Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

# Improving Context and Category Matching for Entity Search

**Yueguo Chen[†], Lexi Gao[‡], Shuming Shi[§], Xiaoyong Du[‡], Ji-Rong Wen[‡]**
[†]Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), MOE, China
[‡]School of Information, Renmin University of China
[§]Microsoft Research Asia, China
[†‡]{chenyueguo, gaolexi, duyong, jrwen}@ruc.edu.cn, [§]shumings@microsoft.com

## Abstract

Entity search is to retrieve a ranked list of named entities of target types to a given query. In this paper, we propose an approach of entity search by formalizing both context matching and category matching. In addition, we propose a result re-ranking strategy that can be easily adapted to achieve a hybrid of two context matching strategies. Experiments on the INEX 2009 entity ranking task show that the proposed approach achieves a significant improvement of the entity search performance (xinfAP from 0.27 to 0.39) over the existing solutions.

## Introduction

Entity search has recently attracted much attention (Balog, Serdyukov, and de Vries 2011; Demartini, Iofciu, and de Vries 2009). In contrast to general web search whose goal is to retrieve a list of relevant documents, the goal of entity search, however, is to generate a short list of relevant entity names. Compared to general web search, entity search provides users more succinct answers. It has a wide range of applications such as question-and-answer (Raghavan, Allan, and Mccallum 2004), knowledge services (Weikum 2009), and web content analysis (Demartini et al. 2010).

There are some variant definitions (Cheng, Yan, and Chang 2007; Demartini, Iofciu, and de Vries 2009; Balog, Serdyukov, and de Vries 2011) of the entity search problem in terms of both inputs and outputs. The widely accepted input is a list of query words plus one or more desired entity types. In this paper, the problem is defined as: given a list of keywords or a natural language question, where the types of the target entities are explicitly specified, return a ranked list of relevant entity names of target types. We consider the above problem over a web scale entity search application which has a large number of entities ($> 10^6$) and their types ($> 10^5$) as the domain of entity search results.

Early solutions of entity search mainly take a voting strategy (Demartini, Iofciu, and de Vries 2009; Balog et al. 2009b; Santos, Macdonald, and Ounis 2010). Given a query, they first retrieve top relevant documents from a corpus. Then, entities embedded in the top relevant documents are extracted, and ranked mainly based on their occurring frequency within the retrieved documents. The voting approach (Macdonald and Ounis 2006) is applied for entity search in (Santos, Macdonald, and Ounis 2010). In the expert search domain, people have proposed a number of generative language models such as candidate model and document model (Balog, Azzopardi, and de Rijke 2006), as well as their hybrid (Balog and de Rijke 2008; Serdyukov and Hiemstra 2008). However, these models rank entities simply based on their contexts. They are therefore *context matching* solutions which are inadequate when applying to entity search because of the ignorance of entity types.

The importance of *category matching* has been verified by many solutions of entity search (Balog, Bron, and de Rijke 2010; Kaptein and Kamps 2013), where language models are typically applied to evaluate the category matching between entities and queries. There have been some recent studies that apply a linear combination of term-based (context matching) model and category-based (category matching) model (Balog, Bron, and de Rijke 2011; Raviv, Carmel, and Kurland 2012). However, such a way of hybrid may not be effective enough because of the instinctive distinction between the two models. Their reported results show that the achieved precision is still not good enough (Balog et al. 2009a; Raviv, Carmel, and Kurland 2012), when fairly compared with an alternative (Ramanathan et al. 2009) of the INEX 2009 entity ranking task.

In this paper, based on generative language modeling techniques, we propose a formal model of entity search by formalizing both context matching and category matching, and associating them more effectively. We also propose a result re-ranking strategy that can be easily adapted to achieve a hybrid of two context matching strategies. Extensive experimental results on the INEX 2009 entity ranking task demonstrate that the proposed model achieves much better empirical performance over the existing solutions, by an improvement of xinfAP from 0.27 to 0.39.

Our main contribution in this paper is 3 folds: 1) we apply and extend the existing context matching models, and effectively hybrid them using a result re-ranking technique; 2) we propose a novel approach of category matching other than existing language-model-based solutions; 3) we propose the entity model that effectively combine the proposed context matching and category matching models other than a linear combination.

## Related Work

There have been some contests for entity search, e.g., TREC 2009-2011 Entity Track (Balog, Serdyukov, and de Vries 2011), INEX 2007-2009 Entity Ranking Track (Demartini, Iofciu, and de Vries 2009). The problem definitions of entity search are somehow different in these contests. However, they can all be generalized to our problem statement.

As stated, early approaches of entity search take a voting strategy, i.e., top relevant entities are extracted and ranked from top relevant passages of the query (Fang et al. 2009; Santos, Macdonald, and Ounis 2010; Vercoustre, Thom, and Pehcevski 2008). The document model for expert search proposed by (Balog, Azzopardi, and de Rijke 2006; 2009), is similar to the voting approach (Santos, Macdonald, and Ounis 2010) where the document relevance estimation can be treated as a weight and the document-candidate association can be treated as a vote. An alternative solution of entity search is discussed in (Demartini et al. 2009), where entity profiles can be either materialized or virtualized as indexes. The candidate model proposed in (Balog, Azzopardi, and de Rijke 2006) is one of such solutions, where terms relevant to an entity can be distributed over documents. Studies (Balog and de Rijke 2008; Serdyukov and Hiemstra 2008) on combining the candidate model and document model have also been tried.

The type information has been shown to be quite important for entity search (Balog, Bron, and de Rijke 2011; Balog, Serdyukov, and de Vries 2010). Many of them (Balog, Bron, and de Rijke 2010; 2011; Kaptein and Kamps 2013) model category matching using generative language models. Solutions (Balog et al. 2009a; Ramanathan et al. 2009) achieving top results in INEX 2009 entity track (Demartini, Iofciu, and de Vries 2009) all treat category matching as an important component in their ranking functions. In particular, a linear combination of term-based model and category-based model is proposed in (Balog et al. 2009a; Balog, Bron, and de Rijke 2011). A similar solution (Raviv, Carmel, and Kurland 2012) also applies a linear combination of term matching, category matching, and entity name matching. A recent work (Raviv, Kurland, and Carmel 2013) applies a re-ranking technique that utilizes the similarities among relevant entities to improve the retrieval performance. However, as far as we know, the reported precision of these solutions is still very limited.

## The Entity Model

### Entity Model

In our study, an entity search query $q$ consists of two components $q = \{T, C\}$, where $T$ is the set of query terms describing the desired entities, and $C$ is the set of categories (although $C$ often contains only one category) specified for relevant entities. When $C$ contains more than one categories, a desired entity need to belong to at least one category. For example, a query in the INEX 2009 entity ranking task (Demartini, Iofciu, and de Vries 2009) includes query terms $T = \{$*works by Charles Rennie Mackintosh*$\}$, and categories $C = \{$*buildings, structures*$\}$, which is to retrieve buildings or structures designed by Charles Rennie Mackintosh.

We formalize the relevance scores of entities as the probability of a candidate entity $e$ being a qualified entity of the query $q$, i.e., $p(e|q)$. According to the Bayes's Theorem:

$$p(e|q) = \frac{p(q|e)p(e)}{p(q)} \tag{1}$$

where $p(q|e)$ is the probability of generating the query $q$ given the entity $e$, $p(e)$ is probability of an entity, and $p(q)$ is the probability of a query. We assume that $p(e)$ is uniformly distributed over all entities to be retrieved. $p(q)$ is fixed for a given query $q$. As such, we can use the $p(q|e)$ to evaluate $p(e|q)$ as they are in proportion. Inspired by the assumption of conditional independency among the query terms in the standard language model, we propose an entity model by assuming the conditional independency between the query terms $T$ and the query categories $C$:

$$p(q|e) = p(T, C|e) = p(T|e)p(C|e) \tag{2}$$

where $p(T|e)$ is the probability of generating the query terms $T$ given the entity $e$, and $p(C|e)$ is the probability of generating the query categories $C$ from $e$. The relevance score of an entity is then proportional to the product of $p(T|e)$ and $p(C|e)$. Note that this is quite different from the other solutions (Balog, Bron, and de Rijke 2011; Raviv, Carmel, and Kurland 2012) that apply a linear combination of the probabilities of term-based and category-based models, whose values are usually not in the same order. In the followings of this section, we first introduce the probabilistic models for these two components (called *context matching* and *category matching*) respectively.

### Context Matching

**Long-Range Context Matching** In the document model (Balog, Azzopardi, and de Rijke 2006), the component $p(T|e)$ is estimated as follows:

$$p(T|e) = \sum_d p(T|d)p(d|e) \tag{3}$$

where $p(T|d)$ is the probability of generating the query terms $T$ from a document $d$, and $p(d|e)$ is the probability of generating the document $d$ from an entity $e$. The component $p(T|d)$ is estimated using a standard language model:

$$p(T|d) = \prod_{t \in T} ((1-\lambda)p(t|d) + \lambda p(t))^{n(t,T)} \tag{4}$$

The component $p(t|d)$ is the probability of generating the term $t$ from a document $d$. It is the maximum likelihood estimate of the term $t$ in $d$. $p(t)$ is a background probability of term $t$ over all documents. It is used for smoothing $p(t|d)$ considering that $p(t|d)$ can be zero. $n(t, T)$ is the times of $t$ appears in $T$.

The component $p(d|e)$ of Eq. 3 is evaluated as the document-to-entity association, where $a(d, e)$ is the frequency of entity $e$ occurring in the document $d$. The exact candidate-centric function which has been shown to be better than the document-centric one (Balog, Azzopardi, and de Rijke 2006) in expert search, is applied.

$$p(d|e) = \frac{a(d, e)}{\sum_{e' \in d} a(d, e')} \tag{5}$$

The above document model captures long-range contexts of entities. It however sacrifices the term proximity between query terms and the relevant candidates of entities. To address this, we propose a profiling approach that effectively utilizes the short-range contexts of candidates in documents for building entity profiles and ranking entities.

**Short-Range Context Matching** We propose to create context profiles for entities, so that those relevant short-range contexts can be aggregated in the context profiles, from which term proximity is considered for ranking entities. For each entity $e$, a context profile is created by concatenating a number of highly relevant contexts that $e$ occurs. The context profile of an entity $e$ is denoted as $X_e$. With the context profile, $p(T|e)$ will be estimated as the probability of generating query terms $T$ from the context profile $X_e$:

$$p(T|e) \propto \prod_{t \in T}((1-\lambda)p(t|X_e) + \lambda p(t))^{n(t,T)} \tag{6}$$

$$p(t|X_e) = \frac{\sum_{x \in X_e} a(x,t)a(x,e)}{|X_e|} \tag{7}$$

where $p(t)$ is a background probability of term $t$ over all profiles. $p(t|X_e)$ is the probability of generating the term $t$ from the context profile $X_e$. $a(x,t)$ is the frequency of the term $t$ appearing in a context $x$, $a(x,e)$ is the frequency of the entity $e$ appearing in the context $x$, and $|X_e|$ is the number of words in the profile $X_e$. Compared with the candidate model (model 1 proposed in (Balog, Azzopardi, and de Rijke 2006)), the profiling approach has several distinct characteristics: 1) it only considers short-range contexts of entities to utilize term proximity when ranking entities; 2) the term frequency is normalized by the size of an entity profile in the profile approach; 3) the contexts of an entity are selected for effectively building the context profile of the entity.

A straightforward way of generating the context profile of an entity is to combine all the sentences (contexts) that contain the entity. However, the sentences will be evenly treated if we lose the association of sentences and entities, although their relevance to the entities can be quite different. To address it, we propose to mine and select entity contexts from a web corpus. We partition all top relevant documents (according to $p(T|d)$) into sentences. A context of an entity $e$ is a sentence consisting of a mention of the entity $e$, which can be identified by an entity labeling tool, e.g., Wikipedia-Miner (Milne and Witten 2008). To build the context profile $X_e$ of an entity $e$, we select contexts following two rules: 1) when a context $x$ of an entity $e$ contains an IsA pattern (Zhang et al. 2011) for the entity $e$, it will be selected as a context of $X_e$ because it may contain a definition of $e$; 2) contexts in the top-$h$ relevant documents.

## Category Matching

To effectively evaluate the relevance of entities based on category matching, for each entity $e$, we build a category profile $C_e$ recording the categories that $e$ belongs to. Note that the category profile $C_e$ can be empty for an entity whose categories are hard to mine.

**Category Matching Model** Given the query categories $C$ and the categories $C_e$ of an entity $e$, a straightforward way of category matching is to derive a binary decision based on whether $C_e \bigcap C$ is an empty set or not. However, this often causes false negatives because: 1) a category in $C$ can be a very detailed category (e.g., *Japanese baseball teams*), which is hard to be exactly mined from the corpus for some entities; 2) many tailed entities may not even have a category profile, although they can be an instance of a query category.

An alternative strategy of estimating $p(C|e)$ is to apply a generative language model to estimate the match of categories in $C$ and $C_e$ (Balog, Bron, and de Rijke 2010; Kaptein and Kamps 2013). However, such an estimating strategy is not effective because the texts of categories in $C_e$ are very short. On the other hand, if we simply estimate $p(C|e)$ by computing the textual similarity of categories in $C$ and $C_e$, e.g., using the Jaccard similarity of term sets, it still faces with the problem of empty profile set. Moreover, it will generate more false positives, and diminish the difference between type-matched entities and non matched ones.

A category (either in $C$ or $C_e$) is assumed as a phrase specifying a type of entities. It can be further interpreted as a head word plus some additional qualifiers (Ramanathan et al. 2009). The head word typically specifies a general category, and the qualifiers restrict the general category so that it can be more specific. For example, in the query category *Japanese baseball teams*, the head word can be *teams*, and the qualifiers are *Japanese* and *baseball*. Obviously, the more qualifiers a category has, the more fine-grained the category is. We use $c.d$ to denote the head word of a category $c$, and $c.Q$ to denote the set of qualifiers of $c$.

Given two categories $c_1$ and $c_2$, we say $c_1$ is a supercategory of $c_2$, denoted as $c_1 \preceq c_2$ if both $c_1.d = c_2.d$ and $c_1.Q \subseteq c_2.Q$ satisfy. The two categories $c_1$ and $c_2$ are a pair of matching categories if either $c_1 \preceq c_2$ or $c_2 \preceq c_1$. For example, *Japanese baseball teams* and *baseball teams* are a pair of matching categories simply because *baseball teams* is a supercategory of *Japanese baseball teams*. However, *Japanese baseball teams* and *national baseball teams* are not a pair of matching categories.

We denote the common category between two categories $c_1$ and $c_2$, $m(c_1, c_2)$, as:

$$m(c_1,c_2) = \begin{cases} c_1 & if\ c_1 \preceq c_2 \\ c_2 & else\ if\ c_2 \preceq c_1 \\ NIL & \text{otherwise} \end{cases} \tag{8}$$

Obviously, if $c_1$ and $c_2$ are a pair of matching categories, the common category $c = m(c_1, c_2)$ is the general category between them. If they are not a pair of matching categories, the common category will be $NIL$. The $NIL$ is defined as the most coarse-grained category, that can be the supercategory of all categories, i.e., $NIL \preceq c$ for any category $c$.

Let $E$ be the set of all possible named entities (the domain of entities) that can be retrieved by the entity search system. Given a category $c$, we denote $g(c)$ be the number of all entities (in $E$) that have at least one category $c'$ satisfying that $c \preceq c'$. Obviously, the finer granularity a category $c$ has, the less the $g(c)$. When $c = NIL$, we have $g(NIL) = |E|$ simply because $NIL \preceq c$ for any category $c$.

With the definition of the common category, we are able to estimate $p(C|e)$ in the entity model as:

$$p(C|e) = \max_{c_1 \in C, c_2 \in C_e} \frac{1}{g(m(c_1, c_2))} \quad (9)$$

Note that $p(C|e) = \frac{1}{|E|}$ when $C_e = \emptyset$. Given a query category set $C$ and an entity $e$, the above estimation basically is to find the most fine-grained common category from categories of $C$ and those of $C_e$ such that $g(m(c, e))$ can be minimized. The most fine-grained common category is supposed to have the best discriminability to distinct the entity $e$ from others simply based on the category matching to $C$.

For each entity, we create a category profile consisting of categories extracted from the Wikipedia collection. Considering that some entities do not have enough explicit Wikipedia categories, we apply category mining techniques to mine proper categories of entities from the ClueWeb09 corpus[1]. There have been a number of approaches (Durme and Pasca 2008; Snow, Jurafsky, and Ng 2004; Zhang et al. 2011) for automatically mining categories/hypernyms for entities from a large collection of documents, based on some text patterns. Following the category mining solution of (Zhang et al. 2011), we mine entity-category pairs with the patterns below:

Hearst: *NPC*{,} *(such as|including)* {*NP*,} * {*and*|*or*} *NP*

IsA: *NP (is|are|was|were|being) (a|an|the|any|another) NPC*

where NP and NPC respectively represent an entity and a category name. An entity-category pair is detected if the frequency of patterns that witness the type association of the category $c$ and the entity $e$ is large enough. Considering that the quality of mined categories will not be as good as that of explicit Wikipedia categories (labelled by human), we apply at most (if it has) the top-3 mined categories to each entity.

### Search Result Re-Ranking

Although category matching is applied in the proposed entity model, we may find that many entities of irrelevant types still appear in the top results of some queries. These entities are usually popular entities that frequently occur in many documents and contexts. Their context relevance can be very high even though they are not matched to any query category. It has been shown that a re-ranking technique that utilizes the similarities of top relevant entities can improve the search performance (Raviv, Kurland, and Carmel 2013). Inspired by this observation, we propose to re-rank the search results based on the coherence of entity categories among the top results, using the following equation:

$$r(e) = \sum_{e' \in R_k} \frac{J(C_e, C_{e'})}{f(e, e')} \sqrt{p(q|e)p(q|e')} \quad (10)$$

where $r(e)$ is the relevance score of $e$ after the re-ranking process, $R_k$ is the set of top-$k$ results of the entity model, and $J(C_e, C_{e'})$ is the Jaccard similarity of the category sets of two entities. The component $f(e, e') = g(e)$ if $e$ is a category of $e'$ (note that an entity can also be a category of the other entities), otherwise $f(e, e') = 1$. Note that $f(e, e')$ is designed to suppress entities of general types.

The way we design the re-ranking strategy as Eq. 10 has a merit that can be utilized to achieve an effective hybrid solution of long-range context matching (LRCM) and short-range context matching (SRCM). Note that two alternatives of context matching, LRCM and SRCM, are expected to complement for each other. However, they cannot be easily combined using a linear combination because the derived probability $p(T|e)$ (in Eq. 3 and Eq. 6) may not be in the same order due to the different ranking mechanisms. Based on the proposed re-ranking strategy (Eq. 10), we achieve a simple hybrid of LRCM and SRCM as follows:

$$\begin{aligned} r(e) &= \sum_{e' \in L_k} \frac{J(C_e, C_{e'})}{f(e, e')} \sqrt{s(e)l(e')} \\ &+ \sum_{e' \in S_k} \frac{J(C_e, C_{e'})}{f(e, e')} \sqrt{l(e)s(e')} \quad (11) \end{aligned}$$

where $l(e)$ and $s(e)$ are the relevance scores of $e$ for LRCM and SRCM respectively (where category matching and the re-ranking strategy may be applied), and $L_k$ and $S_k$ are the sets of top-$k$ results for LRCM and SRCM respectively.

## Experimental Evaluation

We apply some labels to clarify the strategies used in the experiments. $L$ and $S$ denotes simple LRCM and SRCM respectively. $C$ denotes the category matching. $R$ stands for the re-ranking strategy. The combination of these notations explains how the strategies are applied for entity ranking. For example, $LC$ stands for the entity model of LRCM; $LCR$ means to re-rank the results of $LC$; $LCR + SCR$ indicates a hybrid of $LCR$ and $SCR$ using the Eq. 11.

### Experimental Settings

**Document Collection** We adopt a public-available document collection in our experiments: Wikipedia INEX 2009 collection[2] (shorted as INEX09). It contains 2.67 millions Wikipedia XML articles. It is created from the October 8, 2008 dump of the English Wikipedia articles. The whole dataset provides semantic markup of articles and outgoing links, based on the semantic knowledge base YAGO (Weikum and Theobald 2010). It explicitly labels more than 5,800 classes of entities like persons, movies, cities, and many more. The categories of Wiki articles have also been extracted as the topics of articles and the types of entities if the titles of articles can be treated as entity names.

Since 2009, INEX uses this dataset as the basic dataset. Considering that entities in many other Web corpus are not explicitly labeled, we therefore ignore all the annotated entities and their Wiki categories by treating the INEX09 collection as a corpus of pure plain texts. The reason we use INEX09 collection is twofold: 1) it has been a benchmark for the INEX 2009 entity ranking task (Demartini, Iofciu, and de Vries 2009). We can compare the performance of our solution with those proposed in INEX 2009, although it is

[1]http://boston.lti.cs.cmu.edu/Data/clueweb09/

[2]http://www.mpi-inf.mpg.de/departments/d5/software/inex/

Table 1: long-range context matching

| $h$ | p@5 | p@10 | p@20 | MRR | R-pre | xinfAP |
|---|---|---|---|---|---|---|
| L: without category matching | | | | | | |
| 100 | 0.065 | 0.076 | 0.088 | 0.136 | 0.100 | 0.071 |
| 200 | 0.062 | 0.076 | 0.091 | 0.133 | 0.099 | 0.071 |
| 300 | 0.055 | 0.075 | 0.091 | 0.131 | 0.100 | 0.072 |
| 500 | 0.051 | 0.075 | 0.092 | 0.129 | 0.101 | 0.072 |
| 1000 | 0.051 | 0.075 | 0.093 | 0.128 | 0.101 | 0.073 |
| LC: with category matching | | | | | | |
| 100 | 0.451 | 0.422 | 0.358 | 0.612 | 0.330 | 0.284 |
| 200 | 0.451 | 0.425 | 0.363 | **0.612** | 0.334 | 0.289 |
| 300 | **0.451** | **0.425** | 0.369 | 0.610 | 0.339 | **0.292** |
| 500 | 0.440 | 0.422 | 0.369 | 0.589 | **0.340** | 0.291 |
| 1000 | 0.444 | 0.422 | **0.370** | 0.592 | 0.340 | 0.292 |
| (Balog, Bron, and de Rijke 2011) baseline (M5) | | | | | | |
| Other | 0.193 | 0.165 | 0.152 | 0.259 | 0.163 | 0.152 |

Table 2: short-range context matching

| $h$ | p@5 | p@10 | p@20 | MRR | R-pre | xinfAP |
|---|---|---|---|---|---|---|
| S: without category matching | | | | | | |
| 100 | 0.095 | 0.089 | 0.083 | 0.197 | 0.086 | 0.063 |
| 200 | 0.087 | 0.089 | 0.085 | 0.200 | 0.083 | 0.062 |
| 300 | 0.084 | 0.087 | 0.078 | 0.192 | 0.081 | 0.061 |
| 500 | 0.073 | 0.084 | 0.077 | 0.192 | 0.076 | 0.060 |
| 1000 | 0.084 | 0.075 | 0.074 | 0.170 | 0.076 | 0.057 |
| SC: with category matching | | | | | | |
| 100 | 0.411 | 0.362 | 0.307 | 0.569 | 0.276 | 0.210 |
| 200 | 0.422 | 0.373 | 0.306 | 0.574 | 0.287 | 0.225 |
| 300 | 0.436 | 0.385 | 0.317 | **0.583** | **0.303** | 0.242 |
| 500 | **0.436** | **0.396** | 0.319 | 0.581 | 0.298 | 0.248 |
| 1000 | 0.429 | 0.373 | **0.320** | 0.562 | 0.293 | **0.250** |
| (Balog, Bron, and de Rijke 2011) with blind feedback | | | | | | |
| Other | 0.184 | 0.182 | 0.162 | 0.246 | 0.154 | 0.149 |

not fair for our solution because of the ignorance of semantic labels; 2) as stated in (Kaptein et al. 2010), the number of entities in INEX09 is large.

**Query Set** We use the query set that has been used in INEX 2009 entity ranking task. There are overall 55 queries. Each query contains information of four domains, *title*, *description*, *narrative*, *categories*, although only the *title* and *categories* domains are used. Each query desires a ranked list of relevant entities (in terms of the *title*) of the types which are explicitly specified in the *categories*. An example of the queries we used is as follows:

*title*: Tom Hanks movies where he plays a leading role.
*categories*: movies, films.

The test result set of INEX09 contains lists of answers for all of the queries, with each query having dozens of relevant entities. Each entity has an entity ID within the INEX09 document collection. There is a Wiki article in INEX09, corresponding to each entity.

**Evaluation Metrics** The evaluation score on a query set is the average over all the topics. We adopt the metrics:

- $p@k$: percentage of relevant entities in the top-$k$ results.
- MRR: the mean reciprocal rank.
- $R$-pre: $p@R$ where $R$ is the number of given correct results for a query.
- xinfAP: proposed by (Yilmaz, Kanoulas, and Aslam 2008) and used as an official evaluation metric by the INEX09ER (Demartini, Iofciu, and de Vries 2009) and many following studies of the entity ranking task.

**Entity Extraction** We obtain a list of 2.36M entities from Wikipedia titles by removing the texts of titles within brackets. To extract entities, we utilize an open source toolkit Wikipedia-Miner (Milne and Witten 2008), which takes the unstructured texts as inputs and detects the Wikipedia concepts in the inputs using machine learning techniques.

## Performance of The Entity Model

To test the performance of the proposed entity model, we compare the results of experiments when only context matching is applied ($L$ and $S$) with those when both context matching and category matching are applied ($LC$ and $SC$). The results for LRCM and SRCM are shown in Table 1 and Table 2 respectively. We also report the results of our implementation of the entity ranking methods proposed in (Balog, Bron, and de Rijke 2011) as baselines. Note that the results on INEX 2009 dataset of (Balog, Bron, and de Rijke 2011) are actually reported in (Balog et al. 2009a), which achieves an xinfAP of 0.189 for the baseline, and 0.209 when blind feedback is applied. The other runs (reported in (Demartini, Iofciu, and de Vries 2009) and (Balog et al. 2009a)) whose xinfAP is higher than 0.27 are based on query dependent feedbacks where human knowledge on previous test results is utilized (Balog et al. 2009a). They are therefore not reported here to make sure the comparison is fairly conducted.

According to the results of Table 1 and Table 2, we can obviously find that the performance of the entity model ($LC$ and $SC$) is much better than that of simply using context matching. When category matching is applied, the xinfAP of LRCM is significantly improved from 0.07 to 0.29, and that of SRCM is improved from 0.06 to 0.25. The achieved xinfAP of $LC$ (0.292) is higher than those of INEX 2009 (Demartini, Iofciu, and de Vries 2009) (top as 0.27), as well as those of (Balog, Bron, and de Rijke 2011) and (Raviv, Carmel, and Kurland 2012) (top as 0.258). The advantage of the entity model is obviously observed.

When analyzing the results of $L$ and $S$, we find that $L$ outperforms $S$ slightly, which convinces the observation of (Balog, Azzopardi, and de Rijke 2006) that the document model outperforms the candidate model. An interesting phenomenon of $L$ is that the precision of $p@10$ and $p@20$ is higher than that of $p@5$. This is simply because popular entities are often among the top results of $L$ as they frequently occur in many documents, although they are not entities of desired types. The problem is partially solved when category matching is applied ($LC$). Note that although $SC$ is not as good as $LC$, as will be shown in the following experiments, their combination will further improve the performance.

For the parameter $h$ ($L$ uses it for retrieving top-$h$ relevant documents of a topic), we set $h = 300$ by default for LRCM and SRCM because larger $h$ can only improve the precision

Table 3: LCR: re-ranking for LRCM

| $k$ | p@5 | p@10 | p@20 | MRR | R-pre | xinfAP |
|---|---|---|---|---|---|---|
| 5 | 0.553 | 0.502 | 0.418 | 0.642 | 0.375 | 0.356 |
| 10 | 0.545 | 0.495 | 0.419 | 0.662 | 0.383 | 0.355 |
| 20 | 0.556 | 0.498 | **0.424** | 0.689 | **0.390** | **0.358** |
| 30 | 0.549 | **0.504** | 0.420 | 0.688 | 0.387 | 0.357 |
| 50 | **0.567** | 0.496 | 0.417 | **0.703** | 0.382 | 0.353 |
| LC | 0.451 | 0.425 | 0.369 | 0.610 | 0.339 | 0.292 |

Table 4: SCR: re-ranking for SRCM

| $k$ | p@5 | p@10 | p@20 | MRR | R-pre | xinfAP |
|---|---|---|---|---|---|---|
| 5 | 0.509 | 0.476 | 0.388 | **0.664** | 0.355 | **0.310** |
| 10 | 0.513 | **0.476** | 0.389 | 0.633 | **0.358** | 0.307 |
| 20 | 0.513 | 0.473 | **0.389** | 0.633 | 0.357 | 0.305 |
| 30 | 0.513 | 0.471 | 0.387 | 0.650 | 0.357 | 0.306 |
| 50 | **0.516** | 0.473 | 0.387 | 0.653 | 0.352 | 0.303 |
| SC | 0.436 | 0.385 | 0.317 | 0.583 | 0.303 | 0.242 |

Table 5: LCR+SCR: the hybrid solution

| $k$ | p@5 | p@10 | p@20 | MRR | R-pre | xinfAP |
|---|---|---|---|---|---|---|
| 5 | 0.640 | 0.542 | 0.438 | 0.737 | 0.415 | 0.383 |
| 10 | 0.640 | **0.545** | 0.443 | 0.734 | 0.412 | 0.386 |
| 20 | **0.655** | 0.545 | **0.447** | **0.740** | 0.416 | **0.390** |
| 30 | 0.651 | 0.542 | 0.445 | 0.738 | **0.416** | 0.390 |
| 50 | 0.644 | 0.536 | 0.445 | 0.737 | 0.415 | 0.388 |

Table 6: an overall comparison

| solution | p@5 | p@10 | p@20 | MRR | R-pre | xinfAP |
|---|---|---|---|---|---|---|
| LC | 0.451 | 0.425 | 0.369 | 0.610 | 0.339 | 0.292 |
| LCR | 0.549 | 0.504 | 0.420 | 0.688 | 0.387 | 0.357 |
| LC+SC | 0.553 | 0.518$^{\triangle}$ | 0.432 | 0.684 | 0.399 | 0.356 |
| (LC+SC)R | 0.647$^{\triangle}$ | **0.564**$^{\triangle}$ | 0.446 | 0.724 | 0.410$^{\triangle}$ | 0.384$^{\triangle}$ |
| LCR+SCR | **0.655**$^{\triangle}$ | 0.545$^{\triangle}$ | **0.447** | **0.740**$^{\triangle}$ | **0.416**$^{\triangle}$ | **0.390**$^{\triangle}$ |

very slightly. For the parameter $\lambda$, we adjust it and plot the results of $L$, $S$, $LC$ and $SC$ in Figure 1 respectively. It can be seen that the precision is stable for a wide range of $\lambda$. As such, we simply set $\lambda = 0.5$ for all the other experiments.

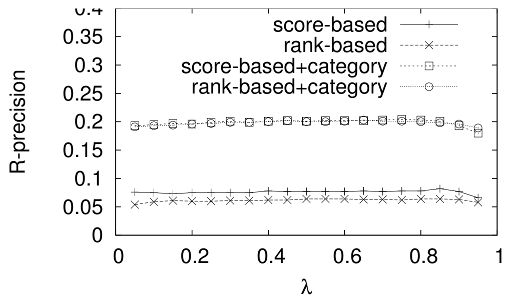

Figure 1: The impacts of the parameter $\lambda$

## Performance of The Re-Ranking Strategies

In (Raviv, Kurland, and Carmel 2013), it was shown that a re-ranking technique can improve the entity search performance slightly. We test the performance of our proposed re-ranking technique. The results for $LCR$ and $SCR$ are shown in Table 3 and Table 4 respectively. Compared with those of $LC$ and $SC$, it can be seen that our re-ranking technique significantly improves the search performance. For the xinfAP metric, the performance can be improved from 0.292 to 0.358 for $LC$, and from 0.242 to 0.31 for $SC$. Especially for $LC$, the improvement on $p@5$ and MRR is quite significant, showing the re-ranking solution does help to suppress irrelevant top results of $LC$. For the parameter $k$, we can see that $SCR$ appeals to small $k$, and $LCR$ achieves the best performance when $k$ is a bit larger (e.g., 20). This is reasonable because $SC$ is less accurate than $LC$. Larger $k$ in $SC$ means more feedbacks from false positive entities.

The results in Table 5 show that the hybrid solution ($LCR + SCR$) further improves the search performance, for xinfAP from 0.358 ($LCR$) to 0.39. This is a very high entity search performance, that outperforms the best of the published results, 0.27 as far as we know, very significantly.

For the parameter $k$ of $LCR+SCR$, we find that the best performance is achieved when $k = 20$ in our experiments. Less $k$ means that not enough top results are used for re-ranking. More $k$ means that the impacts from false positive entities has surpassed the impacts of positive ones.

To further compare the performance of various re-ranking and the hybrid strategies, we also conduct another two experiments for $LC + SC$ (where $l()$ and $s()$ in Eqn. 11 are the relevance scores for $LC$ and $SC$ respectively) and $(LC+SC)R$ respectively, where $LC+SC$ is a hybrid of $LC$ and $SC$, and $(LC + SC)R$ applies the re-ranking strategy to the results of $LC + SC$. The results, compared with other alternatives of re-ranking and hybrid strategies, are shown in Table 6. Note that in this test, we treat $LC$ as the baseline. We compare the performance of the strategies using a one-tailed t-test at a significance level of $p = 0.05$. The notation $^{\triangle}$ in Table 6 denotes significant difference over the baseline.

As we can see from the results of Table 6, a simple hybrid $LC + SC$ of $LC$ and $SC$ can achieve a similar performance to $LCR$. When a re-ranking process is applied to $LC + SC$, the performance can be further improved by $(LC + SC)R$, which is similar to that of $LCR + SCR$.

We also test the performance of the above strategies when only Wikipedia categories are applied for building category profile. For $LC$ and $SC$, the performance drops very slightly. The xinfAP drops from 0.357 to 0.345 for $LCR$, from 0.39 to 0.374 for $LCR+SCR$, and from 0.356 to 0.35 for $LC + SC$. The results show that the re-ranking process benefits a bit from the introduction of mined categories.

## Conclusion

We address the entity search problem by proposing a formal model which effectively combines both context matching and category matching, as well as a result re-ranking strategy that can be easily adapted to achieve a hybrid of two proposed context matching alternatives. Extensive experiments on the INEX 2009 entity ranking task show that the proposed entity model and the re-ranking strategies can significantly improve the entity search performance over the existing solutions of entity search.

## Acknowledgments

The authors would like to acknowledge Yu Sun for his comments and feedback. This work is partially supported by National Basic Research Program of China (973 Program) No. 2012CB316205, the National Science Foundation of China under grant NO.61170010, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China, and the Chinese National "111" Project "Attracting International Talents in Data Engineering and Knowledge Engineering Research".

## References

Balog, K., and de Rijke, M. 2008. Combining candidate and document models for expert search. In *TREC*.

Balog, K.; Azzopardi, L.; and de Rijke, M. 2006. Formal models for expert finding in enterprise corpora. In *SIGIR*, 43–50.

Balog, K.; Azzopardi, L.; and de Rijke, M. 2009. A language modeling framework for expert finding. *Inf. Process. Manage.* 45(1):1–19.

Balog, K.; Bron, M.; de Rijke, M.; and Weerkamp, W. 2009a. Combining term-based and category-based representations for entity search. In *INEX*, 265–272.

Balog, K.; Serdyukov, P.; Vries, A. P. D.; and Thomas, P. 2009b. Overview of the trec 2009 entity track. In *TREC*.

Balog, K.; Bron, M.; and de Rijke, M. 2010. Category-based query modeling for entity search. In *ECIR*, 319–331.

Balog, K.; Bron, M.; and de Rijke, M. 2011. Query modeling for entity search based on terms, categories, and examples. *ACM TOIS* 29(4):22.

Balog, K.; Serdyukov, P.; and de Vries, A. P. 2010. Overview of the trec 2010 entity track. In *TREC*.

Balog, K.; Serdyukov, P.; and de Vries, A. P. 2011. Overview of the trec 2011 entity track. In *TREC*.

Cheng, T.; Yan, X.; and Chang, K. C.-C. 2007. Entityrank: Searching entities directly and holistically. In *VLDB*, 387–398.

Demartini, G.; Firan, C. S.; Georgescu, M.; Iofciu, T.; Krestel, R.; and Nejdl, W. 2009. An architecture for finding entities on the web. In *LA-WEB/CLIHC*, 230–237.

Demartini, G.; Missen, M. M. S.; Blanco, R.; and Zaragoza, H. 2010. Taer: time-aware entity retrieval-exploiting the past to find relevant entities in news articles. In *CIKM*, 1517–1520.

Demartini, G.; Iofciu, T.; and de Vries, A. P. 2009. Overview of the inex 2009 entity ranking track. In *INEX*, 254–264.

Durme, B. V., and Pasca, M. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *AAAI*.

Fang, Y.; Si, L.; Yu, Z.; Xian, Y.; and Xu, Y. 2009. Entity retrieval with hierarchical relevance model. In *TREC*.

Kaptein, R., and Kamps, J. 2013. Exploiting the category structure of wikipedia for entity ranking. *Artif. Intell.* 194.

Kaptein, R.; Serdyukov, P.; de Vries, A. P.; and Kamps, J. 2010. Entity ranking using wikipedia as a pivot. In *CIKM*, 69–78.

Macdonald, C., and Ounis, I. 2006. Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM*, 387–396.

Milne, D. N., and Witten, I. H. 2008. Learning to link with wikipedia. In *CIKM*, 509–518.

Raghavan, H.; Allan, J.; and Mccallum, A. 2004. An exploration of entity models, collective classification and relation descriptions. In *in LinkKDD 2004*.

Ramanathan, M.; Rajagopal, S.; Karthik, S. V.; Murugeshan, M. S.; and Mukherjee, S. 2009. A recursive approach to entity ranking and list completion using entity determining terms, qualifiers and prominent n-grams. In *INEX*, 292–302.

Raviv, H.; Carmel, D.; and Kurland, O. 2012. A ranking framework for entity oriented search using markov random fields. In *JIWES*, 1:1–1:6.

Raviv, H.; Kurland, O.; and Carmel, D. 2013. The cluster hypothesis for entity oriented search. In *SIGIR*, 841–844.

Santos, R. L. T.; Macdonald, C.; and Ounis, I. 2010. Voting for related entities. In *RIAO*, 1–8.

Serdyukov, P., and Hiemstra, D. 2008. Modeling documents as mixtures of persons for expert finding. In *ECIR*, 309–320.

Snow, R.; Jurafsky, D.; and Ng, A. Y. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Vercoustre, A.-M.; Thom, J. A.; and Pehcevski, J. 2008. Entity ranking in wikipedia. In *SAC*, 1101–1106.

Weikum, G., and Theobald, M. 2010. From information to knowledge: harvesting entities and relationships from web sources. In *PODS*, 65–76.

Weikum, G. 2009. Search for knowledge. In *SeCO Workshop*, 24–39.

Yilmaz, E.; Kanoulas, E.; and Aslam, J. A. 2008. A simple and efficient sampling method for estimating ap and ndcg. In *SIGIR*, 603–610.

Zhang, F.; Shi, S.; Liu, J.; Sun, S.; and Lin, C.-Y. 2011. Nonlinear evidence fusion and propagation for hyponymy relation mining. In *ACL*, 1159–1168.