Entity Search by Leveraging Attributive Terms in Sentential Queries over RDF Data

Wiradee IMRATTANATRAI†, Makoto P. KATO†, and Katsumi TANAKA†

† Graduate School of Informatics, Kyoto University
36-1 Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501 Japan
E-mail: †{wiradee,kato,tanaka}@dl.kuis.kyoto-u.ac.jp

Abstract This paper proposes methods of finding relevant entities from RDF data for a given sentential query (e.g. "Cars 3", "Toy Story 4", or "The Incredibles 2" for the query "upcoming animated films pixar") by leveraging different types of modifiers in the query through identifying corresponding properties (e.g. **released** and **studio's industry** for the modifiers "upcoming" and "animated", respectively). While a major search engine like Google provides a function that shows a list of entities with respects to users' queries on their search result page, the entities are neither presented for a large fraction of users' queries, nor in the order that users expect. To enhance the efficiency of this function, we propose a method of finding entities based on the similarity between a query and entity type names together with the frequency of entities in search results returned in response to the query. We also propose a method of identifying a property corresponding to each modifier in a query based on the frequency of property values containing the modifier and co-occurrence of the modifier and property names. Moreover, the proposed method utilizes the difference in property value distributions of entities in the search results for a query with and without the modifier. The experimental results showed that our proposed methods could predict relevant entities based on relevant entity types for more queries than the existing function and achieved the best performance for identifying relevant properties when all of the criteria were combined.

Key words RDF, entity search, knowledge base

1. Introduction

Users often search using *entity type* names as a query in a Web search according to our survey which was performed by randomly sampling 100,000 queries from a query log. The result shows that approximately 36% of queries contained *entity type* names (the details are reported in Section 4.). Entity type names usually refer to a collection of entities. For example, the entity type name "Retail stores in America" refers to the entities "Walmart", "Costco", and "The Home Depot". This moderate fraction of queries suggests that many users want to find entity type related information that includes a list of entities from Web search results. To satisfy this information need, major search engines such as Google have started to provide a function that presents a list of entities at the top of the search engine result page for a given search query. Thus, providing entities in response to users' search queries becomes important as users can immediately find entity information without browsing among the search results.

However, since search queries are as varied as users' need, it is crucial for this kind of function to handle a wide variety of search queries such as "highest profitable companies japan", "upcoming american mystery movies" and "recent fantasy novels". Furthermore, queries sometimes contain modifiers such as "highest profitable", "upcoming", and "recent" that modify other components in the queries like entity type names. These modifiers are useful to narrow down the list of entities, yet the existing function does not select or sort entities by taking into account such modifiers. Accordingly, it is necessary to interpret modifiers in queries and sort entities in the order of their relevance to the modifiers.

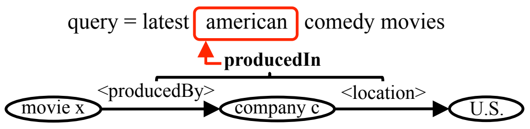


Figure 1: Example of multiple predicates as a property.

To overcome these limitations, we propose methods that utilize RDF (Resource Description Framework) data from a large-scale knowledge base (*e.g.* DBpedia(注1)) for 1) finding a set of entities for a larger fraction of Web search queries and 2) identifying properties that are relevant to the modifier terms in the query for entity ranking. The structure of RDF data is a collection of triples where each triple con-

(注1) : http://wiki.dbpedia.org/

sists of a subject, a predicate, and an object. Predicates within triples denote the relationship between pairs of subjects and objects. A set of such RDF triples can be viewed as a graph in which predicates are edge labels connecting subjects with objects. By utilizing the RDF data, we can easily capture the relationship among subjects and objects. For example, the subject "Japanese bands" and the object "Nintendo" are associated by the predicate <is a subject of> that represents the relationship between "Japanese bands" as the entity type and "Nintendo" as the entity. Likewise, the subject "Nintendo" as the entity is connected by the predicates <founder>, <numEmployees> and <revenue>. These predicates represent the different essences and link the subject "Nintendo" as the entity with different objects. By using these kinds of relationship, we can identify which entities are associated with entity types and which properties relevant to modifiers with respect to a given query. For example, given the query "highest profitable companies japan", we find a set of companies in Japan based on their entity types along with the property corresponding to the term "highest profitable", which is the property **revenue** derived from the predicate <revenue> that describes the amount of money earned by the companies. In addition, the corresponding properties can consist of one or more predicates. For example, the predicates (<producedBy>, <location>) as the property **producedIn** for the modifier "american" in the query "latest american comedy movies" as shown in Figure 1.

Our proposed method for finding a set of entities is based on the similarity between a query and entity type names as well as the document frequency of entities in the search results returned in response to the query. We identify a relevant property for a given modifier based on three criteria: 1) the frequency of property values including the modifier, 2) the co-occurrence of the modifier and property names, and 3) the difference in property value distributions of entities in the search results for a query with and without the modifier.

In the experiments, we used 40 queries to evaluate the effectiveness of our proposed methods. The experimental results showed that our methods could predict relevant entities for more queries than the existing function provided by a major search engine, and achieved the best performance for identifying relevant properties when all criteria were combined. The contributions of this paper are:

- We introduced two fundamental tasks: entity set retrieval and property identification for entity ranking.
- We proposed methods for the two tasks, which utilize the RDF data with some statistics that can be computed by performing a Web search.
- We conducted two experiments to evaluate the performance of the methods and demonstrated the effectiveness over the existing system.

The rest of the paper is organized as follows. Section 2. surveys related work on finding relevant entities and properties. Section 3. introduces our proposed methods, and Section 4. presents experimental results. Finally, Section 5. concludes the paper by outlining future work.

2. Related Work

This section introduces some related works to the problem of finding relevant entities and properties. One of the most related works is *entity ranking*, which has been addressed in some tracks in INEX and TREC [2], [4], [5], [7]. The INEX entity ranking track is comprised of two tasks: entity ranking and entity list completion tasks. The entity ranking task expects systems to return relevant Wikipedia articles (or entities) in response to a given query where there is assumption that all entities have the corresponding pages in Wikipedia. There are some proposed approaches for this task. Kaptein *et al.* proposed methods to rank Wikipedia entities by estimating relevant Wikipedia categories for a given query, and rank entities based on the query likelihood model with estimated categories [11]. Demartini *et al.* expanded queries for the entity ranking task in many different ways such as using hierarchical relationship in the ontology and synonyms, and extracting named entities from the queries [6]. Balog *et al.* proposed a probabilistic framework for the entity ranking task, which can model not only keyword queries, but also the other input such as categories in which relevant entities should belong to and examples of relevant entities [1].

The entity track in TREC 2010 introduced another task called *related entity finding*. This task is related to the entity ranking tracks in INEX although it is different in many ways: the main input is an entity name or a homepage of an entity and output is a list of homepages of entities. There are also some works that proposed methods for this task. For example, Bron *et al.* proposed methods to reduce problems in ranking by using four components including co-occurrence, type filtering, context modeling, and homepage finding [3].

Moreover, there are some related works about finding and ranking a set of RDF subgraphs by considering relevant predicates (or properties) based on keyword queries [9], [12]. The basic idea of the keyword search over graph-structured data is to match keywords with the elements in the graph, and to rank a set of related subgraphs containing the matching elements using the predicates. If the keyword often appears in relation to the predicates, its subgraphs would rank higher. Therefore, the top ranked subgraphs contains the related elements (or entities).

Similar to the keyword search problem, the question answering over RDF graph is also considered as one of related works [13], [15]. These works attempted to find answers (or entities) based on questions (or queries). The authors of these works also consider the modifier in the questions in which they finds the corresponding properties for the modifier terms. Zou *et al.* proposed methods to find relevant predicates or predicate paths (or properties) using supporting entity pairs for the relation phase (or modifier). For example, the relation phase "play in" has the supporting pairs such as Julia_Roberts with Runaway_Bride, while Unger *et al.* proposed methods that use of the pattern library extracted by the BOA framework [10] besides string matching for finding relevant properties.

More study related to the problem of finding properties corresponding to modifiers was authored by Zhang *et al.* [14]. In this work, relevant properties (of *facets*) were estimated for a given query in order to provide better snippets for structured documents. While a machine learning approach was employed for finding relevant properties, most of the features were textual similarity that often used for learning to rank for Web search (*e.g.* TF-IDF or BM25).

The problem of finding relevant entities is almost the same as the entity ranking tasks, keyword search and question answering over RDF except for types of queries in which we are interested in. The query type that we target includes the modifiers that are not necessarily included in the Wikipedia articles or the elements in the RDF graphs (*e.g.* "latest" or "most profitable"). Moreover, we use not only textual similarity based on a knowledge base, but also Web search engine results for finding relevant entities. This means that we make use of both structured and unstructured information for finding relevant entities.

The existing approaches for finding relevant properties and sequences of properties such as the textual similarity are not suitable for modifiers that represent numerical attributes (*e.g.* "oldest", "newest", and "latest"), for which we propose methods based on the co-occurrence of terms on the Web and property value distributions of entities.

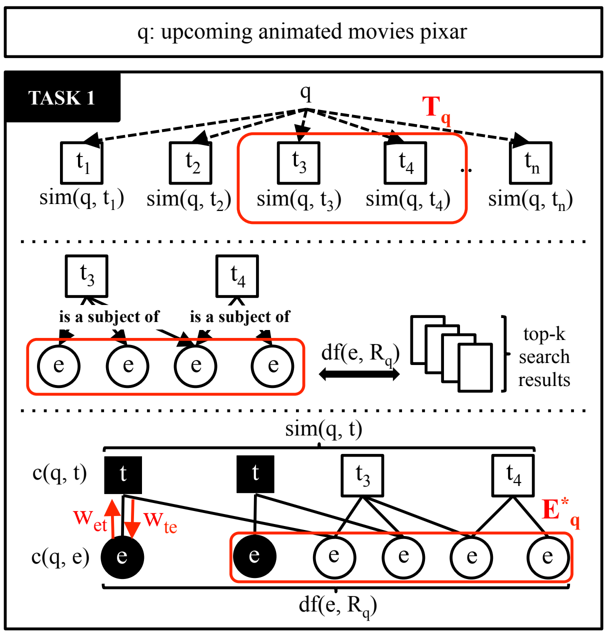
3. Methodology

In this section, we propose two fundamental tasks and methods for these tasks. The first task is entity set retrieval where given a query, we need to find a set of relevant entities. The second task is property identification, *i.e.* identifying the most appropriate properties for modifier terms in a query.

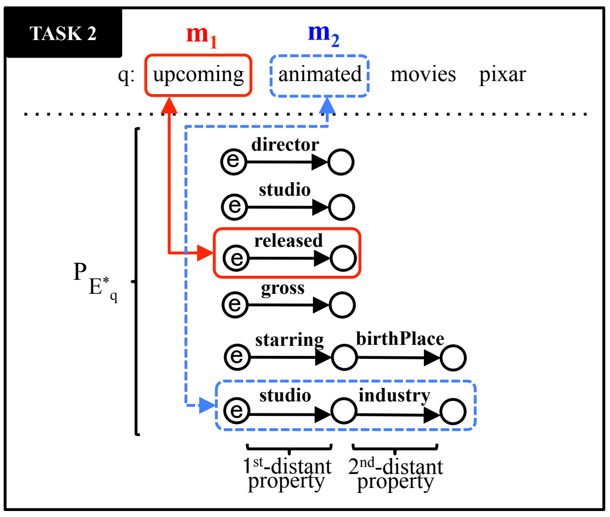
3.1 Problem Definition

Before going through each method, we formally define our problem in this subsection. Our problem is to return a ranked list of entities for a given query $q \in Q$. We are specifically interested in queries Q that contain entity type names and modifiers. A modifier is an attributive term that describes a typical feature of entities. Examples of modifiers are "oldest", "american", "comedy", and "private". Examples of queries Q are "ancient temples" ([modifier] [type]), "latest comedy movies" ([modifier] [modifier] [type]), and "universities new york" ([type] [modifier] [modifier]), where [modifier] and [type] in parentheses indicate that the term in the query is a modifier and entity type name, respectively.

As was mentioned earlier, we build on RDF data from a knowledge base that indicate a relationship between entities E and entity types T and properties P. Entities of the entity type $t \in T$ are denoted by E_t, entity types to which entity $e \in E$ belongs are denoted by T_e, and a set of properties of entity $e \in E$ are denoted by P_e. This set of properties consists of k^{th}-distant properties which are the edge labels of the k-edge far from entity $e \in E$ in the RDF graph where $k \in [1, K]$. In our work, the problem is addressed through two tasks: 1) finding a set of entities $E^*_q \subset E$ for query q and



(a) Entity set retrieval



(b) Relevant property identification

Figure 2: Overview of Two Fundamental Tasks

2) finding a set of relevant properties $P^*_q \subset P_{E^*_q} = \bigcup_{e \in E^*_q} P_e$ for query q, where E^*_q should belong to entity types relevant to the entity type names in query q, and P^*_q should correspond to the modifiers in query q. Having obtained E^*_q and P^*_q, we get a ranked list of entities by ranking E^*_q based on properties P^*_q.

In the following subsections, we describe methods for each of the two tasks for finding a set of candidate entities and a set of relevant properties. The overview of both tasks is illustrated in 2a and 2b.

3.2 Finding Set of Candidate Entities

Since we assume that a given query contains entity type names, entity types and their entities should be relevant if the name of the entity types is similar to the query. Here, we use the Okapi BM25 to measure their similarity. In addition, relevant entities may frequently appear in Web search results returned in response to the query as some Web pages list up entities that belong to a certain entity type.

3.2.1 BM25 Score between Query and Entity Type

We measure the similarity between query $q \in Q$ and entity type $t \in T$ using the Okapi BM25 as $\mathrm{sim}(q,t)$. We regard entity types whose $\mathrm{sim}(q,t)$ is $\theta\%$ (0.8 in our experiments) of the maximum score or larger as a set of entity types denoted as T_q:

$$T_q = \{\, t \mid \mathrm{sim}(q,t) \geqq \theta \max_{t' \in T}(\mathrm{sim}(q,t')) \,\} \qquad (1)$$

The set of candidate entities is $E_q = \bigcup_{t \in T_q} E_t$.

In some cases, even though the BM25 score is high, the resultant entity types T_q can still be irrelevant to the query. For example, given the query "suspension bridges in manhattan", T_q includes "Suspension bridges", "Bridges in Manhattan", "Suspension bridges in Japan", "Suspension bridges in the United States", and "Suspension bridges in Hungary", of which two are irrelevant to the query. To deal with this problem, we utilize the Web document frequency of entities.

3.2.2 Document Frequency of Entities in Web Search

To exclude irrelevant entities from T_q, we utilize the Web search results for query q. We issue the original search query q to a Web search engine and collect the top S search results R_q (S was set to 20 in our experiments). For each candidate entity $e \in E_q$, we count the number of search results that include the entity name in their content. More formally, the search result frequency is defined as $\mathrm{df}(e, R_q) = |\{\, r \mid r \in R_q \wedge e \in E_r \,\}|$ where E_r is the set of entities whose name is included in the content of search result r. Entities that have a high document frequency can be regarded as relevant to query q because they appear many times in the Web search results for q.

3.2.3 Co-HITS Algorithm using BM25 Score and Document Frequency

Given the BM25 scores of entity types and document frequency of entities, we utilize the Co-HITS algorithm [8] to combine these two criteria based on two assumptions: 1) Entity types are likely to be relevant if they contain relevant entities. 2) Entities are likely to be relevant if they belong to relevant entity types. As entities that have a high document frequency are likely to be relevant to query q, we can infer that entity types including such entities are also relevant. In a similar way, we can further infer that entities included in relevant entity types are also relevant. The Co-HITS algorithm can be used to make these inferences and enables us to find more relevant entities and entity types that would be missed when using methods described above.

The Co-HITS algorithm can be described with the following equations:

$$c(q,t) \;=\; (1-\lambda_T)\ \mathrm{sim}(q,t) + \lambda_T \sum_{e \in E_q} w_{et}\, c(q,e), \qquad (2)$$

$$c(q,e) \;=\; (1-\lambda_E)\ \mathrm{df}(e,R_q) + \lambda_E \sum_{t \in T_q} w_{te}\, c(q,t), \qquad (3)$$

where $c(q,t)$ and $c(q,e)$ are the Co-HITS scores for entity types and entities, w_{et} is the edge weight from entity e to entity type t, w_{te} is the edge weight from t to e, and λ_T and λ_E are the parameters that control the effect of the BM25 score and document frequency on the Co-HITS score. The edge weights indicate an is-a relationship between the entity types and entities and are defined as $w_{et} = 1/|T_e|$ and $w_{te} = 1/|E_t|$ if entity e belongs to entity type t; otherwise, the edge weights are zero. Entities with the highest Co-HITS scores are used as E^*_q and are later ranked by properties identified in the second task.

The Co-HITS score reflects our two assumptions. It becomes higher for entity type t if the Co-HITS scores for entities of t (the rightmost term in Equation 2) become higher, while it becomes higher for entity e if the Co-HITS scores for entity types of e (the rightmost term in Equation 3) become higher.

By using the example query "suspension bridges in manhattan", the top 5 entity types in terms of Co-HITS score are "Suspension bridges", "Bridges", "Suspension bridges in the United States", "Bridges in Manhattan", and "Bridges in New York City". This shows that the more relevant entity types can be ranked higher by combining the BM25 score and document frequency.

3.3 Finding Set of Relevant Properties

For this second task, we firstly identify the terms to be modifiers in a given search query. These modifiers are used to find corresponding properties based on the three methods: 1) counting the frequency of property values including the modifier, 2) measuring the co-occurrence of the modifier and property names, and 3) measuring the difference in property value distributions of entities in search results for a query with and without the modifier.

3.3.1 Identifying Query Terms to be Modifiers

Since we have found relevant entity types, we assume that the modifiers are terms that are not used for indicating the relevant entity types, *i.e.* ones that do not contribute to the BM25 scores of the relevant entity types. Specifically, for each term w in query q, we sum up all the BM25 scores of the entity types in T_q that contain term w in their entity type names. Then, we select term w as a modifier only if the sum of the BM25 scores for term w is relatively low. The set of modifiers in query q is denoted by M_q and obtained as follows:

$$M_q = \left\{\, w \;\middle|\; \textstyle\sum_{t \in T_q \wedge w \in n(t)} \mathrm{sim}(q,t) \leqq \phi \sum_{w' \in q} \sum_{t \in T_q \wedge w' \in n(t)} \mathrm{sim}(q,t) \,\right\}, \qquad (4)$$

where $n(t)$ represents a set of terms in the name of entity type t, and ϕ is a parameter that we set to 0.2 in our experiments. The left term in the inequality is the sum of BM25 scores for term w, while the right term is that for all the terms in the query.

For each modifier $m \in M_q$, we attempt to find a corresponding property from properties $P_{E^*_q}$ $(= \bigcup_{e \in E^*_q} P_e)$ that can appropriately describe the modifier.

3.3.2 Counting Frequency of Property Values

The first method to find relevant properties is to count the frequency of property values for each property p that contain modifier term m. Some kinds of modifiers indicate a

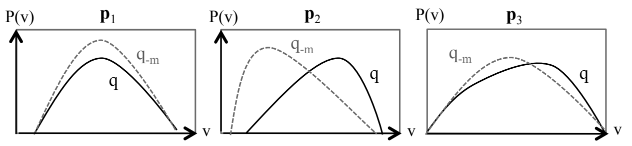


Figure 3: Distributions of q and q_{-m} for each property (p_j) where x-axis (v) represents the property value of p_j, and y-axis ($P(v)$) represents probability density of property value v. In these examples, largest distribution difference can be seen with property p_2, which is possibly relevant to modifier m.

certain type of entities (*e.g.* "zen" is a property value of relevant property <denomination> and "new york" is included in the property values of the relevant property <address>) and often specifies the property values of relevant properties. The frequency of property values including modifier m is computed as follows:

$$\text{freq}(q, m, p) = |\{ e \mid e \in E_q^* \wedge m \in p(e) \}|, \tag{5}$$

where $p(e)$ represents the property value of property p for entity e, which is represented as a set of terms in this case.

This method can only be applied to modifiers that appear as a part of property values. Some modifiers such as "latest", "oldest", and "recent" cannot be directly found in their relevant property by only looking at the property value. Thus, we need to use the second and third methods for these kinds of modifiers.

3.3.3 Measuring the Co-occurrence on the Web

The second method is to measure the co-occurrence of modifiers and property names on the Web. When people describe the characteristic of entities by modifiers (*e.g.* "old", "large", and "highest profitable"), they are likely to show their property values as evidence that prove the statement is reliable (*e.g.* "foundation date", "area size", and "income", respectively). Thus, a high co-occurrence between a modifier and a property name indicates that the modifier specifies a certain property value of the property. We compute how frequently modifier m co-occurs with property p as follows:

$$\text{co}(m, p) = \frac{|D_m \cap D_{n(p)}|}{|D_m|}, \tag{6}$$

where D_m is the set of web pages containing term m, and $n(p)$ is the name of property p.

3.3.4 Measuring KL Divergence between Property Value Distributions

The third method is to measure the difference in property value distributions of entities in search results for a query with and without the modifier. We expect that Web search engines can return search results containing some entities whose property values are relevant to modifiers when the original query is used as a Web search query, but they cannot do this when modifiers are excluded from the query. Based on this assumption, we can measure the change in property value distributions of each property for these cases and estimate that properties that cause big changes are relevant to modifiers excluded from the query. Figure 3 shows some examples of pairs of property value distributions. We estimate that property p_2 is relevant to modifier m since the distributions for the original search query (q) and original search query without modifier term m (q_{-m}) are significantly different.

The property value distribution of property p for query q can be estimated as follows:

$$P_p^q(v) = \frac{|\{ e \mid e \in E_{R_q} \wedge p(e) = v \}|}{\sum_{v' \in V_p} |\{ e \mid e \in E_{R_q} \wedge p(e) = v' \}|}, \tag{7}$$

where v is a property value ($v \in V_p$; V_p is the set of all possible property values for property p), and E_{R_q} is a set of entities included in the search results for query q (formally defined as $E_{R_q} = \bigcup_{r \in R_q} E_r$).

Letting q_{-m} be query q with modifier m excluded (*i.e.* $q - \{m\}$ if q is represented as a set of terms), the difference between property value distributions for queries q and q_{-m} is measured by Kullback-Leibler divergence as follows:

$$\text{diff}(q, m, p) = D(P_p^q \parallel P_p^{q_{-m}}) = \textstyle\sum_{v \in V_p} \log P_p^q(v) \frac{P_p^q(v)}{P_p^{q_{-m}}(v)}. \tag{8}$$

3.3.5 Combination of Three Criteria for Identifying Relevant Properties

Lastly, after applying all three methods to every property p for each modifier $m \in M_q$, we need to find most appropriate property by linearly combining these methods:

$$f(q, m, p) = \alpha_1 \text{freq}(q, m, p) + \alpha_2 \text{co}(m, p) + \alpha_3 \text{diff}(q, m, p), \tag{9}$$

where parameter α_1 should be higher than the other parameters α_2 and α_3 according to our preliminary experiments. Thus, we opted to use $\alpha_1 = 0.8$, $\alpha_2 = 0.1$, and $\alpha_3 = 0.1$ in our experiments described in the next section.

We selected the property with the highest $f(q, m, p)$ for each modifier $m \in M_q$ and finally obtained a set of properties $P_q^* = \{ \text{argmax}_{p \in P_{E_q^*}} \lambda_{e,p} f(q, m, p) \mid m \in M_q \}$ where $\lambda_{e,p} = \frac{1}{d(e,p)}$. $d(e, p)$ denotes the distance of property p from entity $e \in E$ since the nearer the property is to the entity, the more relevance it can be for the modifier.

4. Experiments

In this section, we explain our survey on the entity type names in queries, introduce evaluation methodology for our proposed methods, and show and discuss our experimental results.

In the experiments, we used RDF data from DBpedia as knowledge base to evaluate our methods. We obtained a set of entity types, entities, properties and property values from DBpedia, which contains structured information derived from Wikipedia. We extracted pairs of an entity and an entity type by using the predicate <is a subject of> as the property, where the subjects and objects are treated as entity types and entities, respectively. For extracting properties and property values, we retrieved all triples whose the subjects are the entities, and treated the predicates as the property names and the objects as the property values. After the

Table 1: Precision, Recall and F-score for entity types for 25 combinations of λ_E and λ_T.

(a) Precision

$\lambda_E \backslash \lambda_T$	0.00	0.25	0.50	0.75	1.00
0.00	**0.659**	0.553	0.554	0.552	0.569
0.25	0.517	0.554	0.560	0.554	0.571
0.50	0.517	0.543	0.549	0.562	0.571
0.75	0.517	0.540	0.547	0.552	0.550
1.00	0.517	0.504	0.499	0.493	0.494

(b) Recall

$\lambda_E \backslash \lambda_T$	0.00	0.25	0.50	0.75	1.00
0.00	0.260	0.443	0.445	0.441	0.456
0.25	0.198	0.448	0.452	0.446	**0.457**
0.50	0.198	0.429	0.442	0.449	0.453
0.75	0.198	0.438	0.446	0.446	0.438
1.00	0.198	0.413	0.406	0.398	0.371

(c) F-score

$\lambda_E \backslash \lambda_T$	0.00	0.25	0.50	0.75	1.00
0.00	0.325	0.449	0.451	0.448	0.463
0.25	0.254	0.453	0.457	0.451	**0.465**
0.50	0.254	0.439	0.449	0.458	0.464
0.75	0.254	0.442	0.448	0.450	0.446
1.00	0.254	0.413	0.408	0.400	0.391

Table 2: Examples of top five entity types obtained by best parameter values for λ_E (0.25) and λ_T (1.00) with the Co-HITS algorithm.

Query	**Entity Types**
upcoming animated films pixar	Films directed by John Lasseter Films featuring anthropomorphic characters Films directed by Rob Zombie Films directed by Andrew Stanton 2000s science fiction films
newest video games 2016	Upcoming video games scheduled for 2016 PlayStation 4 games Video games set in 2016 Video games featuring female protagonists Dinosaurs in video games
private business schools europe	Business in Europe Education in Budapest Europe Business Schools Business schools in Germany Universities and colleges in Cyprus
large companion dog breeds	Dog breeds Dog breeds originating in Germany Companion dogs Spaniels Sighthounds

extraction, we got 974,417 entity types including 4,811,226 entities and 60,252 properties from all the entities in total.

4. 1 Survey on Entity Type Names in Queries

To find out that how many search queries issued by users contain an entity type name, we randomly picked 100,000 queries from an AOL query log and found queries containing entity type names from DBpedia. We used the BM25 score to measure the similarity between queries and entity type names and selected queries with high scores using the same criterion as that used in finding relevant entity types. Queries were considered as ones including an entity type name only if they were retrieved by BM25 and satisfied the criteria shown below:

$$\exists t \in T \left(\frac{|q \cap n(t)|}{|q|} \geq \beta_1 \wedge \frac{|n(t)|}{|q|} \geq \beta_2 \right), \qquad (10)$$

where q represents the set of terms in the query, $n(t)$ is a set of terms in the name of entity type t, and β_1 and β_2 were set to 0.65 and 0.5, respectively. These two criteria guaranteed that 65% or more terms in the queries were terms in an entity type name, and the number of terms in the queries was double or less than that in the entity type name. For example, "retail stores america" and "taylor swift songs" were counted because of entity types "Retail stores in America" and "Songs written by Taylor Swift", whereas "walmart" and "taylor swift" were not. As a result, we found that approximately 36% of the search queries contain entity type names, and conclude that our methods can be applied to a moderate fraction of Web queries.

4. 2 Evaluation Methodology

We separately evaluated resultant entities and properties for measuring the performance of our proposed methods in detail. In the experiments, we used 40 search queries that contain modifiers and an entity type name (examples of the queries can be found in Table 2).

4. 2. 1 Evaluation of Entities

To evaluate the performance of the proposed methods for finding candidate entities, for simplicity, we opted to evaluate entity types rather than entities since the relevance of retrieved entity types usually indicates that of entities in our problem setting.

For each query, we evaluated 10 entity types with the highest Co-HITS scores ($c(q,t)$) for 25 possible combinations of the two parameters involved in the Co-HITS algorithm, where λ_T and λ_I were set to 0.00, 0.25, 0.50, 0.75, and 1.00. Parameter λ_T controls the effect of the BM25 score of entity types, while parameter λ_I controls the effect of document frequency of entities on the Co-HITS score.

As a result, we obtained 25 sets of entity types for each query, and *pooled* the results for evaluation. The relevance assessment was carried out by three assessors including two of the authors. An entity type was considered relevant if at least one of its entities was relevant for a given query. The inter-rater agreement was measured by Fleiss' Kappa and was considered to be moderate at 0.601. We used majority voting to decide the relevance of each entity type.

We computed the precision, recall, and F-measure based on the relevance given by three assessors. Moreover, we compared our methods with the existing function provided by Google. We computed the percentage of queries for which the top ranked entity type was relevant for our methods and percentage of queries for which Google's function can present a list of entities. Note that these results are not perfectly comparable due to the difference in the metrics, but they can still show the coverage of queries achieved by our methods and Google's function.

4. 2. 2 Evaluation of Properties

This part of the evaluation was designed for evaluating the performance of the three methods for finding properties relevant to modifiers in queries. In this part, we initially set K to 1 for finding a set of properties of entities with the same set of queries as that used in the evaluation of entities. We detected 49 modifiers from 40 queries by the method explained in Section 3. 3, and evaluated the top five properties for each modifier using the following methods: Method I: Counting the frequency of property values (Equation 5), Method II: Measuring the co-occurrence on the Web (Equation 6), Method III: Measuring KL divergence (Equation 8), and Method IV: Combination of Methods I, II, and III (Equation 9). No baseline method was used due to lack of existing methods.

The relevance of each property was assessed by the same assessors as those involved in the evaluation of entities. They were required to judge a property as relevant only if a given modifier seemed to specify a certain type of entity by property values of the property in a sequence. The inter-rater agreement was measured by Fleiss' Kappa and was considered to be substantial at 0.745. Majority voting was used to decide the relevance of each property.

We compared the results by Mean Reciprocal Rank (MRR), which is defined as follows: $\mathrm{MRR} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{r_i}$, where M is the number of modifiers in our experiment (*i.e.* 49), and r_i denotes the rank of the first relevant property for the i-th modifier. The MRR is often used for evaluating ranked lists where only the first relevant result matters and is suitable for this evaluation since we expect only a relevant property can be used for each modifier.

4. 3 Experimental Results for Entities

Table 1a and 1b show the precision and recall of the resultant entity types for 25 combinations of parameter λ_E and λ_T, respectively. We noticed that the highest precision (0.659) was achieved when $\lambda_E = 0.00$ and $\lambda_T = 0.00$. Although the highest precision was achieved when $\lambda_T = 0.00$, the lowest recalls were obtained (0.198) with the same setting. Since $\lambda_T = 0.00$ indicates that the Co-HITS score was determined solely by the BM25 score, this result suggests that only the BM25 score can achieve high precision but low recall.

We achieved high performance for both of the precision and recall with $0.00 \leqq \lambda_E \leqq 0.50$ and $\lambda_T = 1.00$. Among these parameter values, the highest recall was achieved when $\lambda_E = 0.25$ and $\lambda_T = 1.00$. Moreover, we obtained the highest F-measure for $\lambda_E = 0.25$ and $\lambda_T = 1.00$ as shown in Table 1c. Table 2 shows examples of entity types obtained by the best parameter values. This result suggests that we should not rely much on the similarity, but we should give a high weight to the document frequency in the Co-HITS algorithm. Note, however, that this does not mean the BM25 score is not necessarily used in our method as the BM25 score is still required to retrieve entities used in the Co-HITS algorithm. We could get more irrelevant entities if we directly found entity names from Web search results without knowing the candidate entities by the entity types, since irrelevant but common entity names may appear in the search results (*e.g.* "I" (a song title) or "IT" (a film title)).

We also compared our method with Google's function. To this end, we computed the percentage of queries whose top ranked entity type was relevant with the best combination of parameters ($\lambda_T = 1.00$ and $\lambda_E = 0.25$) and percentage of queries for which Google's function can present a list of entities. We founded that our methods (57.5%) performed approximately twice as well as Google's (25.0%). Again, note that different metrics were used for ours and Google's, and this comparison could be an over- or under-estimation of the performance of our method.

Table 3: MRR for results of identifying relevant properties.

	Method I	Method II	Method III	Method IV
MRR	0.326	0.223	0.194	**0.426**

Table 4: Examples of properties ranked at top by each method for identifying relevant properties.

Method	Modifier	Properties	Example of property values
I	kyoto	**address**	Fushimi-ku, Kyoto
	italian	**foodType**	Italian cuisine
II	ancient	**founded**	1049-11-22
	recent	**time**	1200.0
	latest	**year**	1999
III	ancient	**founded**	1049-11-22
	large	**femaleheight**	75.0
	oldest	**established**	mid 10th century
IV	ancient	**founded**	1049-11-22
	kyoto	**address**	Fushimi-ku, Kyoto
	oldest	**established**	mid 10th century
	recent	**time**	1200.0

4. 4 Experimental Results for Properties

Table 3 shows the MRR achieved by each method for identifying relevant properties and indicates that combining all of the three methods (Method IV) could achieve the best performance.

To drill down the results in Table 3, Figure 4 illustrates the performance in terms of the RR achieved for each modifier, where the x-axis represents the modifiers, and the height of each portion of the stacked bars represent the MRR achieved by each method. Note that the total height of the stacked bars does not represent the overall performance: it simply means the average performance of all the methods. It can be seen that the different methods could identify relevant properties for different types of modifiers. For some modifiers, relevant properties could be found at high ranks by Method I, while Methods II and III could not rank them near the top (*e.g.* 10th and 20th modifiers from the left). On the other hand, if relevant properties could be ranked high by Methods II or III, Method I could not rank them near the top in some cases (*e.g.* 14th and 16th modifiers from the left). These observations suggest that the combination of the three methods (Method IV) could identify relevant properties most accurately by complementing each other.

Table 4 shows examples of relevant properties found at the top by each method. It suggests that if the modifier specifies a categorical value such as "kyoto", "zen", and "italian", their properties can be successfully found by Method I. If the

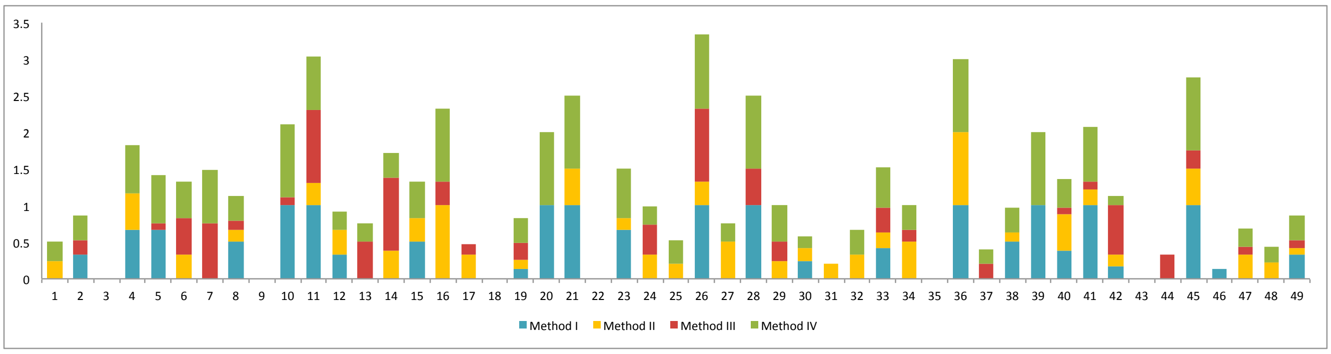


Figure 4: Reciprocal Rank (RR) of four methods for identifying relevant properties.

modifier specifies a certain range of numerical values or a certain order by numerical properties (*e.g.* "latest", "recent" and "oldest"), their properties can be identified by either Method II or Method III. These results are also consistent with our observation in Figure 4 that different methods could identify relevant properties for different modifiers, and they suggest that Method IV achieved the best performance because the three different methods complemented each other.

5. Conclusions

This paper proposes methods of finding relevant entities for a given sentential query by leveraging different types of modifiers in the query using RDF data. We proposed a method of finding entities based on the similarity between a query and entity type names together with the frequency of entities in search results returned in response to the query. We also proposed a method of identifying a property corresponding to each modifier in a query using the methods 1) counting the frequency of property values including the modifier, 2) measuring the co-occurrence of the modifier and property names, and 3) measuring the difference in property value distributions of entities in search results for a query with and without the modifier. Experimental results showed that our proposed methods could predict relevant entity types for more queries than the existing function and achieved the best performance for identifying relevant properties when all of the criteria were combined.

Our implementation mainly focused on finding the relevant entities and properties for a given sentential query with attributive terms. In addition, for our future works, we aim to find other property identification methods that can handle the properties which are derived from multiple predicates in the RDF data. After that, we are going to apply those relevant properties as for ranking the resultant relevant entities. To actually rank the entities, it may also requires various kinds of methods.

Acknowledgement This work was supported in part by Grants-in-Aid for Scientific Research (Nos. 15H01718, and 26700009) from MEXT of Japan and Yahoo Japan Corporation.

References

[1] K. Balog, M. Bron, and M. De Rijke. Query modeling for entity search based on terms, categories, and examples. *ACM TOIS*, 29(4):22:1–22:31, 2011.

[2] K. Balog, P. Serdyukov, and A. P. d. Vries. Overview of the TREC 2010 entity track. In *TREC*, 2010.

[3] M. Bron, K. Balog, and M. De Rijke. Ranking related entities: components and analyses. In *CIKM*, pages 1079–1088, 2010.

[4] A. P. De Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the INEX 2007 entity ranking track. In *INEX*, pages 245–251, 2007.

[5] G. Demartini, A. P. de Vries, T. Iofciu, and J. Zhu. Overview of the INEX 2008 entity ranking track. In *INEX*, pages 243–252, 2008.

[6] G. Demartini, C. S. Firan, T. Iofciu, R. Krestel, and W. Nejdl. Why finding entities in Wikipedia is difficult, sometimes. *Information Retrieval*, 13(5):534–567, 2010.

[7] G. Demartini, T. Iofciu, and A. P. De Vries. Overview of the INEX 2009 entity ranking track. In *INEX*, pages 254–264, 2009.

[8] H. Deng, M. R. Lyu, and I. King. A generalized Co-HITS algorithm and its application to bipartite graphs. In *KDD*, pages 239–248, 2009.

[9] S. Elbassuoni and R. Blanco. Keyword search over rdf graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 237–242. ACM, 2011.

[10] D. Gerber and A.-C. N. Ngomo. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction@ ISWC*, volume 2011, 2011.

[11] R. Kaptein, P. Serdyukov, A. De Vries, and J. Kamps. Entity ranking using Wikipedia as a pivot. In *CIKM*, pages 69–78, 2010.

[12] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 405–416. IEEE, 2009.

[13] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM, 2012.

[14] L. Zhang, Y. Zhang, and Y. Chen. Summarizing highly structured documents for effective search interaction. In *SIGIR*, pages 145–154, 2012.

[15] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM, 2014.