q,e_t)\overset{\text{rank}}{=}\sum_{d\in D}\left(P(e_{\text{in}},r\|d,\mathcal{R}=1)\times\frac{P(e_t\|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t\|d',\mathcal{R}=1)}\right)\times\frac{P(\mathcal{R}=1\|e_t)}{P(\mathcal{R}=0\|e_t)}$
REG	$P(\mathcal{R}=1\|q,e_t)\overset{\text{rank}}{=}\sum_{d\in D}(P(e_t\|d,\mathcal{R}=1)\times P(e_{\text{in}},r\|d,\mathcal{R}=1))$

and

$$\frac{P(\mathcal{R}=1|e_t)}{P(\mathcal{R}=0|e_t)}\propto\frac{c(e_t,D)}{\sum_{e\in E_t}c(e,D)}.$$

3.2.6. Related entity generation model (REG)

The REG model assumes that a related entity candidate is generated by a probabilistic model based on the input entity and their relation (as shown in Fig. 2f). Applying similar assumptions made for other models, the model is shown as follows:

$$P(\mathcal{R}=1|q,e_t)\overset{\text{rank}}{=}\frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)}\overset{\text{rank}}{=}P(e_t|e_{\text{in}},r,\mathcal{R}=1), \tag{8}$$

where $P(e_t|e_{\text{in}},r,\mathcal{R}=1)$ can be estimated as:

$$P(e_t|e_{\text{in}},r,\mathcal{R}=1)\overset{\text{rank}}{=}\sum_{d\in D}P(e_{\text{in}},r|d,\mathcal{R}=1)P(e_t|d,\mathcal{R}=1)$$

3.3. Discussions

Table 1 summarizes all the six derived models when leveraging supporting documents for estimation. By comparing with the derivations of the six models above, we find that models in the same cell are similar to each other. Let us take *QG* and *REG* for example. The main difference to distinguish them is that *QG* contains a candidate normalizer, i.e., $\sum_{d'\in D}P(e_t|d',\mathcal{R}=1)$. Since the normalizer penalizes the popular candidates, we hypothesize that *QG* with uniform candidate prior may over-favor noisy candidates that occur less frequently in the supporting documents, which would lead to worse retrieval performance than *REG* if the noisy entity candidate ratio is high. Similar hypotheses can be made for the models in the other two cells. We conduct empirical evaluation in Section 5, and the empirical results are consistent with the hypotheses above.

We now describe how to estimate the conditional probability $P(X|d,\mathcal{R}=1)$, which essentially describes how likely X is "generated" from the observed document d. Here X may be individual query components (i.e. e_t, e_{in} and r), or the combinations between any two query components (i.e. $\{e_{\text{in}},r\}$, $\{e_t,r\}$ and $\{e_{\text{in}},e_t\}$) where the combination is essentially the union of term sets from two query components. The conditional probability can be estimated with language modeling approach by representing both X and d as bags of terms. In this paper, we use Dirichlet smoothing method for the estimation since it has been shown to be more effective for the document retrieval problem (Zhai & Lafferty, 2001). Since our framework is rather general, other ways of estimating the models could also be applied. We will discuss how to utilize feedback information to better estimate $P(r|d,\mathcal{R}=1), P(e_{\text{in}},r|d,\mathcal{R}=1)$ and $P(e_t,r|d,\mathcal{R}=1)$ in Section 4.

Recall that we have discussed two methods to estimate the prior probabilities such as $\frac{P(\mathcal{R}=1|e_t)}{P(\mathcal{R}=0|e_t)}$, i.e., *UniPrior* and *OccPrior*. Which one should we choose? It depends on whether the assumption (i.e., popular candidate entities tend to be relevant ones) holds. There are two types of noisy candidates in the candidate set: non-relevant entities and non-real-world entities. When a candidate set contains lots of noisy entities with the second type, i.e., those are not real entities, this assumption may not hold because those noisy candidates with common words might also occur very frequently. However, when a candidate set contains only the first type noisy entities, i.e., most of the candidates are real-world entities, the assumption would hold and the *OccPrior* is expected to improve the performance.

3.4. Connections with other entity track models

We now discuss the connections between the derived models and methods used in the top performed runs in the TREC Entity Track.

- *REG* is similar to the models used by Purdue (Fang, Yu, Xian, & Xu, 2009), PRIS (Wang et al., 2010) and NiCT (Wu & Kashioka, 2009; Wu, Hori & Kawai, 2010). The main differences come from the estimation of component functions. For example, Fang et al. proposed to estimate the relevance scores based on not only documents but also passages (Fang et al., 2009), Wang et al. tried to incorporate a document prior (Wang et al., 2010), and Wu. et al. utilized Wikipedia and term proximity in relevance modeling (Wu & Kashioka, 2009; Wu, Hori & Kawai, 2010).
- *RG* is similar to the models used by ICT (Zhai, Cheng, Guo, Xu & Liu, 2010; Cao et al., 2010) and University of Amsterdam Bron et al. (2010). The main difference is the estimation of candidate priors. For example, instead of relying on the collection, external resources such as DBPedia and Freebase were used to compute the prior, i.e., how likely a candidate has the specified type.
- *REG* and *QG* with uniform prior were studied in our previous work in the context of entity track (Zheng, Gottipati, Jiang, & Fang, 2010). However, the performance was not very satisfying because the quality of the candidate entity sets was low.

It is clear that our proposed framework is general enough to include three existing ranking methods as well as three new ones. Although some of these models have been studied, it remains unclear which ranking models are more effective based on TREC official results. For example, REG has been used by multiple groups in TREC 2009 (Fang et al., 2009; Wu & Kashioka, 2009; Zheng et al., 2010), but the performance were quite different (e.g., the results measured with $nDCG@N_R$ ranged from 0.0488 to 0.3061). The reason for the performance differences could be twofold. First, the pre-processing quality varies as they use different ways of entity candidate extraction ranging from simply applying off-the-shelf NER taggers (Zheng et al., 2010) to more sophisticated methods such as extracting entities from tables and lists (Fang et al., 2009) or using Wikipedia link information for entity filtering (Wu & Kashioka, 2009). Furthermore, they estimated the component function using different resources, such as using only documents (Zheng et al., 2010), using only supporting snippets (Wu & Kashioka, 2009), and using multiple resources (Fang et al., 2009). In fact, one of our goals is to set up a common ground for comparing and understanding the effectiveness of different ranking models.

4. Relation feedback method

Feedback is a commonly used technique to improve retrieval performance through finding more terms that are related to the given topic. In the scenario of document retrieval, a new query model can be estimated using the feedback documents (Zhai & Lafferty, 2001). In this paper, we study the problem of feedback in the context of related entity finding task, where a query specifies two fields, i.e., relation and input entity. The input entity is often clearly specified with the entity name. However, there may be several different expressions for the relation between the input entity and target entity, the original relation r may not cover all of them. Consider the example query in Fig. 1. The relevant entity "Deborah Estrin" may be described as "Deborah Estrin *received* the ACM Athena Award" or "the ACM Athena Award was *granted to* Deborah Estrin" instead of saying "Deborah Estrin is the *winner of* ACM Athena Award" in many documents. If we use only the original relation description, we may fail to retrieve such relevant entities.

We thus propose a relation feedback method to estimate an enriched relation model $\theta_{\mathcal{R}_f}$. Following the problem setup of document feedback, we may use a basic related entity finding model, such as the ones proposed in the previous section, to retrieve a list of candidate related entities. We then assume that the top ones are relevant. The challenge is how to utilize these feedback entities to estimate an enriched model for the relation specified in the query.

To address the challenge, we propose a novel mixture model based relation feedback method. For each feedback entity, we first generate a new query that includes the input entity, feedback entity and their relation. We then retrieve a set of feedback documents $\mathcal{F}_d$ for every generated new query. Intuitively, these feedback documents contain information about the input entity, related entity and their relations. Thus, we can assume that these feedback documents are generated from a mixture model of input entities, candidate entities, their relations and a background model:

$$\log p(\mathcal{F}_d|\theta_{\mathcal{F}}) = \sum_{d_i \in \mathcal{F}_d} \sum_{w} c(w; d_i) \log(\beta_1 p(w|\theta_{in}) + \beta_2 p(w|\theta_t) + (1 - \beta_1 - \beta_2 - \lambda) p(w|\theta_{\mathcal{R}_f}) + \lambda p(w|\mathcal{C})) \qquad (9)$$

where $c(w; d)$ is the occurrences of term w in document d, θ_{in} is the language model of input entity e_{in}, θ_t is the language model of candidate entity $e_t, \theta_{\mathcal{R}_f}$ is the enriched language model of relation $r, \mathcal{C}$ is the collection language model that explains the background information or non-relevant topics and $\theta_{\mathcal{F}}$ is the language model of the supporting documents $\mathcal{F}_d$. θ_{in} and θ_t can be estimated by taking e_{in} and e_t as queries and using model-based feedback (Zhai & Lafferty, 2001) to estimate the expanded query model, while $\mathcal{C}$ and $\theta_{\mathcal{F}}$ can be estimated based on the term frequency distribution over the whole document collection and supporting documents, respectively. Note that β_1, β_2 and λ are coefficients of linear interpolation and they represent the likelihood that a term w is generated by each of the language models. We will set these three parameters to some constants and estimate only the enriched relation model $\theta_{\mathcal{R}_f}$ using EM algorithm (Dempster, Laird, & Rubin, 1977).

The EM updates for $p(w|\hat{\theta}_{\mathcal{R}_f})$ are:

$$t^{(n)}(w) = \frac{\gamma p^{(n)}(w|\theta_{\mathcal{R}_f})}{\beta_1 p(w|\theta_{in}) + \beta_2 p(w|\theta_t) + \gamma p^{(n)}(w|\theta_{\mathcal{R}_f}) + \lambda p(w|\mathcal{C})}, \tag{10}$$

$$p^{(n+1)}(w|\theta_{\mathcal{R}_f}) = \frac{\sum_{j=1}^{n} c(w;d_j)t^{(n)}(w)}{\sum_i \sum_{j=1}^{n} c(w_i;d_j)t^{(n)}(w_i)}, \tag{11}$$

where $\gamma = (1 - \beta_1 - \beta_2 - \lambda)$. The relation language model estimation process can be interpreted as extracting the relation language model by eliminating the collection background noise as well as language models of e_{in} and e_t.

Since we may use several top entities as feedback and each of them will derive one estimation of enriched relation language model, we take the average of them to get the final relation language model $\hat{\theta}_{\mathcal{R}_f}$ with regard to the initial query. After we get $\hat{\theta}_{\mathcal{R}_f}$, we may interpolate it with the original relation model $\hat{\theta}_r$ to obtain the updated relation model $\hat{\theta}_{r_{new}}$ as follows:

$$\hat{\theta}_{r_{new}} = (1-\alpha)\hat{\theta}_r + \alpha\hat{\theta}_{\mathcal{R}_f}, \tag{12}$$

where α is the coefficient of linear interpolation and it works to control the influence of the feedback model.

Finally, we can incorporate the enriched relation model $\hat{\theta}_{r_{new}}$ into our derived generative models through generalizing the likelihood $p(r|d,\mathcal{R}=1)$ as the cross entropy of the enriched relation model and the document model, i.e.,

$$p(r|d,\mathcal{R}=1) = z \cdot \exp\left(\sum_w p(w|\hat{\theta}_{r_{new}}) \log(p(w|\theta_d))\right) \tag{13}$$

where the document model θ_d is estimated based on d using Dirichlet prior smoothing, and z is a normalization factor estimated as follows:

$$z = 1 \Big/ \sum_{r'} p(r'|d, R=1). \tag{14}$$

The likelihood $p(e_{\text{in}}, r|d,\mathcal{R}=1)$ can be estimated in a similar way.

5. Experiments

5.1. Experiment setup

We evaluate the proposed methods on two standard collections developed in the TREC Entity track.

Table 2
Statistics of two standard Entity Track collections.

Related entity type	Ent09		Ent10	
	# Of query	Avg. # Of RelEnt	# Of query	Avg. # Of RelEnt
Product	3	10.3	1	7
Person	6	17.8	9	10.2
Organization	11	19	33	17.8
Location	–	–	7	18.1
All	20	17.4	50	16.3

Table 3
Comparison of different generative models over **NER**. Note that *NoPrior* is the notation for *EG*, *RERG* and *REG* as their ranking functions cannot incorporate entity candidate prior.

Model	**Ent09**		**Ent10**	
	NoPrior/UniPrior	OccPrior	NoPrior/UniPrior	OccPrior
RG	0.073	0.096	0.047	0.054
EG	0.091	–	0.058	–
IEG	0.048	0.090	0.045	0.055
RERG	0.101	–	0.050	–
QG	0.108	0.107	0.039	0.059
REG	0.120	–	0.047	–

Table 4
Statistics of two entity candidate sets.

Data set	Entity candidate set	# of ENT	# of REL-ENT	Precision (%)	Recall (%)
Ent09	**NER**	8,321	169	2.0	67.9
	NER + DBPedia	1,595	110	6.9	44.2
Ent10	**NER**	23,418	519	2.2	63.8
	NER + DBPedia	3,573	384	10.7	47.2

Table 5
Comparison of different generative models over **NER + DBPedia**.

Model	**Ent09**		**Ent10**	
	NoPrior/UniPrior	OccPrior	NoPrior/UniPrior	OccPrior
RG	0.150	0.210	0.218	0.238
EG	0.167	–	0.204	–
IEG	0.215	0.220	0.229	0.231
RERG	0.201	–	0.222	–
QG	0.216	0.221	0.222	0.245
REG	0.213	–	0.232	–

Table 6
Top 5 entities of query #7 "Airlines that currently use Boeing 747 planes" using *RG* from two candidate sets in **Ent09**. Relevant entities are denoted in **bold** text.

NER	NER + DBPedia
Boeing	**Northwest Airlines**
China	American Airlines
Northwest Airlines	**Lufthansa**
Northwest	**China Airlines**
American Airlines	**Japan Airlines**

- *Ent09:* The collection is developed and used in TREC 2009 Entity Track (Balog et al., 2010). It includes (1) ClueWeb09 category B collection with 50 million English web pages crawled from the Internet; (2) 20 queries, each of which corresponds to an information need related to entities, and the types of related entities including person, organization and product; and (3) the judgement file with relevant entities and homepage for every query, where each query has around 17 relevant entities on average.
- *Ent10:* The collection is developed and used in TREC 2010 Entity Track. It includes (1) English portion of ClueWeb09 collection with about 500 million web pages crawled from the Internet; (2) 50 queries, the format is the same as that of 2009 Entity Track, while a new related entity type *Location* is introduced; and (3) the judgement file with relevant entities and homepage for every query, where each query has around 16 relevant entities on average.

Table 2 shows the statistics of the two collections including the number of queries and average number of relevant entities per query for different related entity types.

The performance is primarily measured with the official measure used in the track: $nDCG@N_R$, i.e., normalized Discounted Cumulative Gain at N_R, where N_R is the number of relevant entities for a given query. Note that the measure is based on the homepage of the entity rather than the entity itself, one homepage has three relevance levels: primary, relevant and non-relevant, and the corresponding gains are 2, 1, 0, respectively.

The smoothing parameter μ is set to 1000 for all the experiments.

5.2. Comparison of derived ranking models

We conduct two sets of experiments to evaluate the proposed six retrieval models. Recall that the related entity finding requires two steps: entity candidate extraction and entity ranking. Since we focus on entity ranking in this paper, we use two simple strategies for entity candidate selection.

In the first set of experiments, we apply an existing NER tagger to extract entity candidates based on the specified type. Specifically, a query is first formatted into Indri query language format, and then used to retrieve top ranked paragraphs from the document collection. We then apply the Stanford NER tagger (Krishnan & Manning, 2006) to extract all the candidates from the retrieved paragraph collection (denoted as **NER**). The reason to choose paragraphs rather than whole documents to

Table 7
Performance (*nDCG@N_R*) comparison for relation-based feedback methods on *RG* and *QG* models. ★ means improvement over **NO-FB** is statistically significant at 0.05 level based on Wilcoxon signed-rank test.

Entity candidate set	Models	**Ent09**		**Ent10**	
		NO-FB	**OPTIMIZED**	**NO-FB**	**OPTIMIZED**
NER	RG + OccPrior	0.096	0.111★(+15.6%)	0.054	0.065(+20.4%)
	QG + OccPrior	0.107	0.111(+3.7%)	0.059	0.063★(+6.8%)
NER + DBPedia	RG + OccPrior	0.210	0.241★(+14.8%)	0.238	0.253★(+6.3%)
	QG + OccPrior	0.221	0.225★(+1.8%)	0.245	0.247(+0.8%)

Table 8
Recommended parameter settings for relation-based feedback.

Parameter	n_e	α	λ	β_1	β_2
Value	10	0.60	0.60	0.10	0.10

Table 9
Performance comparison (*nDCG@N_R*) for relation-based feedback methods on *RG + OccPrior* model over **NER + DBPedia**.

Data set	**OPTIMIZED**	**TRAINED**	**FIXED**
Ent09	0.241	0.239	0.237
Ent10	0.253	0.244	0.244

apply NER is that we found the quality of NER extracted entity candidates from paragraphs are better than that from documents because paragraphs have better proximity for entities. After that, we apply the proposed ranking models and feedback method to rank the entity candidates.

Table 3 shows the performance comparison on the NER generated entity candidate set. The performance of all the models is similar, and incorporating *OccPrior* into *RG*, *IEG* and *QG* models can improve the performance in most cases. Moreover, with entity candidate prior, models on the top row of Fig. 2 have comparable performance with the other models in the bottom row of same column (e.g. comparing *RG + OccPrior* with EG).

By analyzing the results, we find that using only existing NER taggers cannot generate a high quality set of candidates because many results from NER extraction are non-real-world entities with the correct type. For example, consider the query "Winners of ACM Athena Award" with the entity type "person", there are many noisy entity candidates from the NER extraction: "Murray Hopper Award", "Dr. J", "Palo Alto", "Kate". Among the 8321 entity candidates extracted in **Ent09**, 169 (2.0%) are relevant, and among 23,418 entity candidates in **Ent10**, 519 (2.2%) are relevant.

To construct a better candidate set, we follow a heuristic used in TREC (Bron et al., 2010) and apply DBPedia-based filtering over all the entity candidates from NER extraction (denoted as **NER + DBPedia**). We keep the entities which have valid DBPedia URI and belong to the type specified in the query according to the ontology.[1] The statistics of both entity candidate sets are shown in Table 4. We find that NER based entity extraction can lead to high recall, and DBPedia-based filtering can reach higher precision with acceptable loss of recall, showing it is an effective heuristic.

We then conduct another set of experiments to evaluate the performance of our entity ranking models on **NER + DBPedia**, and summarize the results in Table 5. Compared with Table 3, we find that the models perform much better on the new candidate set. Moreover, incorporating priors is more effective in improving the retrieval performance because all "non-entity" noises are filtered out.

To better understand the impacts of two entity candidate sets on the performance of entity ranking, we choose one query from **Ent09** (query #7 "Airlines that currently use Boeing 747 planes"), and show the top 5 entities for *RG* model with *OccPrior* on the two entity candidate sets in Table 6. We find that DBPedia based filtering can purify the entity candidate set (i.e. removing non-type-match entities like "Boeing", "China", etc.).

5.3. Effectiveness of relation-based feedback methods

We conduct experiments to examine the effectiveness of the proposed relation feedback method. Specifically, we choose *RG* and *QG* model with *OccPrior* for evaluation because QG is the one that performs best, and RG is the one that can more

[1] The DBpedia Ontology: http://wiki.dbpedia.org/Ontology.

naturally incorporate feedback. We choose the original relation model as the baseline, denoted as **NO-FB**. We first do parameter tuning of our related-based feedback methods and report the optimized results as **OPTIMIZED**.

Table 7 shows the performance of our feedback methods on the two entity candidate sets respectively. It is clear that our proposed relation feedback method can significantly improve the performance on both data sets under optimized parameter settings. Moreover, the improvement of *RG* model is larger than that of *QG*. By analyzing the results, we find that the proposed relation feedback method is able to find terms that are relevant to the relation described in the query. For example, for query ''Winners of ACM Athena Award'', our proposed model can find useful terms, such as ''nominate'' and ''receive'', for the relation ''winners of''.

There are five parameters in the relation-based feedback method: the number of feedback entities n_e, α (Eq. (12)), λ (Eq. (10)), β_1 (Eq. (10)), and β_2 (Eq. (10)). Since the optimized parameter settings on different data sets are different, we would like to see whether the optimized parameter setting on one data set can lead to optimized performance on others. We use one data set to train the optimized parameter setting and apply it to the other. The results are denoted as **TRAINED**. By conducting experiments to examine the parameter sensitivity of the relation feedback method, we find that the performance is relatively robust (±3%) with regard to the variation of most parameters. Therefore, we recommend a set of fixed parameter settings based on the observations of parameter tuning process as a guidance for further research efforts, as shown in Table 8. The results associated with the recommended parameter setting are denoted as **FIXED**. Table 9 shows the results comparison for **OPTIMIZED**, **TRAINED** and **FIXED**. We find that both the trained parameter setting and the recommended fixed parameter setting can lead to near-optimized performance for our related-based feedback method.

5.4. Comparison with TREC results

We compare our models with the methods in TREC 2009 Entity Track (Balog et al., 2010). With the fixed parameter setting, our system is able to be ranked at the third place in Entity Track 2009 and fourth in Entity Track 2010, respectively. Note that the top 2 runs used additional information such as passage-based relevance, Wikipedia link information and term proximity (Fang et al., 2009; Wu & Kashioka, 2009), which are not used in our work. We expect that the performance of our proposed method could be further improved if combined with these additional techniques.

6. Related work

The related entity finding task is similar to the traditional ad hoc document retrieval task (Salton & McGill, 1983) in the sense that both tasks rank a list of information items, i.e., entities or documents, based on their relevance to a given query. Related entity finding has attracted much attention due to the launch of the TREC Entity track (Balog et al., 2010).

Most participants first extracted entity candidates using either off-the-shelf or custom-made named entity recognizers, and then ranked the entities using different ranking strategies. Fang et al. (2009) applied the hierarchy relevance model to estimate the relevance score at document, passage and entity levels. Kaptein, Koolen, and Kamps (2010) used the outgoing links from the page of input entity to the page of target entities to model the relevance score between them. Serdyukov and de Vries (2010) utilized the external links on the Wikipedia pages of entities to rank the entities. Kaptein, Serdyukov, de Vries, and Kamps (2010), Kamps, Kaptein, and Koolen (2010) leveraged category information and external entities on the Wikipedia pages as a pivot to rank the entities. The voting model (Santos, Macdonald, & Ounis, 2010; Macdonald & Ounis, 2009) took the relevance score of each overlapped document between the query retrieved set and entity profile set as a vote for the entity. Hold, Leben, Barczynski, and Brauer (2010) combined the Jaccard and Jaro-Winkler similarity to merge duplicate entities into groups and took the distance in sentences between input and target entities to model the relevance. Wu, Hori and Kawai (2010) integrated the relevance score between the query and both the entity and homepage of the entity. Besides these, some ranking strategies were combined with the use of various heuristics such as result filtering (Vydiswaran, Ganesan, Lv, He, & Zhai, 2010), the use of anchor text (Zhai, Cheng, Guo, Xu & Liu, 2010) and the use of external resources such as WordNet (Fang et al., 2009), Google results (Wu & Kashioka, 2009) or DBPedia (McCreadie et al., 2010; Serdyukov & de Vries, 2010). Besides these heuristics and difference between the estimation of component function, the basic idea of entity ranking models in Fang et al. (2009), Wu, Hori and Kawai (2010), Wu and Kashioka (2009) are essentially very similar to our *REG* model.

The related entity finding problem is also related to the expert finding problem (Balog, Meij, & de Rijke, 2007). In fact, it can be regarded as a more general problem than expert finding (Balog et al., 2007), since the latter focuses only on a specific type of related entities (i.e., person), a specific type of input entities (i.e., expertise area) and a specific relation (i.e., expert of).

Another body of related work is the entity ranking track of INEX (de Vries, Vercoustre, Thom, Craswell, & Lalmas, 2008; Demartini, de Vries, Iofciu, & Zhu, 2009; Demartini, Iofciu, & de Vries, 2010), which emphasizes more on the type of the targeted entities (i.e, categories) than the relation between the target and input entities. Vercoustre, Pehcevski, and Naumovski (2009) utilized a four-class prediction step to predicate the topic difficulty which can be used to further improve the entity ranking performance by setting the retrieval parameters to the optimal values based on the prediction. Rode, Hiemstra, Vries, and Serdyukov (2009) used the PF (PathFinder)/Tijah system which combines database and information retrieval technology through integrating the retrieval language NEXI with the database query language XQuery and applies relevance score propagation to rank the entities. Kaptein and Kamps (2009) explored relational information of Wikipedia pages, categories to estimate the distance between document categories and target categories and utilized the link information to perform relevance propagation. Both the relation and link information can lead to performance improvement. Itakura and Clarke (2009) used redundancy technique for question answering (Clarke, Cormack, & Lynam, 2001) as well as category estimation to retrieve top rank passages from which entities are extracted. Entities are ranked based on the number of passages which contain them then. Balog, Bron, De Rijke, and Weerkamp (2009), Balog, Bron, and de Rijke (2010), Balog, Bron, and De Rijke (2011) performed entity ranking using a KL-Divergence based generation model which integrates both term-based and category-based representations of queries and entities. The underlining generation model is similar to our *QG* model with uniform entity prior, but the estimation of generation probability is different from ours: they used Wikipedia documents associated with query entity and target entity to connect them while we use the whole collection to bridge them. Koolen, Kaptein, and Kamps (2009) proposed a probability framework which also exploits the category information explicitly. Both queries and entities are represented by term-based and category-based model which essentially are probability distributions. The ranking of entities can be done by measuring similarities between the probability distributions.

There are also some studies on how to apply relevance feedback on entity ranking. Balog et al. (2010), Balog et al. (2011) focused on expanding query term model and category model directly from the documents associated with feedback entities. Iofciu, Demartini, Craswell, and de Vries (2011) utilized the Wikipedia category graph structure to estimate the entity ranking weight which is interpolated with the initial ranking as the final one. However, their models heavily rely on the structure of Wikipedia (e.g. links, categories) and cannot be extended to web scale corpus with unstructured documents (e.g. ClueWeb09 used in TREC Entity Track). Different from their models, our relevance feedback model estimates the enriched entity relation model from the supporting documents of feedback entities and it does not use any specific structure information of the document.

Compared with the previous work, our main contributions include the following: (1) We derive six generative models for related entity finding, connect them with methods used in Entity track, and conduct both analytical and empirical studies to compare these models. (2) We propose a novel relation-based feedback method that can better capture the information about the relation between the input and related entities. Note that both retrieval models and feedback method are complementary to the heuristics used in the Entity track and INEX and can be combined to further improve the performance.

7. Conclusions and future work

We study the related entity finding problem in this paper. Specifically, we focus on ranking candidate entities based on their relevance to a structured query. To tackle this problem, we first derive six generative models to rank candidates. As far as we know, this is the first systematic study of the formal models for related entity finding. Experimental results indicate that the retrieval performance is closely related to the quality of extracted candidate sets. It is more effective to incorporate candidate prior when the quality of candidates is higher. Moreover, we propose a novel relation based feedback method that can utilize the pseudo relevant entities to better capture the relation between entities. Experimental results show that the proposed feedback method is effective to improve the performance.

There are many interesting future research directions. First, we plan to explore alternative ways of estimating component functions in the models. Second, we could study how to leverage the NER results in the ranking procedure. Finally, it would be interesting to study how to combine our proposed ranking strategies with existing QA methods to improve the performance.

References

Balog, K., Bron, M., & De Rijke, M. (2011). Query modeling for entity search based on terms, categories, and examples. *ACM Transactions on Information Systems, 29*, 22:1–22:31.

Balog, K., Meij, E., & de Rijke, M. (2007). The University of Amsterdam at the TREC 2006 enterprise track. In *Proceedings of TREC*.

Balog, K., Bron, M., De Rijke, M., & Weerkamp, W. (2009). Combining term-based and category-based representations for entity search. In *Proceedings of INEX*.

Balog, K., de Vries, A. P., Serdyukov, P., Thomas, P., & Westerveld, T. (2010). Overview of the TREC 2009 entity track. In *Proceedings of TREC*.

Balog, K., Bron, M., & de Rijke, M. (2010). Category-based query modeling for entity search. In *Proceedings of ECIR*.

Bron, M., He, J., Hofmann, K., Meij, E., de Rijke, M., Tsagkias, M., et al. (2010). The University of Amsterdam at TREC 2010 session, entity, and relevance feedback. In *Proceedings of TREC*.

Cao, L., Bai, L., Cheng, X., Guo, J., Xu, H., Liu, Y., et al. (2010). ICTNET at entity track TREC 2010. In *Proceedings of TREC*.

Clarke, C. L. A., Cormack, G. V., & Lynam, T. R. (2001). Exploiting redundancy in question answering. In *Proceedings of SIGIR*.

Demartini, G., de Vries, A. P., Iofciu, T., & Zhu, J. (2009). Overview of the INEX 2008 entity ranking track. In *Advances in Focused Retrieval*.

Demartini, G., Iofciu, T., & de Vries, A. (2010). Overview of the INEX 2009 entity ranking track. In *Focused Retrieval and Evaluation*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological), 39*, 1–38.

de Vries, A. P., Vercoustre, A.-M., Thom, J. A., Craswell, N., & Lalmas, M. (2008). Overview of the INEX 2007 entity ranking track. In *Focused Access to XML Documents*.

Fang, H., & Zhai, C. (2007). Probabilistic models for expert finding. In *Proceedings of ECIR*.

Fang, Y., Yu, Z., Xian, Y., & Xu, Y. (2009). Entity retrieval by hierarchical relevance model, exploiting the structure of tables and learning homepage classifiers. In *Proceedings of TREC*.

Hold, A., Leben, M., Barczynski, B. E. W., & Brauer, F. (2010). ECIR G a lightweight approach for entity-centric information retrieval. In *Proceedings of TREC*.

Iofciu, T., Demartini, G., Craswell, N., de Vries, A. (2011). ReFER: Effective relevance feedback for entity ranking. In *Advances in Information Retrieval* (pp. 264–276).

Itakura, K. Y., & Clarke, C. L. A. (2009). University of waterloo at INEX 2009: Ad hoc, book, entity ranking, and link-the-wiki tracks. In *Proceedings of INEX*.

Kamps, J., Kaptein, R., & Koolen, M. (2010). Using anchor text, spam filtering and Wikipedia for web search and entity ranking. In *Proceedings of TREC*.
Kaptein, R., & Kamps, J. (2009). Finding entities in Wikipedia using links and categories. In *Advances in Focused Retrieval*.
Kaptein, R., Koolen, M., & Kamps, J. (2010). Experiments with result diversity and entity ranking: Text, anchors, links and Wikipedia. In *Proceedings of TREC*.
Kaptein, R., Serdyukov, P., de Vries, A., & Kamps, J. (2010). Entity ranking using Wikipedia as a pivot. In *Proceedings of CIKM*.
Koolen, M., Kaptein, R., & Kamps, J. (2009). Focused search in books and Wikipedia: Categories, links and relevance feedback. In *Proceedings of INEX*.
Krishnan, V., & Manning, C. D. (2006). An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of ACL*.
Lafferty, J., & Zhai, C. (2003). Probabilistic relevance models based on document and query generation. *In Language modeling and information retrieval*. Kluwer International Series on Information Retrieval.
Macdonald, C., & Ounis, I. (2009). Searching for expertise: Experiments with the voting model. *Computer Journal, 52*, 729–748.
McCreadie, R., Macdonald, C., Ounis, I., Peng, J., & Santos, R. L. T. (2010). University of Glasgow at TREC 2009: Experiments with terrier. In *Proceedings of TREC*.
Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, 294–304.
Rode, H., Hiemstra, D., Vries, A., & Serdyukov, P. (2009). Efficient XML and entity retrieval with PF/Tijah: CWI and University of Twente at INEX'08. In *Advances in Focused Retrieval*.
Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill.
Santos, R. L., Macdonald, C., & Ounis, I. (2010). Voting for related entities. In *Proceedings of RIAO*.
Serdyukov, P., & de Vries, A. (2010). Delft University at the TREC 2009 entity track: Ranking Wikipedia entities. In *Proceedings of TREC*.
Vercoustre, A.-M., Pehcevski, J., & Naumovski, V. (2009). Topic difficulty prediction in entity ranking. In *Advances in Focused Retrieval*.
Vydiswaran, V. V., Ganesan, K., Lv, Y., He, J., & Zhai, C. (2010). Finding related entities by retrieving relations: UIUC at TREC 2009 entity track. In *Proceedings of TREC*.
Wang, Z., Tang, C., Sun, X., Ouyang, H., Lan, R., Xu, W., et al. (2010). PRIS at TREC 2010: Related entity finding task of entity track. In *Proceedings of TREC*.
Wu, Y., & Kashioka, H. (2009). NiCT at TREC 2009: Employing three models for entity ranking track. In *Proceedings of TREC*.
Wu, Y., Hori, C., & Kawai, H. (2010). NiCT at TREC 2010: Related entity finding. In *Proceedings of TREC*.
Zhai, C., & Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM*.
Zhai, C., & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*.
Zhai, H., Cheng, X., Guo, J., Xu, H., & Liu, Y. (2010). A novel framework for related entities finding: ICT-NS at TREC 2009 entity track. In *Proceedings of TREC*.
Zheng, W., Gottipati, S., Jiang, J., & Fang, H. (2010). UDEL/SMU at TREC 2009 entity track. In *Proceedings of TREC*.