Leveraging Fine-Grained Wikipedia Categories for Entity Search

Denghao Ma
Renmin University of China
madenghao@ruc.edu.cn

Yueguo Chen*
Renmin University of China
chenyueguo@ruc.edu.cn

Kevin Chen-Chuan Chang
UIUC
kcchang@illinois.edu

Xiaoyong Du
Renmin University of China
duyong@ruc.edu.cn

Chuanfei Xu
Huawei Technologies Co. Ltd
xuchuanfei@huawei.com

Yi Chang
Huawei Research America
Yi.Chang@huawei.com

ABSTRACT

Ad-hoc entity search, which is to retrieve a ranked list of relevant entities in response to a query of natural language question, has been widely studied. It has been shown that category matching of entities, especially when matching to fine-grained entity types/categories, is critical to the performance of entity search. However, the potentials of the fine-grained Wikipedia entity categories, has not been well exploited by existing studies. Based on the observation of how people describe entities of a specific type, we propose a headword-and-modifier model to deeply interpret both queries and fine-grained entity types/categories. Probabilistic generative models are designed to effectively estimate the relevance of headwords and modifiers as a pattern-based matching problem, taking the Wikipedia type taxonomy as an important input to address the ad-hoc representations of concepts/entities in queries. Extensive experimental results on three widely-used test sets: INEX-XER 2009, SemSearch-LS and TREC-Entity, show that our method achieves a significant improvement of the entity search performance over the state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Retrieval models and ranking**;

KEYWORDS

Entity search; Category matching; Language model

ACM Reference Format:

Denghao Ma, Yueguo Chen, Kevin Chen-Chuan Chang, Xiaoyong Du, Chuanfei Xu, and Yi Chang. 2018. Leveraging Fine-Grained Wikipedia Categories for Entity Search. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3178876.3186074>

1 INTRODUCTION

A common definition of ad-hoc (list) entity search is to retrieve a ranked list of relevant entities in response to an entity search query [8, 11–13]. For example, for a query *Japanese players in Major*

League Baseball, we may expect to retrieve a list of Japanese baseball players such as *Hideki Matsui*, *Kazuhiko Sasaki*, *Ichiro Suzuki* who have ever played in Major League Baseball as immediate answers. This is a large difference from general web search engines which mainly return relevant documents as search results.

Early studies of entity search [13] assume that at least one target type (also called category alternatively throughout the paper) of relevant entities has been explicitly specified in the query. This is important because many studies [2, 8, 16] have shown that category matching plays a very important role for finding relevant entities. However, in many situations, providing explicit target entity types in a query leads to cognitive overhead to users, especially when they are not familiar with the underlying type taxonomy of entities to be retrieved. This motivates some studies [2, 15, 16] by assuming that a query topic is only a natural language question. They address the absence of the target entity types by retrieving top-*k* relevant entity types of the query and treating them as substitutes of the missing target entity types. Consequently, existing entity search methods, achieving category matching by computing the relevance of entity types/categories to target entity types, still work even without explicit target entity types in the query.

To find the top-*k* relevant entity types of a natural language question is then a critical component of category matching. Balog et al. [2] apply a bag-of-words model to evaluate the relevance of entity types to a query using language models. However, such a term-centric method does not perform well because the BOW based language models are often not effective enough to estimate the relevance of short texts where vocabulary gaps commonly exist. Kaptein et al. [16, 17] propose an entity-centric method to vote relevant entity types from the top relevant entities retrieved from the query. This however does not perform well because some popular and general types/categories are likely to be more relevant. Balog et al. [4] also try to rank entity types based on their profiles (type-centric method) built from the entities of each entity type, but find that it is not as good as entity-centric methods. A learning to rank approach [15] has also been tried to apply more features for ranking entity types. The target entity types retrieved by this method however tend to be general entity types.

The above category matching framework of many entity search methods has some shortcomings: 1) The category matching models are mainly lexicon-based. Important concepts and entities, which are critical for interpreting the query intent as well as the type semantics, are not identified from queries and entity types. The pure term-based category matching methods therefore lead to a low recall because users may present their intent using alternative

*Corresponding author.

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW 2018, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5639-8/18/04.

<https://doi.org/10.1145/3178876.3186074>

concepts/entities. 2) Due to the limitation of existing ranking models of category matching, many relevant entity types of a query are incorrect interpretations of the query intent [15], whose impact on the entity search performance will be further amplified when they are used as target entity types of the query.

The Wikipedia collection has been widely used by many entity search methods [2, 8, 16], as the underlying corpus to support entity search. Entities (Wiki documents) in Wikipedia are collectively labelled with many fine-grained types/categories, which provide much semantics to the entities. They have been exploited by some recent studies [9, 14, 16], which show that the performance of entity search benefits a lot from the usage of fine-grained Wikipedia categories. However, we believe that existing category matching methods [2, 8, 16] do not unleash the power of fine-grained Wikipedia categories due to the above mentioned shortcomings.

We observe that in many cases of list entity search, people try to find entities of a specific type, whose intent can usually be represented in a similar way of creating fine-grained Wikipedia categories. They tend to use a general concept to describe the basic type of relevant entities, and then apply some terms/entities to constrain the type to be a more specific one. As our key idea of this paper, we propose to model both queries and fine-grained entity categories as a headword plus some modifiers, where the headword represents the general query/type intent and the modifiers limit the scope of the intent. For example, in the case of *Japanese players in Major League Baseball*, we may use *players* as the headword, and the other terms/entities as modifiers. By interpreting queries and categories using headword-modifier patterns, other than the bag-of-words model, we are able to design a comprehensive category matching model that matches patterned queries against patterned categories, as a novel pattern-based document (category) retrieval model. However, there are two challenges to achieve such a headword-modifier pattern-based retrieval model.

First, given the decomposed headwords and modifiers of queries and categories, how to construct the match effectively and holistically from the matches of individual components (i.e., headword to headword, modifier to modifier, and headword to modifier)? We treat headwords as first-class citizens and effectively decompose the relevance model into four individual components which are further estimated using probabilistic generative models.

Second, when matching individual components, how do we address the vocabulary/conceptual gap of important concepts/entities? Word embedding techniques [18] are not promising. For example, *Novels by Alan Moore* will be highly relevant to *Novels by Frank Miller* under the embedding model. We exploit the structure of fine-grained Wikipedia categories to discover alternative representation of concepts/entities in the Wikipedia type taxonomy.

To the best of our knowledge, this is the first work to model both queries and categories as a headword-modifier model, and design comprehensive pattern-based document (category) retrieval models to quantify the relevance of a patterned category to a patterned query. We conduct extensive experiments on three widely-used entity search test sets: INEX-XER 2009 [13], SemSearch-LS [6] and TREC-Entity [3]. The experimental results show that our method achieves a significant improvement of the entity search performance over the state-of-the-art methods.

2 RELATED WORK

In recent years, much attention has been paid to the list entity search problem. The expert finding task (TREC Enterprise track [10]) is focused on retrieving a list of persons relevant to a query. The INEX Entity Ranking task [13] extends the expert finding task to include more entity types in Wikipedia. Topics in this task include both query terms and one or more target entity types [11]. The TREC 2009 Entity track [3] introduces the Related Entity Finding task, which is to identify homepages of target entities that satisfy a specified relation with a source entity and have a target type as constraints. These tasks use text corpus as the basis of entity search. Some recent tasks such as the Semantic Search challenge [6], the INEX Linked Data track [28] and the Question Answering over Linked Data challenge [19] apply structured knowledge bases (e.g., DBpedia) to support entity search. However, they are beyond the scope of this work.

Solutions of entity search are focused on context matching (evaluating the relevance to query terms) and category matching (evaluating the relevance to target entity types) [8]. The document model [1] has been shown to be an effective model of context matching [2, 7, 8]. Category matching has been verified to be critical for the performance of entity search [2, 8, 16]. To fully utilize the entity categories, people assume that some target entity types are explicitly provided in the query topics [2, 8, 13, 16]. Authors in [16] evaluate three methods of estimating the relevance between an entity category and a target entity type: 1) binary distance: a simple binary function based on whether two entity types are identical or not; 2) title distance, which evaluates the relevance based on pure text similarity of terms in both entity types; 3) content distance, which estimates the relevance based on the profiles of two entity types, that are generated by collecting the contexts of entities of each type. They show that binary distance works only when the target entity types are from the type taxonomy (i.e. they are well defined), and title distance is a better choice when the target entity types are specific categories.

The target entity types used in the queries directly affect the effectiveness of category matching. However, providing specific and accurate target entity types is an extra burden for users since they often do not have the knowledge of categories in the type taxonomy [2, 15, 16]. Consequently, authors in [2, 15, 16] propose to retrieve top relevant entity types of the query as the target entity types fed to the entity search methods. Strategies on ranking target entity types can be majorly classified as: 1) term-centric strategy [2]: a standard language model is proposed to estimate the relevance between an entity type and a query based on their terms. However, its performance is not good because estimating the relevance of short texts is very challenging due to the vocabulary gap problem. The work [23] estimates the different weights of individual query terms by using some NLP features, but it does not address the vocabulary gap problem; 2) type-centric strategy [4]: for each entity type, a pseudo-document is created by concatenating the descriptions of all entities that belong to the type. Then, a standard language model is used to estimate the relevance between the query and the pseudo-documents; 3) entity-centric strategy [16, 17]: it first finds relevant entities of the query. Then, the entities vote for their categories, which however often retrieves general entity types;

4) learning-to-rank strategy [15]: it considers not only type-centric and entity-centric strategies, but also other relevant signals such as taxonomy-driven features and type similarities. It shows that [15] the learning-to-rank strategy performs better than the other two strategies. However, it faces the problem of often retrieving general entity types that do not match to the query intent accurately.

The entities (with each having a Wiki document) in the Wikipedia collection are collectively labelled with many fine-grained Wikipedia categories. These fine-grained entity categories are organized as a type taxonomy of overlapping “trees”, with a very small number of top-level entity categories. Each category beyond the top-level may have multiple parent categories [16]. The Wikipedia entity categories are created and labelled by many different human editors. It therefore potentially has some data quality problems. However, to control the quality of category labelling, Wikipedia provides some guidelines when editors assign some categories to a Wiki document. For example, two general rules for category labelling are: 1) a labelled category should be as specific as possible; and 2) similar categories should be avoided. It also provides a peer reviewing mechanism to further control the quality of category labelling.

The work [29] shows that more than 90% entity-bearing web queries contain headword and some modifiers. Some studies [8, 24] of entity search have tried to use the headword and modifiers to model entity categories. They show that category matching benefits from a simple usage of term-based headword and modifiers. Although headwords and modifiers have been widely used for query understanding [29], they however are seldom used in modelling documents. The work [27] tries to extract headword-modifier pairs from queries and documents, and index them as terms. A language model is then applied to rank documents based on terms as well as headword-modifier pairs, in the setting of document retrieval. It however is still based on the bag-of-words model that does not address the vocabulary/conceptual gap challenge.

3 PROBLEM DEFINITION

For entity search, we formalize the relevance of a candidate entity e to a query q as a generative probability $p(e|q)$. According to the Bayes’ Theorem, such a probability can be rewritten as follows:

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

where $p(q|e)$ is the probability of generating the query q from an entity e , $p(e)$ is a priori probability of the entity e , and $p(q)$ is the probability of a query. We assume that $p(e)$ is uniformly distributed over all entities, and thus it does not affect the ranking. The probability $p(q)$ is consistent for a given query q . Consequently, we have $p(e|q) \propto p(q|e)$, and therefore $p(q|e)$ can be used to estimate the relevance of the entity e to the query q .

It has been shown [8] that context matching and category matching are two important components of entity search. We therefore decompose $p(q|e)$ into two components, $p_c(q|e)$ and $p_t(q|e)$, which model the relevance of context matching and that of category matching respectively. They are unified by a weighted aggregation of the two relevance scores:

$$p(q|e) = p_c(q|e)^{1-\alpha_0} \cdot p_t(q|e)^{\alpha_0} \quad (2)$$

where the parameter α_0 is applied to tune the weight between the context matching component and the category matching component. When $\alpha_0 = 0$, only context matching model is applied for ranking entities. Correspondingly, if $\alpha_0 = 1$, only category matching model is applied. In this paper, our focus is on devising the category matching model. The context matching model of [8] is directly applied in our study.

The relevance of category matching is estimated as:

$$p_t(q|e) = \max_{t \in T(e)} p(q|t)p(t|e) \quad (3)$$

where $T(e) = \{t|e \in t\}$, is the set of types that e belongs to, and $p(t|e)$ is the probability of generating the type t from an entity e . The component $p(q|t)$ is the probability of generating the query q from the type/category t . It is to capture the relevance of a type to a query. Since entities are labelled using the fine-grained Wikipedia categories with a very high confidence, we therefore set $p(t|e) = 1$, for $t \in T_e$. Other studies [8, 16] also apply the max function for category matching, by using the highest relevance score of a type $t \in T(e)$ as the category matching relevance of the entity e . The estimation of $p_t(q|e)$ is then to find a category t of e that has the maximal relevance score $p(q|t)$ with regard to the query q . We therefore define the basic category matching problem as:

Problem Definition: Given a natural language question q , and a fine-grained entity category $t \in T$ where T is the type taxonomy, estimate the relevance $p(q|t)$.

Such a way category matching supports ranking entity types in a unified model, without generating target entity types as the intermediate results. In the whole solution of entity search, we do not need to compute the category relevance of every type, but only those categories of the candidate entities derived from the context matching model.

4 CATEGORY MATCHING MODEL

In our method, both queries and entity categories are modeled using headwords and modifiers, so that they are treated as basic elements for effective category matching. How to effectively extract entities and concepts, and how to detect headwords from short texts have been studied by many query understanding techniques (e.g., [26, 29]). It is therefore not the focus of this paper. We simply apply an open source tool [25] to recognize entities/concepts from queries and entity types. For detecting headwords, we apply the following natural language patterns:

- Pattern 1: If a query/type is a sequence of nouns and adjectives (e.g., *Italian Nobel prize winners*), the last noun is extracted as the headword (underlined).
- Pattern 2: If a query/type contains some prepositions for connecting two components (e.g. “A for B”, “A of B”, “A with B”), it is almost true that A includes the headword [29]. For example, *Noble english person from the Hundred Years’ War*. Then, we process the component A by using the Pattern 1.
- Pattern 3: If a query/type is an attributive clause (which often includes “WH” terms such as “where”, “which”), the modifier

of the “WH” term is used as the headword. For example, *Musicians who appeared in the Blues Brothers movies*.

The modifiers are composed of all the other non-stop words in the query/type, as well as those entities/concepts extracted from the query/type (except the detected headword).

In our category matching method, we treat headwords and modifiers individually and emphasize more on the headword part. This is because headwords often serve as the root (general type) of entity categories. An effective matching of headwords between queries and entity categories is therefore a prerequisite of guaranteeing the performance of category matching. On the other hand, the effective matching of modifiers between queries and entity categories is also important because it help us to exploit the semantic constraints implied by the modifiers of the fine-grained entity categories. For a query q , let h_q be its headword, and $M_q = \{m_1, \dots, m_k\}$ be the set of its modifiers. Let h_t and M_t be the headword and the modifiers of the entity type t . By assuming the independence between the headword and the modifiers, we estimate $p(q|t)$ as follows:

$$\begin{aligned} p(q|t) &= p(h_q, M_q|t) = p(h_q|t)p(M_q|t) \\ &= \frac{p(t|h_q)p(h_q)}{p(t)} \frac{p(t|M_q)p(M_q)}{p(t)} \\ &\propto p(t|h_q)p(t|M_q) \\ &= p(h_t, M_t|h_q)p(h_t, M_t|M_q) \\ &= p(h_t|h_q)p(M_t|h_q)p(h_t|M_q)p(M_t|M_q) \end{aligned} \quad (4)$$

where $p(h_q)$ is the probability the headword h_q , $p(M_q)$ is the probability the modifiers M_q , and $p(t)$ is the probability of entity category t which is assumed to be uniformly distributed over all entity types. The probability $p(q|t)$ can then be estimated as a pattern-based matching of 4 components $p(h_t|h_q)$, $p(M_t|h_q)$, $p(h_t|M_q)$, and $p(M_t|M_q)$. The probability $p(h_t|h_q)$ is called headword relevance, and $p(M_t|M_q)$ is called modifier relevance. The components $p(M_t|h_q)$ and $p(h_t|M_q)$ are called headword-modifier relevance.

To avoid the impacts of different scales when estimating these components, we assigned some weights to them so that the probability $p(q|t)$ is actually estimated as:

$$p(q|t) = p(h_t|h_q)^{\alpha_1} p(M_t|h_q)^{\alpha_2} p(h_t|M_q)^{\alpha_3} p(M_t|M_q)^{\alpha_4} \quad (5)$$

These parameters satisfy $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1.0$, and their values can be learned from some validation set of particular test cases. We then introduce probabilistic generative models to estimate the relevance of these components respectively.

4.1 Headword Relevance

The component $p(h_t|h_q)$ is to capture the relevance of two headwords h_t and h_q , to handle the cases of conceptual gap when h_t and h_q are different. For example, for a query *Works by Charles Rennie Mackintosh*. The concrete query intent are actually *buildings, structures*, which are often used as the headwords of the relevant entity categories in the Wikipedia type taxonomy. If we simply retrieve entity categories having the same headword (i.e., *Works*) as the query, we will definitely obtain a low recall of category matching. To effectively estimate $p(h_t|h_q)$, we build a headword hypernymy graph by using the type taxonomy of fine-grained Wikipedia entity categories. In this directed graph, vertexes are those headwords extracted from the Wikipedia entity types/categories. One edge

$h_t \rightarrow h_{t'}$ will be created if a headword h_t is a direct hypernym of another headword $h_{t'}$.

To build the headword hypernymy graph, we statistically identify the headword-headword hypernymy pairs based on the hypernymy of Wikipedia categories. Given two entity categories t and t' (e.g., *American people by occupation* and *American writers*), that satisfy t is the parent category of t' , we create an edge $h_t \rightarrow h_{t'}$ (e.g., *people* $\rightarrow$ *writers*) and count the number of category pairs having such a headword hypernymy relationship as the weight of the edge. By this way, a preliminary headword hypernymy graph can be generated. We then filter those possible noisy edges whose weight is less than 3, to partially avoid the impacts of those noisy pairs of hypernymy categories. Those remaining edges then form the headword hypernymy graph for estimating the headword relevance. Note that such a way of building the headword hypernymy graph may occasionally generate loops. For example, both *works* $\rightarrow$ *books* and *books* $\rightarrow$ *works* exist in the headword hypernymy graph. They however do not affect the estimation of the headword relevance. With the headword hypernymy graph, we are able to estimate the headword relevance $p(h_t|h_q)$ as:

$$p(h_t|h_q) = \begin{cases} 1 & \text{if } h_t \text{ is a child or grandchild of } h_q \\ \frac{1}{|H(h_t)|} & \text{else if } h_t \text{ is a parent of } h_q \\ \frac{1}{|S(h_t)|} & \text{else if } h_t \text{ is a grandparent of } h_q \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $H(h_t)$ is a set of headwords which are children of h_t and $S(h_t)$ is a set of headwords which are grandchildren of h_t . Obviously, those hyponym headwords (which often appear in fine-grained entity categories) of h_q will have larger relevance than those hypernym headword (general headwords, which may deviate from the query intent). According to the above equation, relevant headwords of h_q are limited to those headwords whose graph distance to h_q is no more than two. This is designed to reduce the impacts of noisy pairs of hypernymy headwords.

4.2 Modifier Relevance

To estimate the modifier relevance $p(M_t|M_q)$, we propose to apply a language model which utilizes the type taxonomy T . The modifier relevance is estimated as:

$$\begin{aligned} p(M_t|M_q) &= \frac{p(M_q|M_t)p(M_t)}{p(M_q)} \\ &\propto p(M_q|M_t) \\ &= \prod_{m \in M_q} p(m|M_t)^{n(m, M_q)} \end{aligned} \quad (7)$$

where $p(M_t)$ is the probability of the modifiers M_t , and $n(m, M_q)$ measures the times of the modifier m presenting in the modifiers M_q . The component $p(M_q|M_t)$ is the probability of generating the modifiers M_q from the modifiers M_t . By assuming that the probability $p(M_t)$ is uniformly distributed over all entity types, we are able to transform the estimation of $p(M_t|M_q)$ as $p(M_q|M_t)$, so that a standard language model can be applied to estimate the probability $p(M_q|M_t)$. We therefore need to estimate the component $p(m|M_t)$, which is the probability of generating the modifier m from M_t , as the basis of estimating the modifier relevance.

To estimate the probability $p(m|M_t)$ using a language model, we need to address the cases when m does not present in the M_t by using some smoothing factors. The component $p(m|M_t)$ is then estimated as:

$$p(m|M_t) = (1 - \lambda_1 - \lambda_2) \frac{n(m, M_t)}{n(M_t)} + \lambda_1 p(m|T) + \lambda_2 p(m|D) \quad (8)$$

where $n(M_t)$ is the number of modifiers in M_t , $p(m|T)$ is a background probability of generating the modifier m from the type taxonomy T , and $p(m|D)$ is a background probability of generating the term/entity of m from the corpus D . The parameters λ_1 and λ_2 are used as the weights of the two smoothing factors. As a background probability, the component $p(m|T)$ can then be estimated as $\frac{\sum_t n(m, M_t)}{\sum_t n(M_t)}$. Considering that the modifier m may also not appear in the type taxonomy T , we therefore apply the second smoothing factor $p(m|D)$.

To address the vocabulary gap of those entities/concepts between the query q and the entity category t , we extend the entities/concepts in M_q to generate more expanded modifiers as part of the query. The expansion is achieved in the following way. For each entity/concept e (which is also a modifier) of M_q , we extract the entity categories of e as $T(e)$. Then, an expanded modifier set M_e will be generated by merging the modifiers M_t of each entity type $t \in T(e)$. With the expanded modifier set M_e of each entity/concept in M_q , the estimation of the modifier relevance $p(M_t|M_q)$ in Equation 7 can then be revised as:

$$\begin{aligned} p(M_t|M_q) &\propto p(M_q|M_t) \prod_{e \in M_q} p(M_e|M_t) \\ &= p(M_q|M_t) \prod_{e \in M_q} \sum_{m \in M_e} p(m|M_t) \end{aligned} \quad (9)$$

4.3 Headword-Modifier Relevance

To estimate the headword-modifier relevance $p(h_t|M_q)$ of the Equation 5, we need to utilize the context of the modifiers in the document corpus. The reason of not applying the type taxonomy to estimate $p(h_t|M_q)$ is that, due to ad-hoc query terms may be used in the query, modifiers of M_q may not present in any entity type of T . This will lead to the failure of find enough evidence to evaluate $p(h_t|M_q)$ effectively. The headword-modifier relevance $p(h_t|M_q)$ is therefore estimated using a language model:

$$\begin{aligned} p(h_t|M_q) &= \frac{p(M_q|h_t)p(h_t)}{p(M_q)} \\ &\propto p(M_q|h_t) = \prod_{m \in M_q} p(m|h_t)^{n(m, M_q)} \\ &= \prod_{m \in M_q} \left(\frac{p(h_t|m)p(m)}{p(h_t)} \right)^{n(m, M_q)} \\ &\propto \prod_{m \in M_q} p(h_t|m)^{n(m, M_q)} \end{aligned} \quad (10)$$

where $p(M_q|h_t)$ is the probability of generating the query modifiers M_q from the headword h_t . The component $p(m|h_t)$ is the generative probability of the modifier m from the headword h_t , and the component $p(h_t|m)$ is the generative probability of the headword h_t from the modifier m . The component $p(m)$ is the probability of a modifier m , which is assumed to be uniformly distributed over all

modifiers. The component $p(h_t)$ is also assumed to be uniformly distributed over all headwords. According to the Bayes' Theorem, as shown in the above equation, we are therefore able to model the relevance $p(h_t|M_q)$ using a language model. The estimation of $p(h_t|m)$ is then the only component to be resolved.

We propose to estimate $p(h_t|m)$ from the text contexts of m and h_t . It is then estimated as:

$$\begin{aligned} p(h_t|m) &= p(h_t|ctx(m)) \\ &= (1 - \lambda_3) \frac{n(h_t, ctx(m))}{\sum_w n(w, ctx(m))} + \lambda_3 p(h_t|D) \end{aligned} \quad (11)$$

where $ctx(m)$ is the contexts of the modifier m , which is composed of the surrounding texts of m . The component $n(w, ctx(m))$ measures the frequency of the word w appearing in the context $ctx(m)$. Correspondingly, $n(h_t, ctx(m))$ measures the frequency of entities which have a category with the headword h_t , presenting in the context $ctx(m)$. The component $p(h_t|D)$, as a smoothing factor with a weight λ_3 , represents the probability of generating h_t from the document corpus D , which is also evaluated based on the frequency of entities having a category of the headword h_t over all possible words that can be extracted from D . Equation 11 is therefore an application of the language model for estimating the generative probability of h_t in the contexts of modifier m . To estimate the probability $p(h_t|m)$, we need build a context profile for each modifier in M_q . We first retrieve top relevant documents of q as the document set for building the context profile of its modifiers. The context profile $ctx(m)$ is then the aggregation of contexts (sentences) where m appears in the selected document set.

Note that the headword relevance $p(h_t|h_q)$ (in Equation 6) allows us to find some relevant headwords of the query headword h_q . However, it is not enough to constrain the query intent without the support of the headword-modifier relevance $p(h_t|M_q)$. For example, for the query *Works by Charles Rennie Mackintosh*, candidate headwords such as *Films*, *Novels*, *Stories*, *Books*, *Buildings*, *Structures*, *Architecture* will be the relevant headwords of h_q , in terms of $p(h_t|h_q)$. The headword-modifier relevance $p(h_t|M_q)$ somehow makes up the shortcoming of the headword relevance $p(h_t|h_q)$ by also considering the relevance of headwords in the contexts of M_q .

The estimation of the other headword-modifier relevance $p(M_t|h_q)$ in the Equation 5 is quite similar to that of $p(h_t|M_q)$.

$$\begin{aligned} p(M_t|h_q) &= \prod_{m \in M_t} p(m|h_q)^{\frac{n(m, M_t)}{n(M_t)}} \\ &= \prod_{m \in M_t} \left(\frac{p(h_q|m)p(m)}{p(h_q)} \right)^{\frac{n(m, M_t)}{n(M_t)}} \\ &\propto \prod_{m \in M_t} p(h_q|m)^{\frac{n(m, M_t)}{n(M_t)}} \end{aligned} \quad (12)$$

where the component $n(M_t)$ is designed to normalize the headword-modifier relevance $p(M_t|h_q)$ so that it is not affected by the number of modifiers in M_t . The component $p(h_q|m)$ is estimated using the same way of Equation 11, by switching h_t as h_q . The major difference of computing $p(M_t|h_q)$ from computing $p(h_t|M_q)$ is that, the documents used for building the context profiles of modifiers come from the Wikipedia documents of the entities having a type

of t , instead of the top relevant results using M_q as query terms when computing $p(h_t|M_q)$.

5 EXPERIMENTAL SETUP

5.1 Data Sets

We evaluate the performance of our method and that of the baselines in the settings of three entity search test sets:

- INEX-XER 2009 (INEX): The INEX 2009 Entity Ranking track [13] launched an entity search test set with 55 topics whose answers are Wikipedia entities. We ignore those target entity types explicitly given in INEX for each topic. An example query is *Alan Moore graphic novels adapted to film*.
- SemSearch-LS (LS): This test set is used in the list search task at the 2011 Semantic Search challenge [6]. Same as [5], we use 43 out of 50 topics which have answers from Wikipedia. An example is *Apollo astronauts who walked on the Moon*.
- TREC-Entity (TREC): It is built for the related entity finding task at the TREC 2009 Entity track [3]. Same as [5], we use 17 out of 20 topics which have answers from Wikipedia. An example query is *Members of the band Jefferson Airplane*.

According to these test sets, a document collection of Wikipedia Corpus of 2008 edition is used in our experiments. It contains 2.67 millions entities with each of them having a Wikipedia document, 371,797 fine-grained entity categories, and 8,074,151 entity-to-category pairs. To extract entities from pure texts, we utilize an open source toolkit Wikipedia-Miner [22], which takes the unstructured texts as inputs and detects mentions of the Wikipedia entities in the inputs using machine learning techniques.

Although experiments of these test sets report the overall performance of entity search, the performance of category matching models can however be evaluated by substituting the category matching model of an entity search method with other alternatives.

5.2 Evaluation Metrics

We adopt the following metrics to evaluate the performance of the tested methods:

- $p@k$: the percentage of relevant entities in the top- k results.
- R -pre: the percentage of relevant entities in the top- R results, where R is the number of given correct results for a topic.
- MRR: the reciprocal rank of a topic is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank (MRR) is the average of the reciprocal ranks of all topics.
- MAP: the Mean Average Precision over all topics in a test set. It is estimated as follows [20]:

$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{|M_j|} \sum_{i=1}^{|M_j|} p@R_k \quad (13)$$

where Q is the topic set, M_j is the answer set of the query j , and R_k is the rank of the k th answer in the M_j .

- xinfAP: an extended inferred AP incorporating non-random relevance judgments by employing stratified random sampling [30]. In the evaluation of the INEX Entity Ranking track, it replaces the MAP as an official evaluation metric.

5.3 Baselines

Our method is compared with three baselines of entity search:

BBR [2]: It applies a term-centric category matching method. Both entities and queries are modeled using a term-based representation and a category-based representation. KL divergence is then applied to compute the probability of generating the term-based/category-based representation of a query, given the term-based/category-based representation of an entity. To build the category-based representation of a query, it applies a standard language model to estimate the relevance of entity categories to the query. The top-10 relevant entity categories are used to construct the category-based representation of the query.

KK [16]: A language model with Jelinek-mercer smoothing is applied to estimate the relevance between query terms and entity contexts for context matching. For category matching, it shows that, on the INEX test set, the title distance gives the best results on estimating the relevance between an entity category and a target entity type. It applies an entity-centric strategy to obtain target entity types. The top 50 relevant entities from the context matching model are used to vote for relevant categories. The derived top 2 relevant categories are taken as target entity types.

CGS [8]: It incorporates context matching, category matching, and result ranking for achieving a good performance of entity search. It however assumes that the target entity types are explicitly given. To compare with our method, we apply the learning-to-rank method proposed in [15] to find some relevant entity categories for each query. Still, the top 2 relevant categories are taken as the target entity types in our experiments.

PBM (stands for pattern-based matching): This is our proposed method that also applies the same context matching model of CGS.

5.4 Parameter Setting

To estimate the parameters of our models as well as those of the compared solutions, we apply the Coordinate Ascent (CA) algorithm under the sum normalization and non-negativity constraints [21] and directly optimize the MAP/xinfAP metric. The CA algorithm is a commonly used optimization technique, which iteratively optimizes a single parameter while holding all other parameters fixed. We use 5-fold cross validation for parameter tuning in each test set. For our method, we do not tune the parameters λ_1 , λ_2 and λ_3 , by setting them as $\lambda_1 = \lambda_2 = \frac{1}{3}$, and $\lambda_3 = 0.5$. When estimating $p(h_t|M_q)$, we use top-20 relevant documents to build the profiles of modifiers. We focus on the setting of parameters α_0 , α_1 , α_2 , α_3 and α_4 . As $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1.0$ is required, we initialize them as $[0.5, 1, 0, 0, 0]$ to run the CA algorithm with 3 random restarts.

To measure the statistical significance, a two-tailed paired t-test is applied. We use the * to denote the difference at the 0.01 level and † to denote the difference at the 0.05 level.

6 RESULTS

6.1 Research Questions

We address the following research questions:

- **RQ1:** Can entity search performance be improved by exploiting the fine-grained Wikipedia entity categories using the proposed models of this work? (§6.2)

Table 1: Performance comparison on the mixed test set. Significance is tested against the BBR baseline.

Models	INEX+LS+TREC Topics					
	p@5	p@10	p@20	MRR	R-pre	MAP
<i>BBR</i>	0.256	0.243	0.201	0.350	0.206	0.202
<i>KK</i>	0.231	0.203	0.164	0.345	0.186	0.178
<i>CGS</i>	0.179	0.147	0.131	0.290	0.117	0.110
<i>PBM_H</i>	0.332 [†]	0.301 [†]	0.248 [†]	0.441 [†]	0.263 [†]	0.232
<i>PBM_M</i>	0.209	0.213	0.183	0.296	0.178	0.164
<i>PBM_{H+M}</i>	0.356*	0.333*	0.272*	0.490*	0.289*	0.254*
<i>PBM</i>	0.374*	0.356*	0.284*	0.534*	0.298*	0.272*

- **RQ2:** Can the proposed pattern-based category matching method improve the performance of the baselines? (§6.3)
- **RQ3:** How do the proposed method as well as the baselines perform on different test sets? (§6.4)
- **RQ4:** How robust is our method with respect to different parameter settings? (§6.5)
- **RQ5:** What are the impacts of entity search performance if the headword or entities cannot be accurately recognized from the query? (§6.6)

6.2 Overall Performance

We derive a mixed test set by combining all the topics of the three test set, so the it finally contains 115 topics. Our *PBM* method is compared against the three baselines using the mixed test set. In addition, we also test three alternatives of our method: *PBM_H*, *PBM_M*, and *PBM_{H+M}*. The *PBM_H* method only applies headword relevance (i.e., $\alpha_1 = 1$); the *PBM_M* method only applies modifier relevance (i.e., $\alpha_4 = 1$); the *PBM_{H+M}* method applies both headword relevance and modifier relevance, but not the headword-modifier relevance (i.e., $\alpha_1 + \alpha_4 = 1$). The results are shown in Table 1.

The results of Table 1 show that *PBM* performs much better than the baselines, in terms of all metrics used. By comparing *PBM* with its alternatives, we may find that the headword relevance (*PBM_H*) plays a more significant role than the modifier relevance (*PBM_M*). However, by combining the individual components in the *PBM*, we are able to largely improve the search performance. This addresses the RQ1 that the proposed category matching method *PBM* does exploit more information from the fine-grained Wikipedia categories, to achieve a higher search performance than the baselines.

6.3 Improvements of Baselines

In this experiment, we address RQ2 by applying our category matching method to the baselines. There are two strategies to improve the baselines using *PBM*. Strategy 1 (labelled with “+”): only apply the *PBM* to find target entity types (top-2 relevant types in our experiments) for the baselines. Strategy 2 (labelled with “++”): use our category matching method (Equation 3) to substitute the category matching methods of baselines. The substitution is easy for baselines because they all model context matching and category matching individually as two weighted probabilities. We also test the performance of the pure context matching method (labelled with “-”) of each baseline, so that the performance improvement

Table 2: Performance comparison when applying PBM

Models	INEX+LS+TREC Topics					
	p@5	p@10	p@20	MRR	R-pre	MAP
<i>BBR-</i>	0.184	0.177	0.156	0.312	0.174	0.151
<i>BBR</i>	0.256	0.243	0.201	0.350	0.206	0.202
<i>BBR+</i>	0.290	0.267	0.206	0.418 [†]	0.231	0.218
<i>BBR++</i>	0.367*	0.308*	0.251*	0.506*	0.275*	0.253*
<i>KK-</i>	0.186	0.179	0.155	0.322	0.149	0.152
<i>KK</i>	0.231	0.203	0.164	0.345	0.186	0.178
<i>KK+</i>	0.292 [†]	0.270*	0.216*	0.435*	0.223*	0.215*
<i>KK++</i>	0.329*	0.296*	0.234*	0.475*	0.255*	0.234*
<i>CGS-</i>	0.111	0.110	0.107	0.243	0.107	0.099
<i>CGS</i>	0.179	0.147	0.131	0.290	0.117	0.110
<i>CGS+</i>	0.317*	0.290*	0.231*	0.427*	0.232*	0.217*
<i>CGS++</i>	0.374*	0.356*	0.284*	0.534*	0.298*	0.272*

of category matching methods can be easily identified. The results are shown in Table 2. Significances are tested against the baselines *BBR*, *KK* and *CGS* respectively.

As the results show in Table 2, Strategy 1 (“+”) does help to improve the search performance of each corresponding baseline (without notations). Strategy 2 (“++”) further improves the performance of Strategy 1. All the baselines benefit from the usage of our category matching method. Note that *CGS++* is exactly our proposed method *PBM* that applies the context matching of *CGS* as its context matching component. We may notice that the improvement of *CGS* is much larger than that of *BBR* and *KK*. This is mainly because the context matching model *CGS-* performs worse than the other two context matching models (*BBR-* and *KK-*). The *CGS* applies a document model for context matching, and the other two baselines build their context matching model more like a candidate model [1]. It has been shown that [8] although pure document model for context matching performs worse than the candidate model, it however outperforms the others when combining with category matching model. This is because candidate model benefits from the implicit category information embedded in the contexts of entities. When comparing *CGS+* with *CGS*, we find that *CGS+* significantly outperforms *CGS*. Their only difference is the methods used for finding target entity types. Method *CGS+* applies *PBM* to help to find many fine-grained target entity types. Method *CGS* however applies the learning-to-rank method [15], which is assumed to be the state-of-art method for finding target entity types. It however usually finds general entity types, which are not that beneficial to the overall entity search performance.

6.4 Breakdown by Query Subsets

We then compare *PBM* with baselines on three individual test sets to address RQ3. On each test set, we perform a 5-fold cross validation for parameter tuning. The results for three test sets are shown in Table 3-5 respectively. Significance is all tested against *BBR*. According to the results, we find that *PBM* consistently outperforms the other baselines on the three test sets. Comparatively, the advantage of *PBM* is more prominent on the INEX test set. This is because the INEX test set is created directly based on the Wikipedia

Figure 1: Topic-level difference in xinfAP for INEX, and in MAP for LS and TREC.

Table 3: Performance comparison on INEX-XER 2009

Models	INEX-XER 2009 Topics					
	p@5	p@10	p@20	MRR	R-pre	xinfAP
BBR	0.338	0.324	0.269	0.413	0.255	0.237
KK	0.269	0.271	0.241	0.372	0.223	0.197
CGS	0.255	0.204	0.179	0.392	0.171	0.138
PBM	0.506*	0.495*	0.416*	0.639*	0.386*	0.356*

Table 4: Performance comparison on SemSearch-LS

Models	SemSearch-LS Topics					
	p@5	p@10	p@20	MRR	R-pre	MAP
BBR	0.205	0.170	0.147	0.344	0.176	0.184
KK	0.195	0.174	0.144	0.374	0.183	0.182
CGS	0.140	0.123	0.100	0.258	0.085	0.088
PBM	0.326[†]	0.290[†]	0.216	0.475[†]	0.254	0.237

Table 5: Performance comparison on TREC-Entity

Models	TREC-Entity Topics					
	p@5	p@10	p@20	MRR	R-pre	MAP
BBR	0.129	0.176	0.121	0.167	0.121	0.133
KK	0.118	0.112	0.091	0.259	0.134	0.141
CGS	0.059	0.047	0.068	0.097	0.030	0.068
PBM	0.201	0.213	0.175	0.279	0.185	0.196

corpus. The other two test sets are not built on top of Wikipedia, and many of their result entities are short of relevant Wikipedia categories.

Figure 1 illustrates the per-topic difference by comparing *PBM* against two baselines *BBR* and *KK*. It can be found that *PBM* performs better than the other two baselines on a large number of topics than vice versa, with the most significant difference achieved by the INEX test set. We select some examples from which the advantage of *PBM* can be observed. For topic *operating systems to which Steve Jobs related*, the most relevant entity type retrieved by *PBM* is *Apple inc. operating systems* where the expanded modifier *Apple inc.* is discovered from the categories of the entity *steve jobs*. A good example of headword relevance is the match of a query *astronauts who landed on the moon* with a type *people who have walked on the moon*. For topic *Nordic authors who are known for children's literature*, three most relevant entity types are *Swedish children's writers*, *Danish children's writers* and *Norwegian children's writers*, which all benefit from the way of estimating headword relevance and modifier relevance. However, *PBM* sometimes performs worse when the identified headword is a tailed headword or it does not appear in the type taxonomy, e.g., *axis powers of world war ii*.

6.5 Impacts of Parameter Settings

To study the impacts of parameters, we first fix α_{1-4} as [0.3, 0.1, 0.4, 0.2] and adjust α_0 , the impacts of α_0 on different test sets are then plotted in Figure 2(a). It shows that the best performance on various test sets is achieved when α_0 is set between 0.6-0.8, which indicates that category matching is more important than context matching. We then fix α_0 as 0.7, and optimize α_{1-4} for individual test sets using the CA algorithm [21]. The derived parameter settings for different test sets are shown in Figure 2(b). We can find that, in

Figure 2: Impacts of parameters

general, α_1 and α_3 are assigned more weights than α_2 and α_4 . This is reasonable because they are used to estimate the relevance of the headwords, i.e., α_1 for $p(h_t|h_q)$, and α_3 for $p(h_t|M_q)$. We finally fix the parameters α_{0-4} in the mixed test set, and adjust the parameters λ_{1-3} . When adjusting the parameter λ_1 (or λ_2), we set the parameter λ_2 (or λ_1) as 0.1, so that it can be tuned in a large range. The results in Figure 2(c) show that the impacts of λ_{1-3} are not obvious. We therefore apply their default values in all the other experiments.

6.6 Impacts of Headwords and Entities

The performance of *PBM* will be affected if headwords and entities cannot be recognized from the queries which are more ad-hoc than entity categories. We conduct two experiments by assuming that headwords and entities are not extracted from the queries respectively. Table 6 shows the results. It can be observed that when headword detection fails, the MAP of *PBM* drops from 0.272 to 0.234. It is still better than that of the baselines in Table 1 (e.g., 0.202 for *BBR*). In this case, the headword relevance and a headword-modifier relevance ($p(M_t|h_q)$) cannot be estimated. However, we are still able to estimate category matching relevance (in Equation 5) using the other two components. This makes our method robust in terms of headword recognition. The results also show that when entity recognition fails, the impact is less than that of the headword failure. This is reasonable because it only affects the query expansion of the modifier relevance in Equation 9.

We also test two alternatives of *PBM* by substituting its headword relevance and modifier relevance using the similarities of headword embeddings and modifier (terms/entities) embeddings respectively. The relevance model of [18] is applied for evaluating the relevance of modifier embeddings. The results in Table 6 show that their performance is not as good as the proposed *PBM* method that exploits the type taxonomy for addressing the conceptual/vocabulary gap.

7 CONCLUSION

We show that entity search, especially the category matching part of entity search, benefits a lot from the usage of pattern-based query-to-category matching techniques by modeling both queries and entity categories as a headword plus some modifiers. Such a headword-and-modifier model allows us to exploit the benefits of

Table 6: Impacts of headwords and entities

Models	INEX+LS+TREC Topics					
	p@5	p@10	p@20	MRR	R-pre	MAP
no headword	0.313	0.292	0.237	0.445	0.250	0.234
no entities	0.336	0.315	0.256	0.454	0.269	0.242
<i>PBM</i>	0.374	0.356	0.284	0.534	0.298	0.272
headword embedding	0.327	0.317	0.260	0.450	0.271	0.245
modifier embedding	0.353	0.328	0.267	0.495	0.280	0.255

fine-grained Wikipedia categories, and to design a unified model of category matching where the pattern matching of headwords and modifiers is holistically constructed and further decomposed into headword relevance, modifier relevance, as well as headword-modifier relevance. By exploiting Wikipedia type taxonomy for addressing the vocabulary gap of concepts/entities in the query, we are able to effectively utilize the rich semantics in fine-grained Wikipedia categories.

8 ACKNOWLEDGMENTS

Yueguo Chen is supported by the National Science Foundation of China under grants No. 61472426, U1711261, 61432006, and the State Visiting Scholar Funds from the China Scholarship Council under Grant Number 201706365018. Denghao Ma is supported by the Outstanding Innovative Talents Cultivation Funded Programs 2017 of Renmin University of China and the State Scholarship Fund from China Scholarship Council under Grant Number 201706360309. Kevin Chang is supported by a gift from Huawei.

REFERENCES

- [1] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. 2006. Formal models for expert finding in enterprise corpora. In *SIGIR 2006: Proceedings of the 29th ACM SIGIR*. ACM, 43–50.
- [2] Krisztian Balog, Marc Bron, and Maarten de Rijke. 2011. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.* 29, 4 (2011), 22.
- [3] Krisztian Balog, Arjen P. de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. 2009. Overview of the TREC 2009 Entity Track. In *Proceedings of The 18th TREC*.
- [4] Krisztian Balog and Robert Neumayer. 2012. Hierarchical target type identification for entity-oriented queries. In *21st ACM CIKM*. ACM, 2391–2394.

- [5] Krisztian Balog and Robert Neumayer. 2013. A test collection for entity search in DBpedia. In *The 36th ACM SIGIR*. ACM, 737–740.
- [6] Roi Blanco, Harry Halpin, Daniel M Herzig, Peter Mika, Jeffrey Pound, David R Cheriton, and Henry Thompson. 2017. Entity Search Evaluation over Structured Web Data. In *Proceedings of the 1st International Workshop on EOS*. 65–71.
- [7] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2013. Example Based Entity Search in the Web of Data. In *Advances in Information Retrieval - 35th ECIR*. 392–403.
- [8] Yueguo Chen, Lexi Gao, Shuming Shi, Xiaoyong Du, and Ji-Rong Wen. 2014. Improving Context and Category Matching for Entity Search. In *Proceedings of the 28th AAAI*. 16–22.
- [9] Marek Ciglan, Kjetil Nørvåg, and Ladislav Hluchý. 2012. The SemSets model for ad-hoc semantic list search. In *Proceedings of the 21st WWW*. 131–140.
- [10] Nick Craswell, Arjen P. de Vries, and Ian Soboroff. 2005. Overview of the TREC 2005 Enterprise Track. In *Proceedings of the 14th TREC*.
- [11] Arjen P. de Vries, Anne-Marie Vercoustre, James A. Thom, Nick Craswell, and Mounia Lalmas. 2007. Overview of the INEX 2007 Entity Ranking Track. In *Focused Access to XML Documents, 6th Workshop of INEX*. 245–251.
- [12] Gianluca Demartini, Arjen P. de Vries, Tereza Iofciu, and Jianhan Zhu. 2008. Overview of the INEX 2008 Entity Ranking Track. In *Advances in Focused Retrieval, 7th Workshop of the INEX*. 243–252.
- [13] Gianluca Demartini, Tereza Iofciu, and Arjen P. de Vries. 2009. Overview of the INEX 2009 Entity Ranking Track. In *Focused Retrieval and Evaluation, 8th Workshop of INEX*. 254–264.
- [14] Dario Garigliotti and Krisztian Balog. 2017. On Type-Aware Entity Retrieval. In *Proceedings of the ACM SIGIR ICTIR*. ACM, 27–34.
- [15] Dario Garigliotti, Faegheh Hasibi, and Krisztian Balog. 2017. Target Type Identification for Entity-Bearing Queries. In *Proceedings of the 40th ACM SIGIR*. ACM, 845–848.
- [16] Rianne Kaptein and Jaap Kamps. 2013. Exploiting the category structure of Wikipedia for entity ranking. *Artif. Intell.* 194 (2013), 111–129.
- [17] Rianne Kaptein, Pavel Serdyukov, Arjen P. de Vries, and Jaap Kamps. 2010. Entity ranking using Wikipedia as a pivot. In *Proceedings of the 19th ACM CIKM*. ACM, 69–78.
- [18] Tom Kenter and Maarten de Rijke. 2015. Short Text Similarity with Word Embeddings. In *Proceedings of the 24th ACM CIKM*. ACM, 1411–1420.
- [19] Vanessa López, Christina Unger, Philipp Cimiano, and Enrico Motta. 2013. Evaluating question answering over linked data. *J. Web Sem.* 21 (2013), 3–13.
- [20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.
- [21] Donald Metzler and W. Bruce Croft. 2007. Linear feature-based models for information retrieval. *Inf. Retr.* 10, 3 (2007), 257–274.
- [22] David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM CIKM*. ACM, 509–518.
- [23] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. 2016. Parameterized Fielded Term Dependence Models for Ad-hoc Entity Retrieval from Knowledge Graph. In *Proceedings of the 39th SIGIR*. 435–444.
- [24] Madhu Ramanathan, Srikant Rajagopal, S. Venkatesh Karthik, Meenakshi Sundaram Murugesan, and Saswati Mukherjee. 2009. A Recursive Approach to Entity Ranking and List Completion Using Entity Determining Terms, Qualifiers and Prominent n-Grams. In *Focused Retrieval and Evaluation, 8th Workshop of the INEX*. 292–302.
- [25] Lev-Arie Ratnov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the 13th CoNLL*. 147–155.
- [26] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the 2011 EMNLP*. 1524–1534.
- [27] Young-In Song, Kyoung-Soo Han, Sang-Bum Kim, So-Young Park, and Hae-Chang Rim. 2008. A novel retrieval approach reflecting variability of syntactic phrase representation. *J. Intell. Inf. Syst.* 31, 3 (2008), 265–286.
- [28] Qiuyue Wang, Jaap Kamps, Georgina Ramirez Camps, Maarten Marx, Anne Schuth, Martin Theobald, Sairam Gurajada, and Arunav Mishra. 2012. Overview of the INEX 2012 Linked Data Track. In *CLEF 2012 Evaluation Labs and Workshop, Online Working Notes*.
- [29] Zhongyuan Wang, Haixun Wang, and Zhirui Hu. 2014. Head, modifier, and constraint detection in short texts. In *30th IEEE ICDE*. 280–291.
- [30] Emine Yilmaz, Evangelos Kanoulas, and Javed A. Aslam. 2008. A simple and efficient sampling method for estimating AP and NDCG. In *Proceedings of the 31st ACM SIGIR*. ACM, 603–610.