One Query to Bind Them All

Daniel M. Herzig, Thanh Tran

Institute AIFB
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
{herzig, duc.tran}@kit.edu

Abstract. Recently, SPARQL became the standard language for querying RDF data on the Web. Like other formal query languages, it applies a Boolean-match semantics, i.e. results adhere strictly to the query. Thus, queries formulated for one dataset can not easily be reused for querying other datasets. If another *target dataset* is to be queried, the queries need to be rewritten using the vocabulary of the target dataset, while preserving the captured information need. This is a tedious manual task, which requires the knowledge of the target vocabulary and often relies on computational expensive techniques, such as mapping, data consolidation or reasoning methods. Given the scale as well as the dynamics of Web datasets, even automatic rewriting is often infeasible. In this paper, we elaborate on a novel approach, which allows to reuse existing SPARQL queries adhering to one dataset to search for entities in other dataset, which are neither linked nor otherwise integrated beforehand. We use the results returned by the given seed query, to construct an *entity relevance model* (ERM), which captures the content and the structure of relevant results. Candidate entities from the target dataset are obtained using existing keyword search techniques and subsequently ranked according to their similarity to the ERM. During this ranking task, we compute mappings between the structure of the ERM and of the candidates on-the-fly. The effectiveness of this approach is shown in experiments using large-scale datasets and compared to a keyword search baseline.

1 Introduction

Publishing structured data on the Web according to the Linked Data principles has been advocated by the Linked Data movement and the Semantic Web community and recently also receives an increasing support from companies like Yahoo!, Google, and Facebook, and also from governmental institutions. The amount of Linked Data on the Web increases rapidly and is now in the order of several billion RDF triples. In order to query these structured data effectively, SPARQL an expressive, formal query language has been developed, which became the standard as recommended by the W3C in 2008 [1]. However, like other formal query languages, e.g. SQL for relational databases, this query language applies a Boolean-match semantics, which means that results to a query are crisp and strictly adhere to the specified constraints in the query.

As a consequence, these queries are bound to specific datasets, respectively the corresponding vocabularies. Whereas the database community relies mainly on query rewriting [2], the answer of the Linked Data community to this problem is (1) interlinking datasets manually and with the help of mapping and consolidation techniques [3], (2) promoting the reuse of vocabularies and (3) exploiting the semantics of RDF for reasoning [4]. All these *data integration* approaches require high *upfront* investments, given the scale as well as the heterogeneity of the Web datasets, before the actual querying across datasets is possible. Also most of these integration steps need to be performed again, if the underlying data changes, which is frequently the case in the dynamic data Web scenario. Another way to bridge this schema- and data-mismatch is to apply information retrieval techniques, in particular keyword search, which crosses the gap of two datasets by simply ignoring the structure and applying the 'bag of words' notion not just on webpages, but also for data retrieval [5, 6].

In this paper, we elaborate on a novel approach, which allows to search for entities in a target dataset immediately without any prior integration using a star-shaped SPARQL query adhering to the vocabulary of another dataset. This approach combines the flexibility of unstructured information retrieval solutions with nature of database-style querying by incorporating the structure of the underlying data. The idea is to start with a structured seed query specified for one particular data source. Based on the content as well as the structure of results obtained from this data source, we construct an *Entity Relevance Model* (ERM) that can be seen as an abstract representation of relevant results. Instead of relying on mappings for rewriting the structured seed query, we simply treat the seed query as a keyword query and submit it against other data sources to obtain additional results. Then, we create mappings between the structure of candidate results and the structure of the ERM during runtime, which are used for the subsequent matching and ranking. Candidates matching more closely to the content as well as the structure captured by the ERM are ranked higher. This way, we can use unstructured keywords for querying, but at the same time, incorporate structure information into matching and ranking that are part of the search process.

We investigate the effectiveness of our approach in experiments using DBpedia as the source dataset and an RDF representation of IMdb as target dataset. Our approach is not limited to these datasets, but general enough to be applied to any datasets. However, the scenario involving DBpedia illustrates also an important use case, because DBpedia as the nucleus of the Web of data is probably the best known dataset and many users already have SPARQL queries or know how to create such queries for DBpedia. In addition many applications are build on top of DBpedia, which will also benefit, if the underlying retrieval incorporates entities from other datasets on-the-fly.

Contributions. The contributions of this work can be summarized as follows: (1) We propose a novel approach for searching Web data sources that does not rely on upfront data integration, but uses an *Entity Relevance Model* (ERM) for searching as well as for computing mappings on-the-fly in line with

the pay-as-you-go data integration [7] paradigm. (2) We provide one specific implementation of this approach, show how an ERM can be constructed and applied to rank search results exploiting mappings created during runtime. (3) In experiments, we investigate the feasibility and effectiveness of our approach and compare it to a common keyword search baseline.

Outline. Section 2 defines the research problem and gives an overview and briefly sketches our new approach. This approach of relevance based on-the-fly mappings is presented in detail in Section 3. Evaluation results are presented in Section 4. Section 5 discussed the related work before we conclude in Section 6.

2 Overview

In this section we define the setting of the addressed problem, present existing solutions, and briefly sketch the framework we propose to deal with this problem.

2.1 Data on the Web

The problem we address is situated in a Web data scenario. Therefore, the kind of Web data that is of most interest is RDF data. For reasons of simplicity and generality, we employ a generic graph-based data model that omits specific RDF features such as blank nodes. In this model, entity nodes are RDF resources, literal nodes correspond to RDF literals, attributes are RDF properties, and edges stand for RDF triples.

Data Graph. The data graph is a directed and labeled graph $G = (N, E)$. The set of nodes N is a disjoint union of entities N_E and literals N_L, i.e. $N = N_E \uplus N_L$. Edges E can be conceived as a disjoint union $E = E_E \uplus E_L$ of edges representing connections between entities, i.e. $a(e_1, e_2) \in E_E$, iff $e_1, e_2 \in N_E$, and connections between entities and literals, i.e. $a(e_1, v) \in E_L$, iff $e_1 \in N_E$ and $v \in N_L$. Given this graph structure, we define the set of edges including the targeted nodes $A(e) = \{a(e, \cdot) \in E\}$ as *the description* of the entity $e \in N_E$, and each member $a(e, \cdot) \in A(e)$ is called an attribute of e. Thus, we in fact neglect the distinction between E_E and E_L and simply refer to all edges as attributes. The set of distinct attribute labels of an entity e, i.e. $A'(e) = \{a | a \in A(e)\}$, is called the *model* of e.

2.2 Research Problem

Given this model of Web data, structured queries can be specified to search over such datasets. The most commonly used language for querying RDF data on the Web is SPARQL [1]. One essential feature of SPARQL is the Basic Graph Pattern (BGP), which conceptually resembles the notion of conjunctive queries that constitute an important fragment of other widely used structured query languages such as SQL. Basically, a BGP is a set of conjunctive triple patterns, each of the form $predicate(subject, object)$. They represent patterns because either *predicate*, *subject* or *object* might be a variable, or is specified explicitly

(as a constant). Answering these queries amounts to the task of graph pattern matching, where subgraphs in the data graph matching the query pattern are returned as results. Predicates are matched against attributes in the data, whereas bindings to subjects and objects in the query are entities and literal values, respectively.

One particular form of BGP with high importance are so-called *entity queries*. Essentially, they are star-shaped queries with the node in the center of the star representing the entity (entities) to be retrieved. Figure 3 provides several examples. According to a recent Web query logs study performed by Yahoo! researchers, this type of queries is most common on the Web [8] and also the analysis of SPARQL logs showed that it is also the most common type used in the Semantic Web context. Further, most of the current Semantic Web search engines such as Sig.ma [1] and Falcons [5] focus on answering these queries. For the sake of clarity, we also focus on this type of queries in this paper to illustrate the main ideas underlying our approach.

Problem. The problem is finding relevant entities in a set of *target datasets* D_t given a *source dataset* D_s and an entity query q_s adhering to the vocabulary of D_s. Clearly, if there are no syntactical differences at the level of schema and data as discussed above, then q_s can directly be used to retrieve information from D_t. However, given Web data are heterogeneous in that sense, the following two strategies can be applied.

Baseline KW. The first and most widespread solution to this end is to use keyword search over so called 'bag-of-words' representations of entities. That is, the description of an entity is simply a set of terms. A query is also a set of terms, which is then matched against the flat term-based representation of data. This unstructured approach bridges differences at the level of schemas and data simply by ignoring the fine-grained semantics and structure available in the entity descriptions, as illustrated in Fig. 1. Clearly, this approach is simple but also flexible in the sense that the same keyword query specified for D_s can also be used for D_t because results from D_t can be obtained when there are matches at the level of terms.

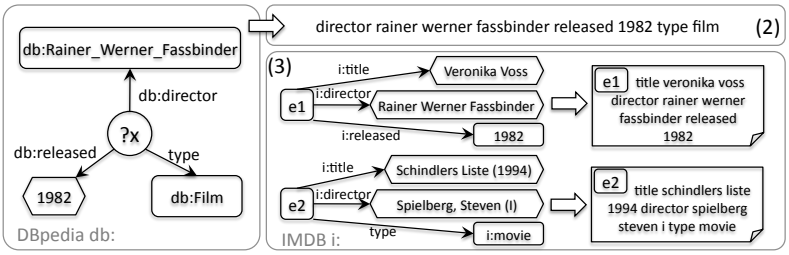


Fig. 1. Baseline: A structured query (1) transformed into a keyword query (2) and matched against *bag of word* representations of entities (3).

[1] `http://sig.ma/`

Our approach. In this paper, we present a framework to address this problem of searching for entities in Web data using on-the-fly mappings computed in a pay-as-you-go fashion based on entity relevance models (ERM). This framework is instantiated involving the following three steps. (1) First, we compute an ERM from the results returned from the source dataset D_s using q_s. (2) Second, we present a light-weight on-the-fly integration technique, which maps the structure of result candidates of the target datasets D_t to the structure of the ERM. (3) Finally, the result candidates are ranked according to their similarity to the ERM using the mappings computing during runtime.

3 Entity Search Over Web Data

In this section, we present how the entity relevance model is constructed and discuss how this model can be exploited for ranking and relevance-based on-the-fly data integration.

3.1 Entity Relevance Model

We aim to build a model that captures the structure and content of entities relevant to the information need, which is expressed in the seed entity query q_s. This proposed model is called the *Entity Relevance Model* (ERM). The model is based on the notion of structured relevance model that has been proposed before [9].

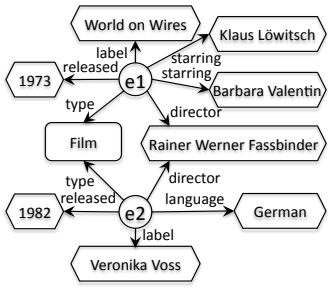
The ERM is a composite model that consists of several, atomic language models[10] and is constructed as follows. Let q_s be the query submitted against the source dataset D_s, and $R_s = \{e_1, \dots e_i, \dots e_m\}$ be the set of m entities obtained for this query. Across all m entities, we observe n distinct attribute labels, $a_s^1 \dots a_s^j \dots a_s^n$. For each observed attribute label a_s^j, we add a field a_s^j to the ERM, and create an attribute-specific language model $ERM^{a_s^j}$ to instantiate this field. This atomic model $ERM^{a_s^j}(w)$ specifies the probability of w occurring in the field a_s^j, for every word $w \in V$, where V is the vocabulary of all words. The $ERM^{a_s^j}(w)$ is computed from the entity descriptions $a_s^j \in A(e)$, $\forall e \in R_s$. In particular, every $ERM^{a_s^j}(w)$ captures the value v (i.e. the content) of the attributes with label a_s^j and is defined as follows:

$$ERM^{a_s^j}(w) = \frac{\sum_{e \in R_s} \sum_{v \in a_s^j} n(w, v)}{\sum_{e \in R_s} \sum_{v \in a_s^j} |v|} \tag{1}$$

where $n(w, v)$ denotes the number of occurrences of word w in the content value v of attribute a_s^j and $|v|$ the length of v. Note that the outer sum goes over the entities $e \in R_s$ returned by q_s and the inner sum goes over all values v of attributes with labels a_s^j. Thus, entity descriptions, which do not have the attribute a_s^j, do not contribute to $ERM^{a_s^j}(w)$. In order to capture the importance of the atomic attribute-specific models, we define the probability of occurrence $P(a_s^j)$ for every $ERM^{a_s^j}(w))$ as $P(a_s^j) = n(a_s^j, e_s^{1 \dots m})/m$, where the numerator denotes

the number of entities having attributes with label a_s^j and $m = |R_s|$ is the total number of entity results. In summary, an ERM has n fields, each corresponds to an attribute with a specific label, has a corresponding language model, which has a probability of occurrence. An example for an ERM constructed from two entities is illustrated in Fig. 2.

(1)



(2) ERM

a_s	$P(a_s)$	$w : ERM^{a_s}(w)$
label	1	world:0.2, on:0.2, wires:0.2, ...
starring	0.5	klaus:0.25, löwitsch:0.25, barbara:...
director	1	rainer:0.33, werner:0.33, fassbinder:0.33
released	1	1973:0.5, 1982:0.5
language	0.5	english:1
type	1	film:1

(3) e_t

a_t	$w : e_t^{a_t}(w)$
i:title	e:0.33, t:0.33, 1994:0.33
i:actors	coyote:0.5, peter:0.5
i:directors	spielberg:0.33, steven:0.33, i:0.33
i:producer	spielberg:0.33, steven:0.33, i:0.33
type	movie:1

Fig. 2. Example ERM (2) build from two entities e1 and e2 as illustrated in (1). The ERM has a field for each attribute with label a_s. Each field is weighted with $P(a_s)$ and has a relevance model $ERM(w)^{a_s}$ defining the probability of w occurring in this field. (3) Representation of an entity e_t with language models for each attribute with label a_t.

3.2 Searching over Web Data Using ERM

We tackle the problem of searching over heterogeneous data in a way similar to entity consolidation. That is, given the results $e_s \in R_s$ from the source dataset obtained for the seed query, we aim to find out which entities in the target datasets are similar to R_s. We use the ERM as the model of those relevant results. In particular, we estimate which entities e_t of D_t are relevant for the query q_s by measuring their similarity to the ERM and rank them by decreasing similarity. First, we capture the values of a_t^k of each entity e_t using a language model $e_t^{a_t^k}(w)$. Similar to the ERM, this model is defined based on the number of word occurrences as follows:

$$e_t^{a_t^k}(w) = \frac{\sum_{v \in a_t^k} n(w, v)}{\sum_{v \in a_t^k} |v|} \tag{2}$$

As before, the sum goes over all values of attributes with label a_t^k, $n(w, v)$ denotes that number of occurrences of w in v, and $|v|$ denotes the length of v. Fig. 2 (3) illustrates an example.

The similarity of a candidate entity e_t from the target dataset D_t to the ERM is calculated as the sum of the weighted similarities between the language

model of field a_s^j and the language model of a_t^k, which is calculated by their cross entropy H:

$$Sim(ERM, e_t) = \sum_{a_s^j} \beta_{a_s^j} \cdot P(a_s^j) \cdot H(ERM^{a_s^j} || e_t^{a_t^k}) \quad (3)$$

In particular, we apply this similarity calculation only when we know which field a_s^j of ERM should be matched against which attribute a_t^k of e_t. We address this problem in the next section and show how the ERM can be exploited to create on-the-fly schema mappings, i.e. mappings between an attribute a_t^k and a field a_s^j of ERM. Equation 3 applies to corresponding pairs of attribute and field. If there is no mapping between a_s^j and a_t^k, then we use a "maximum distance". This distance is computed as the cross entropy between $ERM^{a_s^j}$ and a language model that contains all words in the vocabulary expect the ones of $ERM^{a_s^j}$.

The similarity for each field a_s^j is weighted by the occurrence probability of this field, $P(a_s^j)$, and the parameter $\beta_{a_s^j}$. This parameter gives us the flexibility to boost the importance of attributes that occur in the query q_s as follows:

$$\beta_{a_s^j} = \begin{cases} 1 & \text{if } a_s^j \notin q_s \\ b & \text{if } a_s^j \in q_s, b > 1 \end{cases} \quad (4)$$

For constructing the language models of the ERM and of the candidate entities, a maximum likelihood estimation has been used, which is proportional to the count of the words in an attribute value. However, such an estimation assigns zero probabilities to those words not occurring in the attribute value. In order to address this issue, $e_t^{a_t^k}(w)$ is smoothed using a collection-wide model $c_s(w)$, which captures the probability of w occurring in the entire dataset D_s. This smoothing is controlled by the Jelinek-Mercer parameter λ. As a result, the entropy H is calculated over the vocabulary V of field a_s^j as:

$$H(ERM^{a_s^j} || e_t^{a_t^k}) = \sum_{w \in V_{a_s^j}} ERM^{a_s^j}(w) \cdot \log(\ \lambda \cdot e_t^{a_t^k}(w) + (1-\lambda) \cdot c_s(w)\ ) \quad (5)$$

3.3 On The Fly Integration Using ERM

We want to determine which attribute of an entity needs to be compared to a given field of the ERM constructed for D_s. The ERM is not only used for search, but also exploited for this alignment task. The details for computing mappings between entity attributes $e_t^{a_t^{1\ldots r}}$ and ERM fields $ERM^{a_s^{1\ldots n}}$ are presented in Algorithm (1).The rational of the algorithm is that a field a_s^j is aligned to an attribute a_t^k when the cross entropy $H(ERM^{a_s^j} || e_t^{a_t^k})$ between their language models is low, i.e. a mapping is established, if H is lower than a threshold t (normalized based on the highest cross entropy, line 12).That is, entropy is used as a similarity measure and the features used to compare attributes are based on their entire contents represented using language models. The algorithm iterates

over $n \cdot r$ comparisons in worst case for an ERM with n fields and an entity with $r = |A'(e_t)|$ attribute labels. Note that this on-the-fly algorithm operates only on entities that are requested as part of the search process, i.e. n and r are rather small, compared to full-fledge upfront integration that takes the entire schema into account, see Table 1 and Table 3. Especially, this computation comes for free: searching also requires the same computation (Equation 3) and thus the entropy values computed here are kept and subsequently reused for that. Moreover, for a faster performance, ERM fields having an occurrence probability of $P(a_s^j) < c$ can be pruned, because of their negligible influence. In addition, existing mappings can be used to reduce the number of comparisons even further.

Algorithm 1 On the fly Alignment

Input: $ERM^{a_s^{1\dots n}}$, Entity $e_t^{a_t^{1\dots r}}$, Threshold $t \in [0,1]$
Output: $Mappings\ A := \{(a_s^j, a_t^k) | a_s^j \in ERM, a_t^k \in A'(e_t) \cup null\}$

```
A := new Map
for all a_s^j ∈ ERM do
    candMappings := new OrderedByValueMap
    for all a_t^k ∈ A'(e_t) do
        if a_t^k ∉ A.values then                          // If not already aligned
            h ← H(ERM^{a_s^j} || e_t^{a_t^k})             // see equation (5)
            candMappings.add(a_k^t, h)
        end if
    end for
    bestA ← candMappings.firstValue
    worstA ← candMappings.lastValue
    if bestA < t · worstA then
        a_t^i ← candMappings.firstKey
        A.add(a_s^j, a_t^i)
    else
        A.add(a_s^j, null)                                // no mapping found
    end if
end for
return A
```

4 Experiments

In this section, we report on the experiments conducted using an implementation of the proposed approach. We performed the experiments with the following parameter setting: $\beta = 10$, $c = 0.8$, $t = 0.75$ and follow the *Cranfield*[11] methodology for the experiments on the search performance.

4.1 Web Datasets

Our experiments were conducted with two RDF Web datasets, *DBpedia 3.5.1* and *IMdb*. DBpedia is a structured representation of Wikipedia which contains more than 9 million entities of various types, among them about 50k entities typed as films. The IMdb dataset[2] is derived from www.imdb.com and transformed into RDF. The IMdb dataset contains information on movies and films.

[2] We thank Julien Gaugaz and the L3S Research Center for providing us their version of the IMdb dataset.

Table 1 gives details about each dataset and shows also the varying average size of entity descriptions $|A(e)|$.

Dataset	#Entities	#Distinct Attribute Labels	$\|A(e)\|$ ±StdDev.
DBpedia	9.1M	39.6K	9±18.2
IMdb	859K	32	11.4±6.4

Table 1. dataset statistics

Rel. Entities	IMdb	DBpedia
max	834	47
avg.	114.9	10.9
median	21	5
min	1	1

Table 2. Relevant Entities per Query for each dataset

Source Dataset	$\|ERM\|$ ± StdDev.
IMdb	15.8±6.7
DBpedia	23±5.4

Table 3. Average Number of Fields of an ERM.

4.2 Queries and Ground Truth

Our goal is to find relevant entities in the target datasets D_t for a given query q_s. We determine the relevant entities in D_t by manually rewriting the query q_s to obtain a structured query q_t adhering to the vocabulary of D_t. Fig. 3 shows such a set of queries, one of the queries serves as the source query q_s and the results of the other two queries capture the ground truth for the IR-experiments.

Note that this ground truth is conservative in the sense that it considers only matches that can be obtained through query rewriting based on same-as mappings. However, some possibly relevant entities may either be represented differently (using attributes that cannot be aligned via same-as mappings) or do not contain information for some of the query predicates. Hence, this ground truth defines a lower bound on the actual performance.

We created two query sets, each containing 23 BGP entity queries of different complexities, ranging from 2 to 4 triple patterns and yielded a varying number of relevant entities (see Table 2). The structured queries represent real-world information needs, e.g. retrieve "all movies directed by Steven Spielberg", "all movies available in English and also in Hebrew", or "all movies directed by Rainer Werner Fassbinder, which were released in 1982". The last query is illustrated in Fig. 3.

4.3 Keyword Query Baseline (KW)

We compare the performance of our ERM approach against Semplore [6], which is based on Lucene, a real-world keyword search system. Semplore creates a virtual document for every entity description and use the concatenations of attribute and attribute value as document terms. In the same way, we transform the structured query into a keyword query of disjunctive terms by using the concatenations of predicates and constants of the structured query as keywords. The resulting keyword query retrieves all virtual documents representing entity

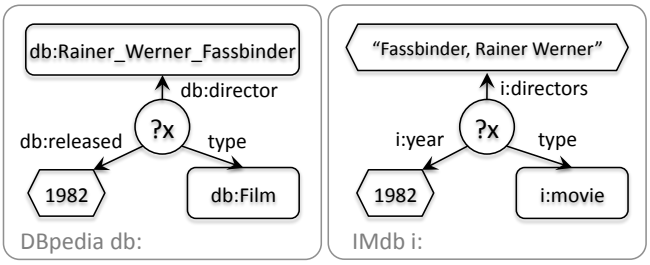


Fig. 3. Example of manually created queries as ground truth. Each query represents the same information need for one of the datasets.

descriptions, which contain some of the corresponding terms. Fig. 1 illustrates an example.

4.4 Search Performance

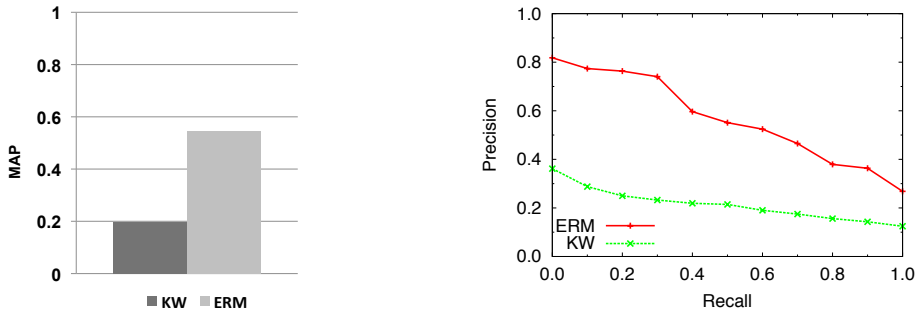
We evaluate search performance using the standard IR measures precision, recall, mean average precision (MAP) and mean reciprocal rank (MRR). We retrieve the top five thousand entities, rank them, and compute the metrics based on the top one thousand entities returned by each method. We evaluate ERM against the baseline described above. ERM performs alignment on-the-fly and operates without any upfront integration or prior knowledge about the schemas. When comparing the performance of ERM and KW in Fig. 4(a), we observe that ERM outperforms KW. ERM yields a MAP of 0.55, whereas KW achieves a MAP of about 0.2.

The robustness of the retrieval performance of ERM can be observed in Fig. 4(b), which shows the interpolated precision across the recall levels. We can observe that ERM considerably outperforms the KW baseline across the recall-levels.

5 Related Work

We have discussed related work throughout the paper. Basically, there are two existing lines of approaches, one that is based on keyword search and the other one is structured query rewriting. Our approach represent a novel combination which combines the flexibility of keyword search with the power of query rewriting. Just like keyword search, it does not rely on precomputed mappings. However, it is able to exploit the fine-grained structure of query and results, which is the primary advantage of structured query rewriting. In addition, it can leverage existing mappings created by tools like [12, 13].

More precisely, there exist similar keyword search retrieval models that also take the structure of the underlying data into account. In fact, the model underlying our approach originates from the concept of language models [10], which



(a) Mean Average Precision (b) Precision-Recall Curve

Fig. 4. IR Performance of ERM compared to KW Baseline.

have been proposed for modeling resources and queries as multinomial distributions over the vocabulary terms. More precisely, the foundation of our work is established by Lavrenko et al., who propose relevance-based language models to directly capture the relevance behind document and queries.In this context, structure information has been exploited for constructing structured relevance models [9] (SRM). This is the one mostly related to ERM. The difference is that while the goal of SRM is to predict values of empty fields in a single dataset scenario, ERM targets searching in a completely different setting involving multiple heterogeneous datasets.

Beside the query rewriting [2] strategy, there are database oriented approaches bridging the schema- and vocabulary gap by query relaxation [14, 15]. However, these approaches need also a precomputing step to either identify duplicates [14] or derive alternative relations[14, 15].

The proposed mapping technique is in principle similar to existing techniques, e.g. [16], to the extent that it relies on the same features, i.e., values of attributes. However, the use of language models for representing these features as well as the similarity calculation based on entropy is common for retrieval tasks, but has not been applied to the schema mapping problem before. We consider this as a promising approach for embedding the pay-as-you-go data integration paradigm [7] into the search process.

6 Conclusion

We have proposed a novel approach for searching Web datasets using one single structured seed query that adheres to the vocabulary of just one of the dataset. It is based on using an entity relevance model computed from the seed query, which captures the structure and content of relevant results. This model is used both for matching and ranking relevant results, as well as for performing data integration on-the-fly. This approach combines the flexibility of keyword search in the sense that no upfront integration is required, with the power of structured

query rewriting techniques that comes from the use of the fined-grained structure of query and results. The experiments showed that our approach outperform a common keyword search baseline.

We demonstrated our approach between one source and one target dataset. However, there is no inherent limitation to the number of target datasets. So theoretically, one query can be used 'to bind them all'.

7 Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen project (grant 01IA08005K).

References

1. W3C: Recommendation 15 January 2008, SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/.
2. Calì, A., Lembo, D., Rosati, R.: Query rewriting and answering under constraints in data integration systems. In: IJCAI. (2003) 16–21
3. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. SIGMOD Rec. **35** (September 2006) 34–41
4. Bonatti, P.A., Hogan, A., Polleres, A., Sauro, L.: Robust and scalable linked data reasoning incorporating provenance and trust annotations. J. Web Sem. **9**(2) (2011) 165–201
5. Cheng, G., Qu, Y.: Searching linked objects with falcons: Approach, implementation and evaluation. Int. J. Semantic Web Inf. Syst. **5**(3) (2009) 49–70
6. Wang, H., Liu, Q., Penin, T., Fu, L., 0007, L.Z., Tran, T., Yu, Y., Pan, Y.: Semplore: A scalable ir approach to search the web of data. J. Web Sem. **7**(3) (2009) 177–188
7. Madhavan, J., Cohen, S., Dong, X.L., Halevy, A.Y., Jeffery, S.R., Ko, D., Yu, C.: Web-scale data integration: You can afford to pay as you go. In: CIDR. (2007) 342–350
8. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Proc. of the 19th Int. Conf. on WWW. (2010) 771–780
9. Lavrenko, V., Yi, X., Allan, J.: Information retrieval on empty fields. In: HLT-NAACL. (2007) 89–96
10. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR. (1998) 275–281
11. Cleverdon, C.: The CRANFIELD Tests on Index Langauge Devices. Aslib (1967)
12. Hu, W., Qu, Y.: Falcon-ao: A practical ontology matching system. J. Web Sem. **6**(3) (2008) 237–239
13. David, J.: Aroma results for oaei 2009. In: Ontology Matching Workshop, ISWC. (2009)
14. Zhou, X., Gaugaz, J., Balke, W.T., Nejdl, W.: Query relaxation using malleable schemas. In: SIGMOD Conference. (2007) 545–556
15. Elbassuoni, S., Ramanath, M., Weikum, G.: Query relaxation for entity-relationship search. In: ESWC. (2011) 62–76
16. Duan, S., Fokoue, A., Srinivas, K.: One size does not fit all: Customizing ontology alignment using user feedback. In: International Semantic Web Conference. (2010) 177–192