Combining Inverted Indices and Structured Search for Ad-hoc Object Retrieval

Alberto Tonon, Gianluca Demartini, and Philippe Cudré-Mauroux
eXascale Infolab
U. of Fribourg—Switzerland
{firstname.lastname}@unifr.ch

ABSTRACT

Retrieving semi-structured entities to answer keyword queries is an increasingly important feature of many modern Web applications. The fast-growing Linked Open Data (LOD) movement makes it possible to crawl and index very large amounts of structured data describing hundreds of millions of entities. However, entity retrieval approaches have yet to find efficient and effective ways of ranking and navigating through those large data sets. In this paper, we address the problem of Ad-hoc Object Retrieval over large-scale LOD data by proposing a hybrid approach that combines IR and structured search techniques. Specifically, we propose an architecture that exploits an inverted index to answer keyword queries as well as a semi-structured database to improve the search effectiveness by automatically generating queries over the LOD graph. Experimental results show that our ranking algorithms exploiting both IR and graph indices outperform state-of-the-art entity retrieval techniques by up to 25% over the BM25 baseline.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval—*retrieval models*; H.3.4 [**Information Storage And Retrieval**]: Systems and Software—*performance evaluation (efficiency and effectiveness)*

General Terms

Algorithms, Experimentation, Performance

Keywords

Ad-hoc Object Retrieval, Entity Search, LOD

1. INTRODUCTION

Many modern websites, such as Web portals or news aggregators, are today including entity-centric functionalities. Common examples include the aggregation of all pieces of content related to a given person, or the extraction of the most important entities appearing in a given article. Some companies, like the New York Times, manually maintain a directory of entities and ask human experts to create links between their resources (e.g., news articles) and the corresponding entities (e.g., celebrities appearing in the articles). Increasingly, however, websites are turning to automated methods due to the sheer size of the resources they have to analyze, and the large number of entities they have to consider.

Recently, the Linked Open Data (LOD) movement[1] started an effort to make entity data openly available on the Web. In this initiative, Uniform Resource Identifiers[2] (URIs) are used to identify entities. Each entity can be looked up (*dereferenced*) online, where it is described and linked to further entities using the Resource Description Framework[3] (RDF). The fundamental difference between LOD and standard entity datasets like, for instance, Wikipedia, lies in the inherent structure of the data. On the LOD cloud, the data is provided by uncorrelated parties and is given as a giant graph of semi-structured data.

In the context of online entity search, the TREC Entity track [3] has studied two related search tasks: "Related Entity Finding" (i.e., finding all entities related to a given entity query) and "Entity List Completion" (i.e., finding entities with common properties given some examples). The SemSearch challenge[4] focused on Ad-hoc Object Retrieval (AOR), that is, finding the entity identifier of a specific entity described by a user query [20].

This paper focuses on Ad-hoc Object Retrieval over semi-structured data. We propose a novel search architecture that exploits both IR and graph data management techniques to effectively and efficiently answer AOR queries. Specifically, we propose a hybrid solution that starts by retrieving an initial list of results from an inverted index using a ranking function, and then re-ranks and extends the result list by exploiting the underlying graph representation of the data. Our extended experimental evaluation performed over standard collections shows that our proposed solution significantly improves the effectiveness (up to 25% improvement over the chosen baseline) while maintaining very low query execution times. We consider our results as especially promising since the LOD test collections that are

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

[1] http://linkeddata.org/
[2] http://tools.ietf.org/html/rfc3986
[3] http://www.w3.org/RDF/
[4] http://semsearch.yahoo.com

today available are noisy and incomplete, and since we expect both the quality and the coverage of LOD datasets to rapidly improve in the future.

In summary, the main contributions of this paper are:

- A comparison of entity search techniques for AOR tasks including state-of-the-art IR techniques such as query extension and pseudo relevance feedback.
- A new hybrid search architecture for AOR that combines IR and graph data management techniques exploiting the graph structure of the underlying data.
- A new, fairer and continuous evaluation methodology for relevance assessment based on iterative crowsdsourcing.
- An extensive evaluation of our novel search system for AOR using two standard test collections.

The rest of the paper is structured as follows. We discuss related work, focussing on Entity Search and AOR approaches, in Section 2. We briefly introduce the Linked Open Data movement in Section 3. Section 4 is devoted to a high-level description of our hybrid system. We describe several approaches for AOR based on inverted indices and NLP techniques in Section 5, and describe complementary techniques based on structured queries and graph data management techniques in Section 6. In Section 7, we experimentally compare the performance of our hybrid approach to a series of state-of-the-art AOR approaches on two standard test collections. Finally, we present our conclusions in Section 8.

2. RELATED WORK

Searching for entities instead of documents or Web pages is a recent trend in IR. Early approaches focused on single-type entity search, for example in the context of the expert finding task at the TREC Enterprise track [4], where standard IR approaches such as language modeling [1] have been applied. Complex entity search tasks have been addressed more recently. For instance, the INEX Entity Ranking track [12] studied the problem of finding entities matching a keyword query (e.g., "Countries where I can pay in Euro") using Wikipedia. The Entity track at TREC [3] evaluated the *Related Entity Finding* task, which aims at finding entities related to a given entity (e.g., "Airlines using the Boeing 747") using both a collection of Web pages as well as a collection of RDF triples. Several effective approaches for this task focus on entity type and co-occurrence information [7, 17]. The task of ranking entities appearing in one document taking into account the time dimension has been studied in [13].

2.1 Ad-hoc Object Retrieval

The main goal of the SemSearch Challenge is to create evaluation collections for the task of Ad-hoc Object Retrieval [20] on the Web of data. This task consists in retrieving entity identifiers (i.e., URIs) given a keyword query describing the single entity the user is looking for (e.g., "harry potter"). Evaluation collections for this task have been created by crowdsourcing relevance judgements [5].

The first approaches for AOR exploited standard IR techniques that had previously been used for other entity search tasks such as expert finding. The basic idea is to construct an entity profile by collecting and aggregating all information available about an entity and to index the resulting collections of entity profiles with standard IR techniques in order to answer entity search queries. Looking back at previous work from entity search, several techniques can be applied to improve the effectiveness of the AOR task. For example, different data types can be used to rank entities for AOR based on an inverted index and BM25F [21] as a ranking function [6].

Various methods already proposed for different entity search tasks can be exploited for AOR as well, such as approaches exploiting probabilistic models [2]. In [11], Demartini *et al.* adopt structured and Natural Language Processing approaches to improve the effectiveness of entity search. We suggest further AOR approaches relying on recent IR techniques in Section 5.

The contribution of this work is a novel hybrid approach that benefits both from known IR ranking functions as well as from an analysis of the graph structure of the entities. We compare below our novel approach against state-of-the-art entity search techniques.

2.2 Hybrid Search

A number of hybrid search systems have already been developed. In [14], Elbassuoni and Blanco propose ranking models for keyword queries over RDF data. Their task is different from AOR as they aim at selecting subgraphs matching the query and then ranking them by means of statistical language models. CE^2 [23] is a hybrid IR-DBMS system that provides ranking schemes for hybrid queries (i.e., keyword queries that describe joins and predicates between entities) over RDF data. $Beagle^{++}$ [18] is a desktop search engine that exploits both an inverted index and a structured repository for file metadata to provide more effective search functionalities to the desktop user. SIREn [10] is a hybrid Web search framework that supports keyword as well as structured queries over RDF data. SIREn focuses on scalability and efficiency through novel indexing schemes that can index data and answer user queries efficiently by using an inverted index. Instead, we propose methods to improve ranking effectiveness for AOR.

3. LOD PRIMER

LOD is an online movement whose origins can be traced back to a note[5] published by Tim Berner-Lee a few years ago. LOD suggests to publish data on the Web following four principles: i) using URIs to identify things ii) using HTTP URIs such that things can be dereferenced online (using for example a Web browser) iii) providing useful, structured information about the things when they are dereferenced, using for example RDF/XML and iv) including links to other, related URIs in the exposed data to foster data aggregation and discovery.

Hence, LOD can be seen as a grassroots effort leveraging on the current architecture of the Web (HTTP, URIs, Content Negotiation, XML, RDF, webservers and browsers) to publish data. Figure 1 below includes on the right-hand side a macroscopic view on the LOD cloud as of September 2011, where each node depicts a separate data set and the edges symbolize links between the data sets[6]. As of Septem-

[5] `http://www.w3.org/DesignIssues/LinkedData.html`

[6] Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. `http://lod-cloud.net/`

ber 2011, the LOD cloud contains approximatively 300 data sets and more than 30 billion RDF triples[7]. In our context, each RDF triple can be seen as a simple sentence, connecting a *subject* (an entity URI), a *property* (a predicate URI) and an *object* (which can be either a URI or a literal). The following triple, for instance

```
(example.net/Alice, foaf:age, ‘‘17’’)
```

encodes the fact that Alice (as identified by the URI `example.net/Alice`) is 17 years old.

Since the same URI can be used as a subject or object in different triples, RDF naturally forms labelled graphs that connect entities to values and further entities. In the following, we make a distinction between object properties—linking entities to other entities—and datatype properties—associating values to a given entity[8].

Large collections of triples are typically stored in a database system (such as Oracle 11g[9], Virtuoso[10] or RDF-3X [19]). LOD data sets are typically referenced online (e.g., on CKAN[11]) and can be queried or crawled over HTTP using the SPARQL[12] declarative query language.

4. SYSTEM ARCHITECTURE

We describe the architecture of our hybrid search system in the following. Our approach is based on an inverted index that supports full text search on one hand, and on a structured repository, which maintains a graph representation of the original data, on the other hand. The ability to query an inverted index and to obtain a ranked list of results allows us to retrieve an initial set of entities that match the user query. Starting from this set of retrieved entities, we take advantage of algorithms exploiting the graph structure of the data thanks to our structured repository: retrieved entities are nodes in the data graph that can be used as starting points for graph traversals and neighborhood queries. Thus, the structured repository is used to refine the original IR-based results by navigating the data-graph, selecting potentially new entities or reinforcing the relevance of the results selected through the inverted index.

Figure 1 illustrates the main components of our system and their interactions. The search process proceeds as follows. First, a user initiates an entity search through a keyword query. The query is parsed and extended (see below Sections 5.2 and 5.3). A first set of results is retrieved using the expanded query and an inverted index on the LOD data. Several mechanisms can be used at this stage to improve ranking effectiveness, such as a multi-field index with BM25F scoring, NLP, or pseudo-relevance techniques (see below Section 5 for a detailed discussion on that point.)

This first set of results is then enriched using structured queries on (a subset of) the LOD graph. We take advantage of two types or graph queries in that context: scope queries following datatype properties (see Section 3), and graph traversal queries following object properties. Both types of queries start from the LOD nodes corresponding to the first set of results, and retrieve additional information (sets of literals and sets of entities respectively) to refine the results. As the LOD graph is highly connected (i.e., there are many properties that connect entities among themselves or that connect entities to literals), we must at this stage identify the most effective properties to follow in order to limit the scope of the graph queries. The details of our graph approach are given in Section 6.1. At this point, new entities may be added to the result set and previously identified entities may change position in the rankings due to the new information gathered. Finally, the top-k results are sent back to the querier.

[7] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`

[8] We hence follow the vocabulary introduced by the Web Ontology Language OWL, see `http://www.w3.org/TR/owl-ref/`

[9] `http://www.oracle.com/technetwork/database/options/semantic-tech/index.html`

[10] `http://virtuoso.openlinksw.com/`

[11] `http://thedatahub.org/group/lodcloud`

[12] `http://www.w3.org/TR/rdf-sparql-query/`

5. INVERTED INDEX AND AD-HOC OBJECT RETRIEVAL TECHNIQUES

The first approach to index graph-structured datasets such as those found in the LOD cloud is to aggregate all information attached to the entities. Thus, we create *entity profiles* by building an inverted index on the entity labels. We consider each entity profile as a document (similarly to candidate-centric expert finding approaches [1]) containing all versions of the entity name directly attached to it in the graph. This index structure considering entity profiles as bag-of-words combined with a BM25 ranking function will be our baseline approach. We describe below three further approaches we propose to improve this baseline. These are based on a structured inverted index, on NLP techniques (query expansion), and on pseudo-relevance feedback.

5.1 Structured Inverted Index

First, we consider a structured inverted index approach, which separately indexes different values attached to the entity. In order to create a structured inverted index, we index separately different types of information attached to the entity. This is similar to previous work [6]. Specifically, we create a structured inverted index for the three following pieces of information:

URI First we tokenize the URI identifying the entity in the LOD cloud. As the URI often contains the entity name (e.g., `http://dbpedia.org/page/Barack_Obama`), this field often matches the user query well.

Labels We also consider a list of manually selected datatype properties (i.e., *label*, *title*, *name*, *full-name*, *family-name*, *given-name*, *has-pretty-name*, *prefLabel*, *given-name*, *nickname*, which frequently occur in various LOD datasets) that point to a label or textual description of the entity.

Attributes Finally, we consider all the other datatype properties (i.e., non-label attributes) and attach their values to the entity.

Once the indices are created, we obtain a ranked list of results by means of various ranking functions such as BM25F.

5.2 Query Expansion

Natural Language Processing techniques have been applied to other entity search tasks. In this paper, we exploit query expansion and relevance feedback techniques on top of the BM25 baseline for the AOR task.

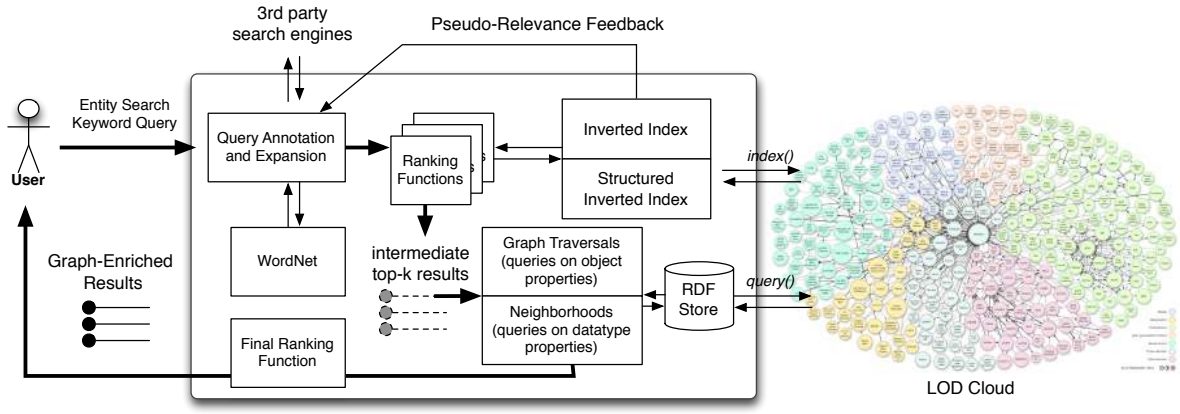

Figure 1: Our hybrid search system, where results retrieved from an inverted index are refined through structured graph queries.

Specifically, given a disjunctive keyword query q we extend it by adding additional related terms such as synonyms, hypernyms, and hyponyms from Wordnet [15]. This makes the query (e.g., "New York") retrieve additional results including different ways of referring to the specified entity (e.g., "Big Apple").

In addition, we exploit commercial search engines query autocompletion features to obtain additional keywords, given the original user query. Specifically, we send the original query q to a commercial search engine (Google) and we expand it by appending the first 5 terms the autocompletion algorithm suggests for q (e.g., "New York Times").

5.3 Pseudo-Relevance Feedback

Finally, we also implement pseudo-relevance feedback techniques on top of the baseline approach by first running the original query q and then considering the labels of the top-3 retrieved entities to expand the user query. For instance, the query "q=NY" can be expanded to "New York", "New York City" and "North Yorkshire" (labels of the top-3 entities retrieved for the original query q).

We report evaluation results comparing those various techniques on standard AOR collections in Section 7.

6. GRAPH-BASED ENTITY SEARCH

All the techniques described above neglect two key characteristics of LOD data: i) LOD entities are *interlinked*, in the sense that related entities are often connected, and ii) data values are *clustered* in the graph, such that many relevant values can be retrieved by following series of links iteratively from a given entity. In this section, we describe how we can leverage structured graph queries to improve the performance of our system taking advantage of those two characteristics.

As previously mentioned, our hybrid ranking solution consists of two steps. The first step (described in the preceding section) uses the IR infrastructure to obtain an initial ranking of entity identifiers. In the second step, the graph structure is exploited. We use the first set of entities from the original ranking as seeds in the graph. From those initial points, we traverse the graph following promising edges, that is, we follow object property edges potentially leading to additional results, and datatype properties leading to additional values.

Let $Retr = \{e_1, e_2, .., e_n\}$ be the ranked list of entities resulting from the query to the inverted index. Our goal is to define a set of functions that use the top-N elements of $Retr$ to create an improved list of results $StructRetr = \{e'_1, e'_2, .., e'_m\}$. This new list of results may contain entities from $Retr$ as well as new entities discovered when traversing the LOD graph. To obtain $StructRetr$, we exploit top LOD properties that are likely to lead to relevant results[13].

Once $StructRetr$ is obtained, we create a final result set and return it to the user. We adopt a simple model that linearly combines the two entity rankings:

$$finalScore(e') = \lambda \cdot BM25(q, e) + (1 - \lambda)(score(q, e, e'))$$

where q is the user query, $score(q, e, e')$ is our new ranking function (see below Section 6.2), and $BM25(q, e)$ is the initial score of the entity $e \in Retr$ (or of the entity in $Retr$ that lead to $e' \in StructRetr$ by following RDF properties).

6.1 Entity Graph Traversals Using SPARQL Queries

We describe below how we take advantage of SPARQL queries to retrieve additional results by following object property links in the LOD graph.

Structured Queries at Scope One.

Our simplest graph traversal approach looks for candidate entities directly attached to entities that have already been identified as relevant (i.e., entities in $Retr$). First, we need to ensure that only meaningful edges are followed. The object property `<dbpedia:wikilink>`, for instance, represents links between entities in Wikipedia. As an example, the entity 'dbpedia:Stevie_Wonder' has a `<dbpedia:wikilink>` pointing to 'dbpedia:Jimi_Hendrix'. Since the AOR task aims at finding all the identifiers of one specific entity, this kind of links is not very promising.

[13]It is possible to obtain such a list of properties by means of a training test collection. In this paper, we perform cross validation across the two different AOR test collections in order to identify the most promising properties.

On the other hand, the <owl:sameAs> object property is used to connect identifiers that refer to the same real world entity. For instance, the entity '<dbpedia:Barack_Obama>' links via <owl:sameAs> to the corresponding Freebase entity '<fbase:Barack_Obama>'. This object property is hence most probably valuable for improving the effectiveness of AOR in a LOD context.

To obtain a list of object properties worth following, we rank all properties in the dataset by their observed likelihood of leading to a relevant results[14]. Table 1 gives the top object property scores for the SemSearch collections. Using such a ranked list of properties as a reference, we define a series of result extension mechanisms exploiting structured queries over the data graph to identify new results. At this stage, we focus on recall rather than precision in order to preserve all candidate entities. Then, we rank candidate entities by means of a scoring function.

The first and simplest approach (SAMEAS) we consider is to follow exclusively the <owl:sameAs> links. Specifically, for each entity <e> in the top-N retrieved results, we issue the following query:

```
SELECT ?x
WHERE { { <e> <owl:sameAs> ?x }
        UNION
        { ?x <owl:sameAs> <e> }
      }
```

which returns all entities that are directly linked to <e> via a <owl:sameAs> object property.

Our second approach is based on the property scores defined above and exploits information from DBPedia. As DBPedia originates from Wikipedia, it contains disambiguation and redirect links to other entities. For example, 'dbpedia:Jaguar' links via <dbpedia:disambiguates> to 'dbpedia:Jaguar_Cars' as well as to 'dbpedia:Jaguar_(computer)' and others. Our second approach (S1_1) also follows such links to reach additional relevant entities. Thus, we define the following structured query:

```
SELECT ?x
WHERE { { <e> <owl:sameAs> ?x } UNION
        { ?x <owl:sameAs> <e> } UNION
        { <e> <dbpedia:disambiguates> ?x } UNION
        { ?x <dbpedia:redirect> <e> }
      }
```

which extends SAMEAS queries with two additional triple patterns.

Our third scope one approach (S1_2) includes properties that are more specific to the user queries. In addition to the generic properties of the previous two approaches, we add to the list of links to follow object properties like <dbpedia:artist>, <skos:subject>, <dbpedia:title>, and <foaf:homepage>, that appear in Table 1 and which lead to more general (albeit still related) entities.

The final scope one approach we propose (S1_3) extends S1_2 by adding matching patterns using the <skos:broader> property that links to more general entities. In addition to those four approaches, we additionally evaluated more complex query patterns based on the property scores defined above but did not obtain significant improvements over our simpler approaches.

[14]We compute the property scores by counting the number of times a property represents a path from a retrieved to a relevant entity (or the other way around) and dividing the result by the total number of such paths.

Structured Queries at Scope Two.

An obvious extension of the above approach is two look for related entities further in the graph by following object property links iteratively. In the following, we describe a few scope two approaches following pairs of links.

The number of potential paths to follow is increasing exponentially with the scope of the queries. To reduce the search space, we rank object property *pairs* by their likelihoods of leading to relevant entities[15].

The first strategy (S2_1) to retrieve potentially interesting entities at scope two is based on structured join queries containing top-two property pairs. The query issued to the structured repository looks as follows:

```
SELECT ?y
WHERE { { <e> <dbpedia:wikilink> ?x .
          ?x <skos:subject> ?y } UNION
        { <e> <dbpedia:wikilink> ?w .
          ?w <owl:sameAs> ?y } UNION
        [...]
      }
```

The second approach (S2_2) uses a selection of the top-two property pairs considering the starting entity e both as a subject as well as an object of the join queries. Finally, the third scope two approach (S2_3) applies a join query with the most frequent property pairs that do not include a <dbpedia:wikilink> property. The assumption is that due to the high frequency of this type of links[16] too many entities are retrieved, which produces a noisy result set. On the other hand, by focusing on non-frequent properties we aim at reaching few high-quality entities.

We note that it would be straightforward to generalize our graph traversal approach by considering scopes greater than two, or by considering transitive closures of object properties. Such approaches impose however a higher overhead and only marginally improve on scope one and scope two techniques from our experience.

6.2 Neighborhood Queries and Scoring

Once a new set of entities ($StructRetr$) has been reached, there is the need to 1) rank them by means of a scoring function and 2) merge the original ranking $Retr$ with $StructRetr$. Given an entity $e' \in StructRetr$ and the entity $e \in Retr$ from which e' originated, we compute the e' ranking score by defining a function $score(q, e, e')$ that is used to rank all entities in $StructRetr$.

Our scoring function exploits a text similarity metric applied to the query and the literals directly or indirectly attached to the entity by means of datatype properties. As noted above, related literals are implicitly clustered around entities in the LOD cloud. While many literals are directly attached to their entities (e.g., age, label), some are attached indirectly, either through RDF blank nodes (e.g., name-> firstname, address->zip code), or are attached to related entities.

[15]Due to space limitations, we do not report here the list of such property pairs which was computed in the same way as for scope one queries.

[16]As described at http://vmlion25.deri.ie/ this is the most frequent property in the test collection.

Table 1: Top object properties in the SemSearch collections by estimated likelihood (Recall) of connecting retrieved and relevant entities.

	From retrieved (x) to relevant entities (y)			From relevant (y) to retrieved entities (x)		
	Object Property	Recall	Prec	Object Property	Recall	Prec
SemSearch 2010	<http://dbpedia.org/property/wikilink>	82.10%	0.81%	<http://dbpedia.org/property/wikilink>	67.73%	0.52%
	<skos:subject>	11.75%	0.7%	<http://dbpedia.org/property/redirect>	7.47%	0.44%
	<http://www.w3.org/2002/07/owl#sameAs>	1.60%	1.54%	<http://www.w3.org/2002/07/owl#sameAs>	2.93%	0.76%
	<http://dbpedia.org/ontology/artist>	0.98%	15.42%	<http://dbpedia.org/property/disambiguates>	1.60%	3.85%
	<http://dbpedia.org/property/disambiguates>	0.68%	1.98%	<skos:subject>	1.33%	1.87%
	<http://dbpedia.org/property/title>	0.55%	1.81%	<http://xmlns.com/foaf/0.1/homepage>	1.33%	2.95%
	<http://dbpedia.org/ontology/producer>	0.43%	2.87%	<http://dbpedia.org/ontology/artist>	0.80%	1.97%
	<http://dbpedia.org/property/region>	0.43%	8.37%	...		
	<http://dbpedia.org/property/first>	0.37%	7.32%			
	<http://dbpedia.org/property/redirect>	0.25%	3.91%			
SemSearch 2011	<http://dbpedia.org/property/wikilink>	89.50%	0.44%	<http://dbpedia.org/property/wikilink>	37.50%	0.41%
	<skos:subject>	4.42%	0.22%	<skos:subject>	37.50%	0.32%
	<http://www.w3.org/2002/07/owl#sameAs>	1.66%	1.66%	<http://www.w3.org/2002/07/owl#sameAs>	10.71%	0.7%
	<http://dbpedia.org/property/disambiguates>	1.10%	0.89%	<skos:broader>	7.14%	0.93%
	<skos:broader>	1.10%	0.67%	<http://xmlns.com/foaf/0.1/homepage>	1.79%	2.08%
	<http://swat[...]/nsfaward.owl#piOrg>	1.10%	67.44%	<http://dbpedia.org/property/reference>	1.79%	1.75%
	<http://xmlns.com/foaf/0.1/page>	0.55%	0.13%	<http://rdfs.org/sioc/ns#links_to>	1.79%	0.04%
	...			<http://dbpedia.org/property/redirect>	1.79%	0.24%

We use neighborhood queries to retrieve all literals attached to a given entity through datatype properties, either at scope one or at scope two, e.g.,

```
SELECT ?S
WHERE { { <e> <datatypeproperty> ?S } UNION
        { <e> ?x ?y .
        ?y <datatypeproperty> ?S }
      }.
```

To minimize the overhead of such queries, we only retrieve the most promising literals. Table 2 lists the datatype properties whose values are most similar to the user queries. We adopt here the Jaro-Winkler (JW) similarity metric, which fits the problem of matching entity names well [9].

Additionally, we adopt a modified version of this scoring function by applying a threshold τ on the value of $JW(e', q)$ to filter entities that do not match well. Thus, only entities e' for which $JW(e', q) > \tau$ are included in the result set. Also, we check the number of outgoing links of each entity in *StructRetr* and only keep those entities having at least one outgoing link. The assumption (which was also made by the creators of the AOR evaluation collections) is that entities with no literals attached and that are not the subject of any statement are not relevant.

The final results are then constructed by linearly combining the Jaro-Winkler scores with the original entity score as noted above. The following section gives more information about this combination and compares the performance of several ranking methods.

7. EXPERIMENTAL EVALUATION

We present below the results of a performance evaluation of our hybrid approaches for AOR. Our aim was to evaluate the overall effectiveness and efficiency of our approach as well as the impact of the various parameters involved. More specifically, we investigated the impact of the following parameters:

- N, the number of top entities from the IR method which are used as a seed for the structured queries;
- *score*(), the scoring function used to rank entities from the structured repository;
- τ, the threshold on the text similarity measure to consider an entity description a match to the query;
- λ, the weight used to linearly combine the results from the IR and the structured query approaches.

In addition, we report results for different IR methods using BM25 scoring, structured IR indices using BM25F scoring, and graph traversals using scope one and two approaches over the structured repository combined with the IR baseline. We conclude this section with a few efficiency remarks on the overhead generated by our two-phase hybrid approach.

7.1 Experimental Setting

In order to evaluate and compare the proposed approaches, we adopt standard collections for the AOR task. We use the testsets created in the context of the SemSearch challenge for the 2010 and 2011 editions[17]. This allows us to compare the different variants we proposed with previous work that has been evaluated on the same collections.

The underlying dataset used in the testset is the Billion Triple Challenge 2009 dataset which consists of 1.3 billions RDF triples crawled from different domains of the LOD cloud. The two test collections contain 50 and 92 queries respectively, together with relevance judgements on a 3-level scale obtained by crowdsourcing the assessment task.

We adopt the official evaluation metrics from the SemSearch initiative and from previous work: Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), and early Precision (P10). Statistical significance is measured against the BM25 baseline by means of a two-tailed paired t-test by considering a difference significant when $p < 0.05$.

7.2 Evaluation Collections Based on Crowdsourcing

The AOR evaluation collections created in the context of the SemSearch initiative used crowdsourcing techniques to create relevance judgements for the entities by asking anony-

[17] `http://km.aifb.kit.edu/ws/semsearch10/` for 2010 and `http://km.aifb.kit.edu/ws/semsearch11/` for 2011.

Table 2: Top datatype properties with 10+ occurrences ranked by $JW(e', q)$ text similarity on the 2010 collection.

Datatype Property	JW(e',q)	Occurrences
<http://www.w3.org/2006/03/wn/wn20/schema/lexicalForm>	0.8449	19
<http://dbpedia.org/property/county>	0.8005	17
<http://www.daml.org/2003/02/fips55/fips-55-ont#name>	0.7674	27
<http://www.geonames.org/ontology#name>	0.7444	78
<http://www.aktors.org/ontology/portal#full-name>	0.7360	55
<http://dbpedia.org/property/wikiquoteProperty>	0.7096	10
<http://www.w3.org/2004/02/skos/core#prefLabel>	0.6911	158
<http://purl.org/dc/elements/1.1/title>	0.6711	236
<http://sw.opencyc.org/concept/Mx4rwLSVCpwpEbGdrcN5Y29ycA>	0.6680	48
<http://dbpedia.org/property/officialName>	0.6623	54

mous Web users to judge entity relevance for a small economic reward. Such a novel approach to relevance judgement triggers obvious questions about the reliability of the results. In [5], Blanco *et al.* experimentally show how such an approach is reliable and, most importantly, repeatable.

Traditional effectiveness evaluation methods (e.g., based on the Cranfield paradigm) have been largely used in TREC initiatives. Document collections, queries, and relevance judgements form test collections, which are typically made available to foster repeatable and comparable experiments. Because of the increasing size of the corpora and the impossibility of manually judging all documents, a pooling methodology is typically used: top retrieved results from each run submitted to the evaluation initiative are judged; non-judged results are assumed to be not relevant. Such an approach has been shown to be fair for comparing participants but unfair with respect to any following systems evaluated on the same collection "because the non-contributors will have highly ranked unjudged documents that are assumed to be not relevant" [22]. Zobel [25] showed that TREC collections can still be used to provide an unbiased comparison. To overcome the limitations of pooling, novel measures taking judgement incompleteness into account (e.g., bpref [8]) or adopting sampling to delve deeper into the ranked list (e.g., infAP [24]) have been proposed. Such refinements are also needed because the coverage of the judgement pools gets smaller over time as the size of the collection grows (e.g., 300,000 documents with a pool depth of 100 in TREC-5 and 1 billion documents with a pool depth of 20 in TREC 2011).

The recent trend of crowdsourcing relevance judgments enables an alternative approach. As shown in [5], the results obtained by crowdsourcing relevance judgements for the same collection at different points in time empirically demonstrates the repeatability of such experiments. Based on this important result and on the vision of "evaluation campaigns that run *continuously*" [5], we propose an approach to fair comparison of different IR systems running on the same evaluation collection.

Assuming systems A, B, and C participated in an evaluation initiative and, therefore, contributed to the assessment pool with their top-10 results, then each result of A in the top-10 will be judged while results beyond rank 10 may be judged or not (they will only be judge if retrieved by B or C in their top-10). On the other hand, a new system D appearing after the evaluation initiative may retrieve in its top-10 results many unjudged results that are typically considered as not relevant. This strongly penalizes runs which retrieve results that are very different from the original results that were part of the evaluation initiative. Instead, new systems should exploit crowdsourcing to obtain the missing judgements by running the same micro-tasks that provided the original (crowdsourced) relevance judgements.

One key advantage of this approach is that subsequent runs created after the initial evaluation campaign can be judged on a fair basis as if they were part of the original results. One drawback however is that the number of relevant results changes (i.e., increases) in this way, thus making measures that take this into account (e.g., MAP) incomparable with the ones computed originally by the evaluation initiative. Anyhow, the availability of the original retrieved results (i.e., the submitted runs which are available for all tasks at TREC) allows the authors of later approaches to recompute the measures and compare them against previous approaches.

As the approach proposed in this paper substantially differs from the approaches run during the evaluation campaign, both in terms of system architecture and retrieved results (see Table 3), we decided to adopt this new approach to obtain fair relevance judgements for the top-10 results of our various approaches. Thus, in the remainder of this paper, we report evaluation measures computed over relevance judgements complemented with additional crowdsourced judgements.

7.3 Completing Relevance Judgement through Crowdsourcing

In order to obtain additional relevance judgements for unjudged entities in the top-10 results of our runs, we published micro-tasks (HITs) on Amazon MTurk. We followed the same task design and setting as the ones used to create the test collections for the AOR task at SemSearch (as described in [5, 16]). We asked the crowd to judge a total of 1583 additional query-entity pairs for the 2010 collection and 1183 for the 2011 collection.

We split the tasks in batches of 10 entities per HIT, which were rewarded 0.20$ each and assigned to 3 different workers. The overall number of relevant entities increased by +8% (passing from 2028 to 2193) for the 2010 collection and by 17% (from 470 to 551) for the 2011 collection[18].

As a consequence, Precision@10 for our approach is comparable to the original SemSearch submissions as it is com-

[18]The extended assessment files we created together with the script to generate them from MTurk output are available at `http://diuf.unifr.ch/xi/HybridAOR`

Table 3: New retrieved results that were *not* retrieved by the BM25 baseline for top-10 results of IR and graph-based approaches in both collections.

	Approach	New Retr.	New Rel.	New Not-judged
SemSearch 2010	Extension	266	1	262
	Query Autoc.	308	2	303
	PRF3	220	0	218
	SAMEAS	92	11	76
	S1_1	211	29	162
	S1_2	312	19	281
	S1_3	312	19	281
	S2_1	15	3	12
	S2_2	16	3	13
	S2_3	16	3	13
SemSearch 2011	Extension	179	0	175
	Query Autoc.	216	2	212
	PRF3	153	0	152
	SAMEAS	20	8	12
	S1_1	56	14	39
	S1_2	103	9	97
	S1_3	103	9	92
	S2_1	2	1	1
	S2_2	2	1	1
	S2_3	4	4	0

puted on fully judged results. On the other hand, absolute values of MAP will be lower when computed with the extended relevance judgements as the number of relevant results has increased. In any case, the baseline we adopted closely matches the approaches used so far for the AOR task evaluated in SemSearch. Moreover, we compare below our proposed hybrid solution against a plethora of standard IR approaches like query extension and pseudo-relevance feedback.

7.4 Evaluation of Entity Ranking Techniques

Table 4 gives the results obtained by using BM25 scoring[19] on the inverted index built on all the literals attached to the entity (entity profile). We compare this baseline approach against standard IR techniques such as query extension using related words (Extension), query autocompletion as provided by commercial Web search engines (Query Autoc.), and Pseudo Relevance Feedback using top-3 retrieved entities (PRF3). We experimented with various approaches to handle the terms in the query and in the end opted for a disjunctive approach (i.e., we take each term in the query separately and 'OR' the results) since it performed best. The only exception is for the approaches based on a structured index and BM25F scoring for which we adopt a conjunctive approach. As we can see, the BM25 baseline performs best on both test collections. This indicates that methods working well for other entity search tasks may not be directly applicable to the AOR task due to the specific semantics of the user query, meant to uniquely describe one specific entity. Anyhow, we observe that the query autocompletion method obtains an Average Precision of 1.0 for a query (q16 of 2011), for which all other methods performed poorly (0.02). This originates from a misspelling in the original query, which was corrected by the external autocompletion functionality.

Table 5 gives results for the structured inverted index approach (see Section 5.1). The table lists results for indices built on entity profiles constructed using different types of literals. When compared to the baseline index (which aggregates all literals directly attached to the entities in a single *document*), we observe that structured indices perform better. Specifically, when a conjunctive query and BM25F ranking is used over a structured inverted index, effectiveness increases. The best results are obtained by the index encompassing all three fields, that is, URI, Label, and Attributes (ULA).

[19]The parameter b in BM25 has been selected by cross-validation on the 2010 and 2011 testsets.

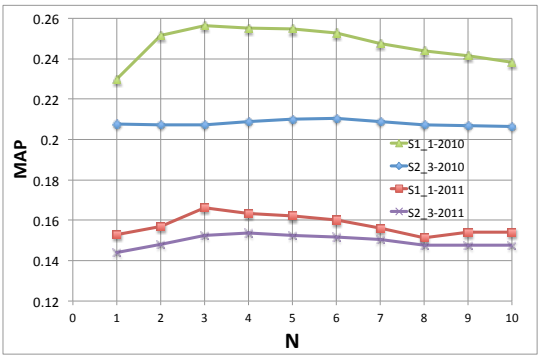

Figure 2: Effectiveness values varying the number of top-N entities retrieved by IR approaches.

7.5 Evaluation of Graph Query Techniques

Parameter Analysis.

Figure 2 shows how effectiveness (MAP) varies when varying the parameter N indicating the number of entities considered as input for the structured queries, both for the 2010 and 2011 collections. We observe that the best number of entities to consider lies between 3 and 4 in most cases, therefore we fix this parameter to 3 for the following experiments.

Moreover, we observe that high values (i.e., $0.8 - 0.9$) for the threshold τ used to discard entities lead to higher MAP in all cases. The optimal value for λ varies for scope one and two approaches. Interestingly, for the best performing method S1_1, the optimal value for λ is 0.5 for both the 2010 and 2011 collections: for such an approach, our hybrid system reaches an optimal tradeoff between IR and graph data management techniques.

Table 6 gives the performance of different functions used to compute $score(q, e, e')$. As we can see, the most effective method is the one that exploits the original BM25 score of e to score e'. This is the scoring function we use to compute the final results of the hybrid system as reported in Table 8.

Table 6: MAP values for S1_1 and S2_3 obtained by means of different instantiation of $score(q, e, e')$ using $\lambda = 0.5$ on the 2010 collection.

score(q,e,e')	S1_1	S2_3
Count $JW(e', q) > \tau$	0.2585	0.2076
Avg count $JW(e', q) > \tau$	0.2509	0.2075
Sum JW(e',q)	0.2558	0.2074
Avg JW(e',q)	0.2571	0.2075
Sum $BM25(e, q) - \epsilon$	0.2586	0.2113

Table 4: Standard ER approaches over the inverted index.

	2010 Collection			2011 Collection		
	MAP	P10	NDCG	MAP	P10	NDCG
BM25	**0.2070**	**0.3348**	**0.5920**	**0.1484**	**0.2020**	**0.4267**
Extension	0.1417	0.2152	0.4912	0.1131	0.16	0.3953
Query Autoc.	0.085	0.1717	0.4301	0.1298	0.1640	0.4136
PRF3	0.13	0.2152	0.5078	0.0998	0.1720	0.3642

Table 5: Approaches based on a structured inverted index with BM25 and BM25F scoring.

		2010 Collection			2011 Collection		
		MAP	P10	NDCG	MAP	P10	NDCG
BM25	URI only	0.2113	0.3120	0.5904	0.1141	0.1600	0.3562
	Label only	0.1156	0.2130	0.4911	0.1018	0.1660	0.3575
	Attrib. only	0.2070	**0.3348**	0.5920	0.1487	0.2020	0.4274
BM25F	URI-Label	0.2209	0.3239	**0.6083**	0.1502	0.2120	**0.4277**
	Label-Attrib.	0.2011	0.3033	0.5411	0.1431	0.1980	0.3966
	URI-Attrib.	0.2238	0.3130	0.5445	0.1538	0.2080	0.3988
	ULA	**0.2307**	0.3326	0.5450	**0.1628**	**0.2140**	0.3973

Table 7: Average query execution time (ms) on the 2011 dataset.

Approach	IR time	RDF time	Total time
BM25 Baseline	285	-	285
Extension	580	-	580 (+104%)
Query Autoc.	1447	-	1447 (+408%)
PRF3	2670	-	2670 (+837%)
SAMEAS	285	30	315 (+11%)
S1_1	285	48	333 (+17%)
S1_2	285	84	369 (+29%)
S1_3	285	86	371 (+30%)
S2_1	285	1746	2031 (+613%)
S2_2	285	2192	2477 (+769%)
S2_3	285	105	390 (+37%)

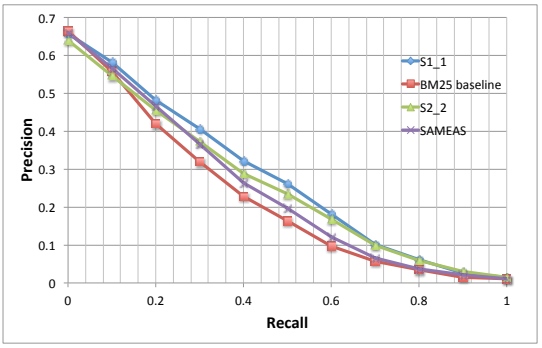

Figure 3: Precision/Recall curves for graph-based approaches on the 2010 collection.

Evaluation of Hybrid Approaches.

Table 8 gives results for the graph-based extension of the baseline ranking. We observe that most approaches based on scope one queries significantly improve over the baseline BM25 ranking. The simple S1_1 approach, which exploits `<owl:sameAs>` links plus Wikipedia redirect and disambiguation information, performs best obtaining a 25% improvement of MAP over the baseline on the 2010 dataset. Figure 3 shows Precision/Recall curves for the baseline BM25 ranking and for graph-based approaches. Again, we can see how S1_1 outperforms other approaches. The S1_2 approach performs best in terms of MAP on the 2011 dataset, although not significantly better than the baseline.

7.6 Efficiency Considerations

Table 7 shows execution times for the different components of our system and for various approaches[20]. Quite naturally, the IR baseline is faster than more complex approaches. Interestingly, we observe that structured approaches based on scope one queries perform very well, only adding a very limited overhead to the inverted index approach. The S1_1 approach, which is the best in terms of effectiveness, adds a cost of only 17% in terms of execution time to the BM25 baseline. The approaches based on scope two queries that use `<dbpedia:wikilink>` are more costly and would require a specific effort to make them scalable over large datasets. The scope two approach S2_6, which uses less frequent object properties, has still a reasonable overhead (37%).

In terms of storage consumption, the original collection containing 1.3 billions statements occupies 253GB. The baseline inverted index created with MG4J[21] is 8.9GB, the structured index used with BM25F scoring is 5GB, while the graph index created with RDF-3X[22] is 87GB. We note that the graph index could easily be optimized by discarding all the properties that are not used by the graph traversals and neighborhood queries (we could save considerable space this way, from 50% to 95% approximatively depending on the graph approaches used).

8. CONCLUSIONS

As more datasets become available on the Web, novel applications can exploit semi-structured data to build entity-centric functionalities. In this paper, we present a hybrid system for effectively and efficiently solving Ad-hoc Object Retrieval tasks. Our approach is based on the combination of results from an inverted index storing entity profiles and from a structured database storing graph data.

[20]We did not put any emphasis on improving the efficiency of our system. Experiments were run on a single machine with a cold cache and disk-resident indices for both the inverted indices and the structured repository.

[21]`http://mg4j.dsi.unimi.it/`

[22]`http://code.google.com/p/rdf3x/`

	2010 Collection			2011 Collection		
	MAP	P10	NDCG	MAP	P10	NDCG
BM25	0.2070	0.3348	0.5920	0.1484	0.2020	0.4267
SAMEAS	0.2293* (+11%)	0.363* (+8%)	0.5932 (+0%)	0.1612 (+9%)	**0.2200** (+9%)	0.4433 (+4%)
S1_1	**0.2586*** (+25%)	**0.3848*** (+15%)	0.5965 (+1%)	0.1657 (+12%)	0.2140 (+6%)	0.4426 (+4%)
S1_2	0.2305* (+11%)	0.3217 (-4%)	0.5724* (-3%)	**0.1731** (+17%)	0.2180 (+8%)	**0.4532** (+6%)
S1_3	0.2306* (+11%)	0.3217 (-4%)	0.5721* (-4%)	0.1716 (+16%)	0.2140 (+6%)	0.4501 (+5%)
S2_1	0.2118 (+2%)	0.3370 (+1%)	0.5971 (+1%)	0.1550 (+4%)	0.2060 (+2%)	0.4376 (+3%)
S2_2	0.2118 (+2%)	0.3370 (+1%)	0.5965 (+1%)	0.1555 (+5%)	0.2080 (+3%)	0.4379 (+3%)
S2_3	0.2113 (+2%)	0.3402 (+2%)	**0.5978** (+1%)	0.1589 (+7%)	0.2120 (+5%)	0.4385 (+3%)

Table 8: Graph-based approaches with scope one and two queries compared to the IR baseline. * indicates statistically significant difference against the BM25 baseline.

Our extensive experimental evaluation over two different evaluation collections shows that the use of structured search on top of standard IR approaches can lead to significantly better results (up to 25% improvement over the BM25 baseline in terms of Mean Average Precision). This comes at the cost of incorporating additional components in the system architecture and of implementing additional merging and ranking functions in the processing pipeline. In any case, our measurements show that the overhead caused by the graph data management components is surprisingly limited and only represents 17% for the most effective approach.

We consider our initial hybrid results as very promising, especially given that the LOD sample data sets used in the test collections were extremely noisy and incomplete (automated data cleaning and entity linking are two prominent research topics in the LOD community, which should hopefully alleviates those issues on the medium term). As future work, we plan to focus on further user queries (e.g, Entity List Completion queries and queries specifying the category of the entity sought) and on improving the efficiency of our proposed hybrid system in distributed and cloud settings.

9. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459, and by the Haslerstiftung in the context of the Smart World 11005 (Mem0r1es) project.

10. REFERENCES

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

[2] K. Balog, M. Bron, and M. De Rijke. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29:22:1–22:31, 2011.

[3] K. Balog, P. Serdyukov, and A. P. de Vries. Overview of the TREC 2011 entity track. In *TREC 2011*.

[4] K. Balog, I. Soboroff, P. Thomas, N. Craswell, A. P. de Vries, and P. Bailey. Overview of the TREC 2008 enterprise track. In *TREC 2008*.

[5] R. Blanco, H. Halpin, D. M. Herzig, P. Mika, J. Pound, H. S. Thompson, and D. T. Tran. Repeatable and reliable search system evaluation using crowdsourcing. In *SIGIR*, pages 923–932, 2011.

[6] R. Blanco, P. Mika, and S. Vigna. Effective and Efficient Entity Search in RDF Data. In *International Semantic Web Conference (1)*, pages 83–97, 2011.

[7] M. Bron, K. Balog, and M. de Rijke. Ranking related entities: components and analyses. In *CIKM*, pages 1079–1088, New York, NY, USA, 2010. ACM.

[8] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR*, pages 25–32, New York, NY, USA, 2004. ACM.

[9] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop On Data Cleaning And Object Consolidation*, 2003.

[10] R. Delbru, N. Toupikov, M. Catasta, and G. Tummarello. A Node Indexing Scheme for Web Entity Retrieval. In *ESWC (2)*, pages 240–256, 2010.

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

[12] G. Demartini, T. Iofciu, and A. P. de Vries. Overview of the INEX 2009 Entity Ranking Track. In *INEX*, pages 254–264, 2009.

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

[14] S. Elbassuoni and R. Blanco. Keyword search over RDF graphs. In *CIKM*, pages 237–242, 2011.

[15] C. Fellbaum. Wordnet. *Theory and Applications of Ontology: Computer Applications*, pages 231–243, 2010.

[16] H. Halpin, D. M. Herzig, P. Mika, R. Blanco, J. Pound, H. S. Thompson, and D. T. Tran. Evaluating Ad-Hoc Object Retrieval. In *International Workshop on Evaluation of Semantic Technologies (IWEST 2010) at ISWC2010*.

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

[18] E. Minack, R. Paiu, S. Costache, G. Demartini, J. Gaugaz, E. Ioannou, P.-A. Chirita, and W. Nejdl. Leveraging personal metadata for desktop search: The beagle^{++} system. *J. Web Sem.*, 8(1):37–54, 2010.

[19] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19:91–113, February 2010.

[20] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW*, pages 771–780, New York, NY, USA, 2010. ACM.

[21] S. E. Robertson and H. Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[22] E. M. Voorhees. The philosophy of information retrieval evaluation. In *CLEF'01*, pages 355–370.

[23] H. Wang, T. Tran, C. Liu, and L. Fu. Lightweight integration of IR and DB for scalable hybrid search with integrated ranking support. *Web Semant.*, 9:490–503, 2011.

[24] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *CIKM*, pages 102–111, 2006.

[25] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *SIGIR*, pages 307–314, New York, NY, USA, 1998. ACM.