Heterogeneous Web Data Search Using Relevance-based On The Fly Data Integration

Daniel M. Herzig
Institute AIFB
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
herzig@kit.edu

Thanh Tran
Institute AIFB
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
duc.tran@kit.edu

ABSTRACT

Searching over heterogeneous structured data on the Web is challenging due to *vocabulary and structure mismatches* among different data sources. In this paper, we study two existing strategies and present a new approach to integrate additional data sources into the search process. The first strategy relies on data integration to mediate mismatches through upfront computation of mappings, based on which queries are rewritten to fit individual sources. The other extreme is keyword search, which does not require any upfront investment, but ignores structure information. Building on these strategies, we present a hybrid approach, which combines the advantages of both. Our approach does not require any upfront data integration, but also leverages the fine grained structure of the underlying data. For a structured query adhering to the vocabulary of just one source, the so-called seed query, we construct an entity relevance model (ERM), which captures the content and the structure of the seed query results. This ERM is then aligned on the fly with keyword search results retrieved from other sources and also used to rank these results. The outcome of our experiments using large-scale real-world data sets suggests that data integration leads to higher search effectiveness compared to keyword search and that our new hybrid approach consistently exceeds both strategies.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Relevance feedback*

General Terms

Experimentation

Keywords

vertical search, structured web data, data integration, rdf

1. INTRODUCTION

A rapidly increasing amount of structured data can be found on the Web today. This development is triggered by the Linked Data movement, Semantic Web community efforts, and recently, also enjoys strong support from large companies including Google, Yahoo! and Facebook, and governmental institutions. The amount of Linked Data alone is in the order of billions of RDF triples, residing in hundreds of data sources [11]. In this paper, we aim at supporting the exploitation of these structured Web data. In particular, we aim at extending vertical search capabilities beyond internal data to also incorporate external Web data into the retrieval process. We illustrate the problem behind it based on the following scenario:

There is a company running a movie shopping website. Users can search for movies on this website via form-based interfaces, and their requests are internally executed as structured queries against the company's dataset. Now, the company aims to exploit the numerous Web data sources available as Linked Data, including data provided by a partner company with similar offerings and an encyclopedia dataset that contains additional movie related information. The goal is to incorporate data from these external sources into the search processes. However, the vocabularies and structure exhibited by these target data sources are different such that issuing the same structured queries (called seed queries) against these external sources may not produce any results. Results satisfying the information needs behind these seed queries may exist but due to mismatches in structural and syntactical representation, they cannot be found.

In this paper, we study three different strategies that are applicable to this search scenario:

(1) There are Information Retrieval (IR) solutions, which treat both the data and queries as bags of words [5, 20]. Because structure information is ignored during query processing, this strategy (called *keyword search*) often leads to non-empty results – albeit with varying quality.

(2) The alternative is to employ database solutions, where information needs are expressed as structured queries. Given the richer representation of the information needs, the structure of the underlying data can be exploited and incorporated into the matching process. While this can improve the quality of the results, this type of solutions requires upfront investment in *data integration*, i.e. computation of ontology and schema mappings and consolidation of data instances that refer to the same object (entity mappings) [8, 9, 13, 4]. Based on these mappings, results from external sources can be obtained via query rewriting [3, 23]. Integration efforts are needed whenever the data changes. Clearly, integration on the Web is hard due to the large number of sources and their scale as well as their heterogeneity regarding differences at the schema and data level, which is illustrated for our scenario in Figure 1. Here, entities representing movies are displayed. One can observe that three different representations of "Steven Spielberg" are used for the same real-world object. Also, different labels are used to express the same attribute.

(3) As the third category, we elaborate on a hybrid solution, which combines the flexibility of unstructured IR solu-

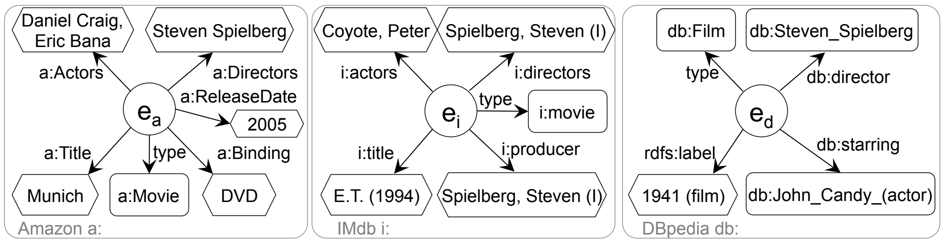


Figure 1: Data heterogeneity on the Web. Entities from three different Web datasets are represented differently at the schema level (e.g. *actors* vs. *starring*) and data level (e.g. *Spielberg, Steven* vs. *Steven Spielberg*).

tions (in the sense that no prior data integration is needed) and the expressiveness 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 source. Based on the content and structure of the results obtained from this source, we construct an *Entity Relevance Model* (ERM) that can be seen as a compact representation of relevant results mirroring the underlying information need. Instead of relying on up-front computed mappings for rewriting the structured seed query, we treat the seed query as a keyword query and submit it against external data sources to obtain additional results. These candidates are obtained using a standard IR-based search engine. Then, we create mappings between the structure of each candidate result and the structure of the ERM on the fly. These mappings are used for an additional round of matching and ranking. Candidates which more closely match the content as well as the structure captured by the ERM are ranked higher. Thereby the structure of the ERM and of the result candidates are incorporated into the search process. Since, the same similarity metrics for creating the mappings are reused for ranking, this on the fly integration comes for free. As a result, this hybrid strategy not only takes structure information into account for more effective search, but also provides on the fly computed mappings that can support a pay-as-you-go integration paradigm where data integration is tightly embedded into the search process [16].

Contributions. The contributions of this work can be summarized as follows: (1) We perform a systematic study of the two main prevailing strategies towards searching external heterogeneous data sources. In particular, we show how to adopt the data integration approach to our scenario where the computation of entity mappings is challenging. (2) To achieve the best of both worlds, we elaborate an a hybrid approach that does not rely on upfront data integration, but uses a query-specific *Entity Relevance Model* (ERM) for searching as well as for computing mappings on the fly. (3) Based on large-scale experiments using real-world datasets, we observe that the data integration approach consistently provides better results than keyword search. The hybrid approach yields best results, outperforming keyword search by 120% and the data integration baseline by 54% on average in terms of Mean Average Precision. Further, this hybrid approach is able to leverage upfront integration results, leading to additional quality improvement when precomputed mappings are considered. The qualitative differences between these approaches are: Keyword search and the hybrid approach do not require upfront data integration. Additionally, the hybrid approach provides on the fly computed mappings that can be used for a pay-as-you-go integration process that can exploit user feedbacks for quality improvement (as discussed in [16]).

Outline. Section 2 defines the research problem and gives an overview of existing solutions 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 discusses the related work before we conclude in Section 6.

2. OVERVIEW

In this section, we present the setting of the addressed problem, and provide an overview of three different solutions.

2.1 Data Heterogeneity on the Web

The problem we address is situated in a Web data scenario. The kind of Web data that is of most interest is RDF data. For reasons of generality and simplicity, 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 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. $a(e_i, e_j) \in E_E$, iff $e_i, e_j \in N_E$, and connections between entities and literals, i.e. $a(e_i, e_j) \in E_L$, iff $e_i \in N_E$ and $e_j \in N_L$. Given this graph, we call the set of edges $A(e_i) = \{a(e_i, e_j) \in E\}$ *the description* of the entity $e_i \in N_E$, and each member $a(e_i, e_j) \in A(e_i)$ is called an attribute of e_i. The set of distinct attribute labels of an entity e_i, i.e. $A'(e_i) = \{a | a(e_i, e_j) \in A(e_i)\}$, is called the *model* of e_i.

It is clear that this notion of data graph is sufficiently general to capture not only RDF but also other types of Web data. For instance, data in a relational database can be mapped to this model by representing tuple ids as entity nodes, other tuple values are literal nodes that are connected to the corresponding ids that are in the same tuple, and foreign key relationships are captured as connections between entity nodes.

Data Heterogeneity. Web data reside in different datasets, each represented by a data graph. Typically, real-world Web datasets exhibit *heterogeneity* at the schema and the data level. At the data level, entities in different datasets, which refer to the same real-world object, may have different descriptions. Differences at the schema level occur when the same entity is represented in different datasets using attributes with different labels (different models). As mentioned in the introduction, Figure 1 exemplifies this heterogeneity exhibited by real-world datasets. Dealing with these types of heterogeneity requires data integration. For this, a large body of work on schema alignment and entity consolidation (record linkage) can be leveraged to compute mappings between data sources [8]. While mappings of varying semantics have been proposed, the most basic and commonly used one asserts that two elements (schema elements or entities) are the same (i.e. *same-as mappings*).

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). 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 explicitly specified 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 edges in the data graph, whereas bindings to subjects and objects in the query are entity or literal nodes.

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 6 provides 3 examples. According to a recent Web query logs study performed by Yahoo! researchers, queries searching for entities constitute the most common type on the Web [18]. Also, 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. Later, we will point out how our approach can be extended towards supporting general graph patterns. This however requires more complex algorithms for searching paths between entity nodes matching the query keywords (i.e. matching the seed query represented as keywords).

Problem. Based on her knowledge about the schema and data of one particular source (e.g. the one owned by the company in our scenario), it is possible for a programmer or expert user to specify complex entity queries that specifically ask for information from this source. It is however not trivial to exploit external datasets for this kind of entity search when they exhibit heterogeneity at the schema and data level as discussed before. The problem we tackle is finding relevant entities in a set of *target datasets* $\mathbb{G}_t$ given a *source dataset* G_s and an entity query q_s adhering to the vocabulary of G_s.

2.3 Solutions

Clearly, if all datasets exhibit the same schema and data representation, then q_s can directly be used to retrieve information from $\mathbb{G}_t$. When this is not the case, the following different solutions can be applied.

Keyword Search (KW). The first and most widespread solution to this end is to use keyword search over so called 'bag-of-words' representations of entities [20, 5]. That is, the description of an entity is simply a bag of terms. A query is also represented as terms, which is then matched against the term-based representation of the entities. Clearly, this

[1]http://sig.ma/

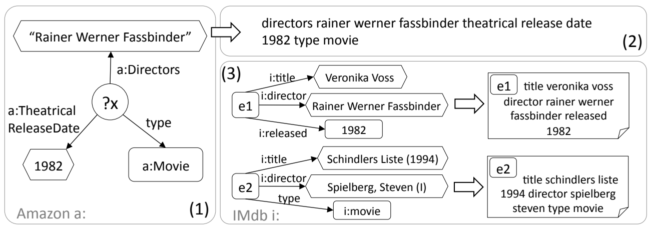

"Rainer Werner Fassbinder"
a:Directors
?x
a:Theatrical ReleaseDate
type
1982
a:Movie
Amazon a:
(1)
directors rainer werner fassbinder theatrical release date 1982 type movie
(2)
(3)
i:title
Veronika Voss
i:director
Rainer Werner Fassbinder
e1
i:released
1982
e1 title veronika voss director rainer werner fassbinder released 1982
i:title
Schindlers Liste (1994)
i:director
Spielberg, Steven (I)
e2
type
i:movie
e2 title schindlers liste 1994 director spielberg steven type movie
IMdb i:

Figure 2: KW: A structured query (1) transformed into a keyword query (2) and matched against *bag of words* representations of entities (3).

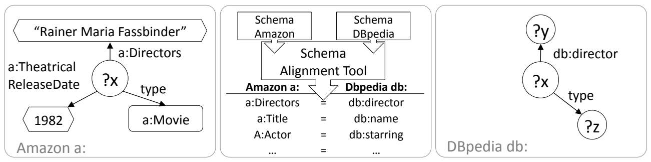


Figure 3: QR: A query for Amazon is rewritten into a query for DBpedia with constants being replaced with variables, and the missing mapping results in an "empty" triple pattern.

approach is simple but also flexible in the sense that the same keyword query specified for G_s can also be used for $\mathbb{G}_t$ because results from $\mathbb{G}_t$ can be obtained when there are matches at the level of terms. As illustrated in Figure 2, this approach ignores structure information and vocabulary mismatches.

Structured Query Rewriting (QR). Another view on this retrieval problem is the database perspective. Here, structure information in the entity descriptions is taken into account. However, this also requires the query to be fully structured. The strategy to query over multiple datasets and to deal with data heterogeneity here is to rewrite the structured seed query q_s into a query q_t that adheres to the vocabulary of the target dataset $G_t \in \mathbb{G}_t$. For this, same-as mappings are computed using entity consolidation and schema mapping tools [8, 9, 13, 4]. Then, predicates and constants in q_s referring to attributes and entities in G_s are replaced with predicates and constants representing corresponding attributes and entities in G_t. While this strategy can exploit the fine grained structure of data and query, it relies on upfront data integration, which is problematic in the Web scenario because Web datasets are heterogeneous and evolve quickly. In our experiment on the datasets prepared for the Billion Triple Challenge[2] for instance, we observe that state-of-the-art entity consolidation approaches [4] do not scale well to large datasets [21]. In particular, they are focused on the single-domain setting such that for these heterogeneous datasets (where many of them exhibit only small pairwise overlaps at the schema level), only a relatively small amount of correct mappings could be produced. Thus, rewriting constants using entity mappings is especially challenging in this scenario.

In fact, it has been recognized that integration at the Web scale is too complex and resource-intensive to be performed completely upfront [16]. A more practical strategy to deal with this dynamic and large-scale environment is to perform integration as you go [16], i.e. at usage time as the system evolves. In this regard, an alternative solution is to precompute schema mappings only. Then, entity mappings

[2]http://challenge.semanticweb.org/

that are needed for a specific query are obtained at runtime. Figure 3 illustrate this: Schema mappings are used to rewrite the query, triple patterns for which no corresponding schema-level mappings exist are omitted, and constants are replaced with variables (instead of being replaced with constants that adhere to the vocabulary of the target source). The resulting query captures only structure constraints of the original query and thus, produces possibly much more results than a query where constants are also rewritten. To achieve that, a standard IR search engine can be leveraged to limit the results to only those, which match the constants expressed as keyword queries. That is, the constants that have been replaced by variables in the first step, act as a keyword query in the second step to perform on the fly entity consolidation, i.e. to find entities in G_t, which match the entities in G_s as represented by the constants (such as "Rainer Maria Fassbinder 1982" in the example).

Our approach. In this paper, we present a framework to address this problem of querying heterogeneous Web data using on the fly mappings computed in a pay-as-you-go fashion based on entity relevance models. This framework is instantiated involving the following four steps. (1) First, we compute an ERM from the results returned from the source dataset G_s using q_s. (2) Second, we treat q_s as keywords and using a standard IR-based search engine, we obtain result candidates from the target datasets G_t. (3) Then, a lightweight on the fly integration technique is employed, which maps the structure of result candidates to the structure of the ERM. (4) Finally, the result candidates are ranked according to their similarity to the ERM using the mappings computed at runtime.

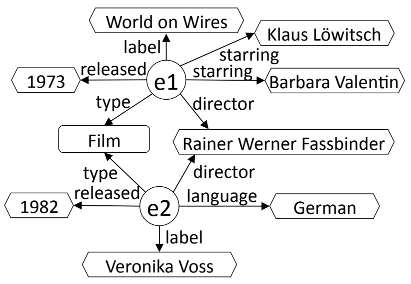


Figure 4: Example set R_s of two entities e_1, e_2 obtain for query q_s

3. SEARCH OVER HETEROGENEOUS 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 at building 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. The proposed model is called the *Entity Relevance Model* (*ERM*). The ERM builds upon the concept of language model, a statistical modeling technique frequently applied in Information Retrieval tasks. We start with a brief overview of language models for IR (see [17] for more details).

Language Model. The main idea here is to see documents and queries as samples from different probability distributions also called language models. More precisely, a language model is a multinomial distribution, which assigns

(1) ERM

a_s	$k(a_s)$	$w : P_s(w\|a_s)$
label	1	world:0.2, on:0.2, wires:0.2, ...
starring	0.5	klaus:0.25, löwitsch:0.25, barbara:0.25,...
director	1	rainer:0.33, werner:0.33, fassbinder:0.33
released	1	1973:0.5, 1982:0.5
language	0.5	german:1
type	1	film:1

(2) e_t

a_t	$w : P_t(w\|a_t)$
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

Figure 5: (1) ERM constructed from the entities e_1, e_2 of Figure 4. The ERM has a field for each attribute with label a_s. Each field is weighted with $k(a_s)$ and has a language model $P_s(w|a_s)$ defining the probability of w occurring in field a_s.
(2) Representation of the entity e_i of Figure 1 with language models for each attribute labeled a_t.

a probability to every word w in the vocabulary. Considering the underlying statistical process that leads to the generation of query and document samples, the corresponding query and document language models can be reconstructed when the samples are large and representative. A Maximum Likelihood Estimator is often used for this. For example, given a document corpus C, the vocabulary of terms V, the language model $P(w|D)$ representing document $D \in C$ can be estimated as follows:

$$P(w|D) = \lambda \frac{n(w, D)}{|D|} + (1 - \lambda) P(w|C) \tag{1}$$

where $n(w, D)$ is the count of word w in D, $|D|$ is the document length, and $P(w|C)$ is a background probability, which is used for smoothing controlled by the parameter λ. While a query Q may be too short as a sample, it has been shown that pseudo-relevance feedback (PRF) results obtained for it can serve as a representative sample of the information need, from which a query model (called relevance model) can be reconstructed. Thus, instead of the query, a relevance model $P(w|Q)$ is reconstructed of PRF results [14]. For ranking, a document D is considered relevant to a given query Q, if their probability distributions are close in "distance". One way to achieve this is using the negative cross-entropy $-H$:

$$H(Q||D) = \sum_{w \in V} P(w|Q) \log P(w|D) \tag{2}$$

We adopt this modeling approach to the problem of searching structured Web data. Our goal is to model both the structure and the content of entities. The idea behind the ERM is to represent the attribute structure of entities by a set of language models, and each language model captures the content of the respective attribute. Hence, instead of using language models to represent entire documents, we use them for modeling attribute values.

Entity Relevance Model. The $ERM = (R_s, A_s, \mathcal{P}_s)$ is a composite model consisting of a set of entities $R_s \subseteq N_E$, a set of attributes $A_s \subseteq E$, and a set of language models $\mathcal{P}_s$. Each $P_s \in \mathcal{P}_s$ is associated with a weight defined through the function $k : \mathcal{P}_s \rightarrow [0, 1]$. The entities R_s are obtained by submitting the query q_s against the source dataset G_s and used as pseudo-relevance feedback. A_s denotes the set of all distinct attribute labels that are associated with the entities

R_s, i.e. $A_s = \{a | a \in A'(e), e \in R_s\}$. For each distinct attribute label $a_s \in A_s$, we compute a corresponding language model $P_s(w|a_s) \in \mathcal{P}_s$ and its weight $k(a_s)$. The language model $P_s(w|a_s)$ specifies the probability of any word $w \in V$ occurring in the nodes of data graph edges with label a_s, where V is the vocabulary of all words. Let $N(e_i, a_s)$ be the set of nodes that is connected with e_i through edges with label a_s, i.e. $N(e_i, a_s) = \{e_j | a_s(e_i, e_j) \in E\}$, we compute $P_s(w|a_s)$ from all entity descriptions for $e_i \in R_s$ as follows:

$$P_s(w|a_s) = \frac{\sum_{e_i \in R_s} \sum_{e_j \in N(e_i,a_s)} n(w, e_j)}{\sum_{e_i \in R_s} \sum_{e_j \in N(e_i,a_s)} |e_j|} \tag{3}$$

where $n(w, e)$ denotes the count of word w in the node e and $|e|$ is the length of e (the number of words contained in e). The outer sum goes over the entities $e_i \in R_s$ and the inner sum goes over all values e_j of attributes with labels a_s. Thus, entity descriptions, which do not have the attribute a_s, do not contribute to $P_s(w|a_s)$. In order to capture the importance of these attribute-specific language models, we compute $k(a_s)$ as the fraction of entities having an attribute with label a_s:

$$k(a_s) = \frac{n(a_s, R_s)}{|R_s|} \tag{4}$$

where the numerator denotes the number of entities having an attribute with label a_s and the denominator is the total number of entities in R_s. In summary, an ERM can be seen as a query specific model built from pseudo-relevance feedback entities retrieved for the seed query q_s. An example for an ERM constructed from two entities is illustrated in Figure 5 (1).

3.2 Search 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 at finding entities in the target datasets which 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 G_t are relevant for the query q_s by measuring their similarity to the ERM and rank them by decreasing similarity. We model a candidate entity e_t analogously to the ERM: $e_t = (A_t, \mathcal{P}_t)$ where $A_t = A'(e_t)$ is the set of attributes of e_t and $\mathcal{P}_t$ is a set of language models. Similar to the ERM, $\mathcal{P}_t$ contains a language model $P_t(w|a_t)$ for each distinct attribute label $a_t \in A_t$. Let $N(a_t)$ be the set of value nodes of the attribute a_t, i.e. $N(a_t) = \{e_j | a_t(e_t, e_j) \in E\}$, $P_t(w|a_t)$ is estimated as follows:

$$P_t(w|a_t) = \frac{\sum_{e_j \in N(a_t)} n(w, e_j)}{\sum_{e_j \in N(a_t)} |e_j|} \tag{5}$$

Here, the sum goes over all values e of attributes with label a_t, $n(w, e)$ denotes that number of occurrences of w in e, and $|e|$ denotes the length of e. Figure 5 (2) illustrates an example.

We calculate the similarity between the ERM and a candidate entity e_t by measuring the "distance" between a language model of ERM and a language model of e_t using the (negative) cross entropy $-H$. We sum over these "distances" and weight each summand by $k(a_s)$ and the parameter $\beta_{(a_s)}$:

$$Sim(ERM, e_t) = \sum_{a_s \in A_s} \beta(a_s) \cdot k(a_s) \cdot H(P_s(w|a_s) || P_t(w|a_t) \tag{6}$$

The parameter β gives us the flexibility to boost the importance of attributes that occur in the query q_s as follows:

$$\beta(a_s) = \begin{cases} 1 & \text{if } a_s \notin q_s \\ b & \text{if } a_s \in q_s, b \geq 1 \end{cases} \tag{7}$$

In particular, we apply this similarity calculation only when we know which attribute label a_s of ERM should be matched against which attribute a_t 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 and a field a_s of ERM. Equation 6 applies to corresponding pairs of attribute and field. If there is no mapping between a_s and a_t, then we use a "maximum distance". This distance is computed as the cross entropy between $P_s(w|a_s)$ and a language model that contains all words in the vocabulary but the ones in $P_s(w|a_s)$.

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, $P_t(w|a_t)$ is smoothed using a collection-wide model $c_s(w)$, which captures the probability of w occurring in the entire dataset G_s. This smoothing is controlled by the Jelinek-Mercer parameter λ. As a result, the negative cross entropy $-H$ is calculated over the vocabulary V of field a_s as:

$$H(P_s||P_t) = \sum_{w \in V} P_s(w|a_s) \cdot \log(\ \lambda \cdot P_t(w|a_t) + (1 - \lambda) \cdot c_s(w)\ ) \tag{8}$$

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 q_s. The ERM is not only used for search, but also exploited for this alignment task. The details for computing mappings between entity attributes $a_t \in A_t$ and ERM fields $a_s \in A_s$ are presented in Algorithm (1). The rational of the algorithm is that a field a_s is aligned to an attribute a_t when the cross entropy H 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). 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 n and r are relatively small (see Table 1 and Table 3) because this algorithm operates only on entities that are requested as part of the search process compared to full-fledge upfront integration that takes the entire schema into account. Further, ranking requires the same computation (Equation 8) and thus the entropy values computed here are kept and subsequently reused for ranking. Moreover, for a faster performance, ERM fields having a weight of $k(a_s) < c$ can be pruned due to their negligible influence (see Section 4.6 and 4.7). In addition, existing mappings can be reused to reduce the number of comparisons even further.

Algorithm 1 On the fly Alignment

```
Input: $ERM$, Entity $e_t$, Threshold $t \in [0,1]$
Output: $Mappings\ A := \{(a_s, a_t) | a_s \in A_s, a_t \in A_t \cup null\}$
 1: $A := new\ Map$
 2: for all $a_s \in A_s$ do
 3:     $candMappings := new\ OrderedByValueMap$
 4:     for all $a_t \in A'(e_t)$ do
 5:         if $a_t \notin A.values$ then              // If not already aligned
 6:             $h \leftarrow H(P_s(w|a_s)||P_t(w|a_t))$     // see equation (8)
 7:             $candMappings.add(a_t, h)$
 8:         end if
 9:     end for
10:     $bestA \leftarrow candMappings.firstValue$
11:     $worstA \leftarrow candMappings.lastValue$
12:     if $bestA < t \cdot worstA$ then
13:         $a_t \leftarrow candMappings.firstKey$
14:         $A.add(a_s, a_t)$
15:     else
16:         $A.add(a_s, null)$                    // no mapping found
17:     end if
18: end for
19: return $A$
```

4. EXPERIMENTS

In this section, we report on the experiments conducted with the three solutions discussed in Section 2. We experimented with different parameter settings and observed that performance is stable when the employed parameters are in certain ranges (will be discussed in Section 4.6). Results reported in the following are obtained using the configuration: $b = 10$, $c = 0.8$, $t = 0.75$. The smoothing parameter λ, whose effect on retrieval performance has been studied extensively for IR tasks, was set to 0.9, a common value used in literature. We follow the *Cranfield*[6] methodology for the experiments on the search effectiveness and adopt the same methodology to analyze the effectiveness of mapping computation.

4.1 Datasets

Our experiments were conducted with 3 RDF Web datasets, *DBpedia 3.5.1*, *IMdb*, and *Amazon*. In every experiment, one of them serves as the source dataset and the other two represent the target datasets. 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 and Amazon datasets are retrieved from `www.imdb.com` and `www.amazon.com` [23], and then transformed into RDF. The IMdb dataset contains information about movies and films, whereas the Amazon dataset contains product information about DVDs and VHS Videos. These three datasets are representative for our Web scenario because a vertical search application running one of these datasets (e.g. the one owned by the company in our scenario) could benefit from incorporating the other two into the search process. Further, the datasets exhibit the heterogeneity previously illustrated in Figure 1. Table 1 gives details about each dataset.

Dataset	#Entities	#Distinct Attribute Labels	$\|A(e)\|$ ±StdDev.
Amazon	115K	28	18.4±3.8
IMdb	859K	32	11.4±6.4
DBpedia	9.1M	39.6K	9±18.2

Table 1: Dataset statistics

4.2 Queries and Ground Truth

Our goal is to find relevant entities in the target datasets $\mathbb{G}_t$ for a given query q_s. In this setting, we can determine the relevant entities in $\mathbb{G}_t$ by manually rewriting the query q_s to obtain a structured query q_t adhering to the vocabulary of $G_t \in \mathbb{G}_t$. Figure 6 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 retrieval experiments.

We created three query sets, each containing 23 SPARQL BGP entity queries of different complexities, ranging from 2 to 4 triple patterns that produce a varying number of results, see Table 2. The queries represent information needs like retrieve "movies directed by Steven Spielberg", "movies available in English and also in Hebrew", or "movies directed by Rainer Werner Fassbinder, which were released in 1982". The last query is illustrated in Figure 6.

Rel. Entities	Amazon	IMdb	DBpedia
max	153	834	47
avg.	32.2	114.9	10.9
median	18	21	5
min	1	1	1

Table 2: Results per query and dataset.

Source Dataset	$\|ERM\|$ ± StdDev.
Amazon	14.1±3.6
IMdb	15.8±6.7
DBpedia	23±5.4

Table 3: Average number of fields of an ERM.

4.3 Systems

We implement the strategies as discussed previously in Section 2.

Keyword Query (KW). IR style keyword search on Web data has been proposed [20, 5] and implemented as an adoption of Lucene[3], an IR engine, which applies a document and query length adjusted TF/IDF-based ranking function. We use the Semplore implementation [5], which uses a virtual document for every entity description and use the concatenations of attribute labels and attribute values as document terms. In the same way, we transform the structured query into a keyword query by using the concatenations of predicates and constants of the structured query as terms. The resulting keyword query retrieves all virtual documents representing entity descriptions, which contain some of the corresponding terms.

Query Rewriting (QR). This system is based on query rewriting using precomputed schema mappings. We created same-as mappings with the tools Falcon-AO [13] and Aroma [7] using their default configurations. Table 4 shows the number of mappings between the datasets. Then, to rewrite constants at runtime as discussed, we apply the KW baseline on top to limit the search results produced by the rewritten query to those that match constants formulated as a keyword query.

Datasets	Falcon-AO[13]	Aroma[7]
Amazon-IMdb	5	8
Amazon-DBpedia	11	11
IMdb-DBpedia	12	4

Table 4: Number of mappings.

Hybrid (ERM). 3 different versions are employed:
(1) *ERM* computes mappings on the fly.
(2) ERM_a relies entirely on the alignment computed upfront by Falcon-AO. This version of ERM can be seen as a combination of our approach and query rewriting that mimics the *QR* baseline. The precomputed mappings are used

[3]http://lucene.apache.org

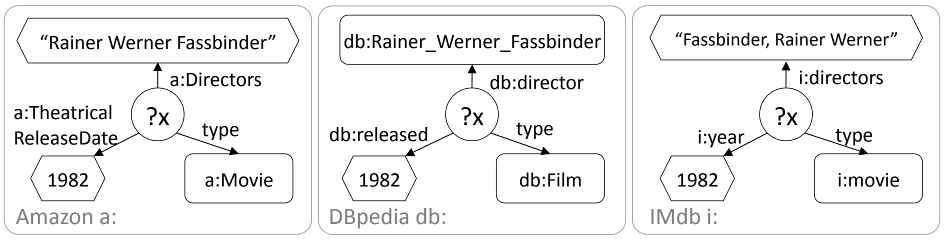


Figure 6: Example of manually created queries that serve as ground truth.

to obtained a rewritten query, which is processed to obtain results. However instead of using keyword search on top, we use the ERM and apply our approach for ranking.

(3) ERM_q combines these two approaches. It uses pre-computed mappings and creates additional mappings on the fly for those attributes, which could not be mapped upfront.

4.4 Search Effectiveness

We use the standard IR measures precision, recall, mean average precision (MAP) and mean reciprocal rank (MRR). We retrieve the top five thousand entities using the initial keyword search, rank them, and compute the metrics based on the top one thousand entities returned by each system. The results for six different retrieval settings are shown in Figure 7:

First, we examine the scenario without prior data integration. Here, finding relevant entities in the target dataset is only possible with KW or ERM. When comparing their results (Figure 7), we observe that ERM outperforms KW across all metrics and retrieval settings and improves over KW by 120% on average in terms of MAP. Looking at the different retrieval settings, we can see that ERM performs best between IMdb and Amazon (i.e. when IMdb or Amazon are either source or target dataset), where MAP are 0.8 and 0.95, respectively. The reason for this is that both datasets hold only entities from similar domains, movies and DVD/Videos, and describe them using similar attributes. DBpedia seems to be the most difficult one, mainly due to its schema complexity: It is very heterogeneous, containing information about different types of entities. Thus, whereas only one type have to be considered in the other datasets, identifying the relevant types out of a much larger set of possible candidates is also part of the retrieval problem here. Further, entities in DBpedia often exhibit redundant attributes with same values, e.g. *name*, *title* and *rdfs:label*, which leads to higher ambiguity during the computation of mappings. Across all retrieval settings, ERM yields MAP above 0.5. Also similarly good performance could be achieved for MRR and P@10, which consider the top of the ranked results. The robustness of the retrieval performance of ERM can be observed in Figure 8, which shows the interpolated precision across the recall levels. It can be observed that precision is fairly stable over different recall levels. One exception is the setting with IMdb as the target and DBpedia as the source dataset (Figure 8(e)). Here, performance decreases notably at recall levels above 0.3. This is because there are some outlier queries, which have much more relevant entities than others, and the rank of some entities obtained for these queries were relatively very low. However, P@R, where R is the number of relevant entities, is still above 0.5 even for this setting (Figure 7(c)).

In the next scenario, we examine the performance in the presence of precomputed alignments. Now, applying QR to retrieve entities is possible. This system considerably outperforms KW. Using pre-computed alignments with the hybrid approach, ERM_a, yields slightly better performance than ERM on average (see Figure 7). Both, ERM and ERM_a outperform QR on average by 54%, respectively 59% in terms of MAP. The performance of ERM_a diverges from ERM most notably in two cases: ERM_a is worse if IMdb and Amazon are involved. It is better in the retrieval setting with IMdb and DBpedia. This is because in the latter, the alignment problem that has to be solved as part of searching is more difficult due to the higher ambiguity and complexity introduced by DBpedia. Thus, applying the rewritten query using precomputed mappings to produce candidate results yields better performance. This effect can be observed in Figure 8(e). The strategy of combining the advantages of pre-computed mappings and computing alignments on the fly implemented by ERM_q outperforms the others across all metrics (see Figure 7).

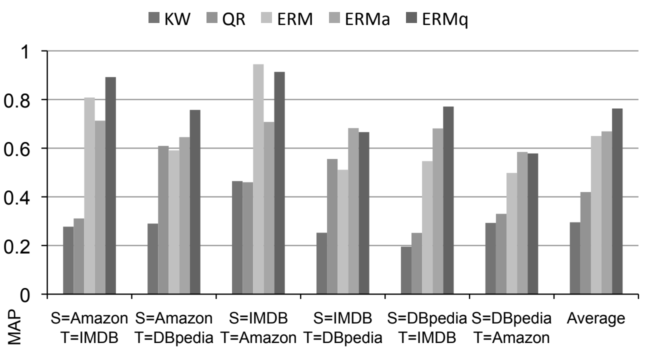
4.5 On The Fly Mappings

We assessed the mappings computed on the fly during the previously discussed experiments. First, we collected all mappings and manually determined the ground truth based on the pooled mappings. Since we operate on heterogeneous datasets, multiple correct mappings for one attribute are possible, e.g. *title* in one dataset might correctly corresponds to *title*, *name* and *label* in another dataset. Given this ground truth, we computed precision and recall of the mappings created between the fields of an ERM and the attributes of an entity. Table 3 shows the average size of an ERM and Table 1 provides the average description size of an entity. Precision and Recall are here defined as follows:

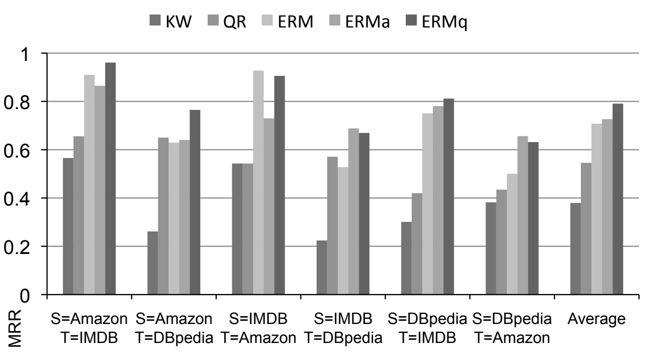
$$Precision = \frac{|\{\text{correct mappings}\}|}{|\{\text{created mappings}\}|} \tag{9}$$

$$Recall = \frac{|\{\text{correct mappings}\}|}{|\{\text{possible, correct mappings}\}|} \tag{10}$$

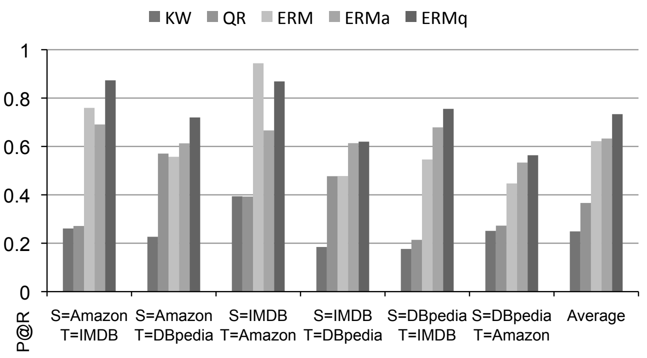
where $\{\textit{possible, correct mappings}\}$ is the set of mappings, which could be established between the ERM and an entity as captured by the ground truth. We computed precision and recall for each individual entity considered during search, averaged over the query and finally over the entire query set. Overall, mappings obtained for 115k entities and the ERMs are taken into account. Figure 9(a) shows precision and recall for the different retrieval settings. Averaging over all entities, precision is 0.46 and recall is 0.12. However, we are primarily interested in the entities, which are actually relevant. Therefore, we examine precision and recall only for these relevant entities. Here, the average over all scenarios is 0.70 for precision and 0.30 for recall, as shown in Figure 9(a). Figure 9(b) gives the average number of actual mappings created between the ERM and entities, and be-



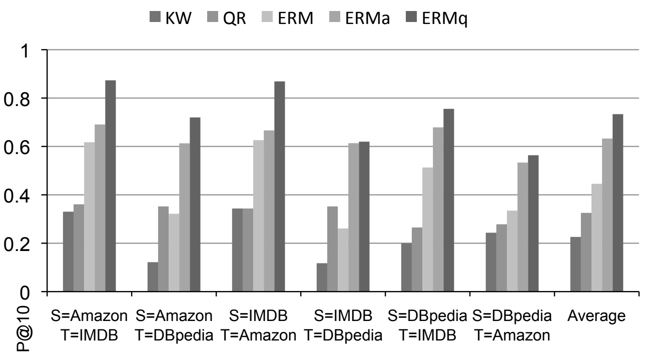
(a) Mean Average Precision (MAP)



(b) Mean Reciprocal Rank (MRR)



(c) R-Precision (P@R)



(d) Precision at 10 (P@10)

Figure 7: Retrieval performance between source dataset (S) and target dataset (T).

tween the ERM and relevant entities. Clearly, better results can be achieved for relevant entities. This is important for our search task, which is focused on finding these relevant entities.

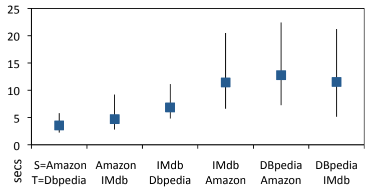
Intuitively, the search performance depends on the quality of the alignment. We verified this by computing the Pearson correlation coefficient ρ between the search performance of the different settings captured by MAP, as reported in Figure 7(a), and the alignment quality in terms of precision and recall for relevant entities, as reported in Figure 9(a). This yields $\rho(\text{MAP}, \text{Precision-Rel}) = 0.98$ and $\rho(\text{MAP}, \text{Recall-Rel}) = 0.97$, indicating strong dependency between quality of the mappings and search performance.

4.6 Parameter Analysis

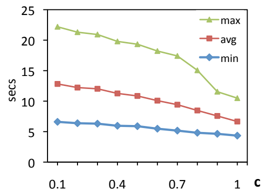
The hybrid approach relies on three parameters: b for boosting fields (attributes) in the seed query (Equation 7), the alignment threshold t and the threshold c for pruning fields of the ERM (Section 3.3). We analyze the robustness of search effectiveness in terms of MAP for the six retrieval scenarios by varying one parameter while keeping the others fixed at the levels we used for the experiments. The results are shown in Figure 10. We observed that boosting helps to improve the performance when dealing with similar datasets (i.e. Amazon and IMDB) but has a negative effect when a different and diverse dataset like DBpedia is involved. However, performance is rather insensitive to this parameter when $b > 10$ (thus we chose $b = 10$). Regarding the alignment threshold t, we observed that performance is fairly stable when t is within the range $[0.2, 0.8]$. Pruning fields has almost no effect on effectiveness.

4.7 Runtime Performance

To analyze the performance of the hybrid approach, we measured query execution time for ERM across all six retrieval scenarios, i.e. for a total of 138 queries. Figure 11(a) shows the min, max and average time in seconds for each retrieval scenario. The times reported cover all steps of the retrieval process, i.e. executing q_s to obtain results for the source dataset, computing ERM, retrieving results for target datasets, computing models for each candidate entity, establishing mappings and ranking. Such a retrieval process takes less than 13 secs on average for the above configuration. The performance can be improved by increasing the pruning parameter c as shown in Figure 11(b), which shows the min, max, and average query execution time over all six scenarios for different values of c. For these runtime experiments, we use a standard laptop with Intel Core 2 Duo 2.4 GHz CPU, 4 GB RAM, Serial-ATA HDD@5400rpm, MacOS 10.6, and implemented our approach using Java 6 and Lucene 3.0 for indexing and retrieval. Computing the language models from the term-frequency vectors was performed at runtime. These tasks can also be performed at indexing time. Still, these preliminary results suggest that the hybrid approach is promising, given that not only search results but also on the fly mappings are obtained during the process.

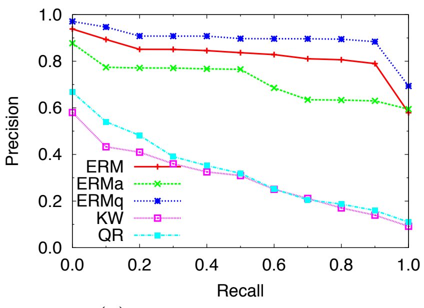


(a) Runtime performance ERM

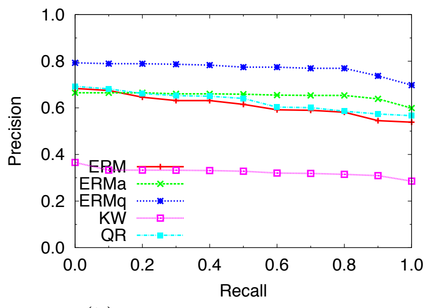


(b) Prunning c

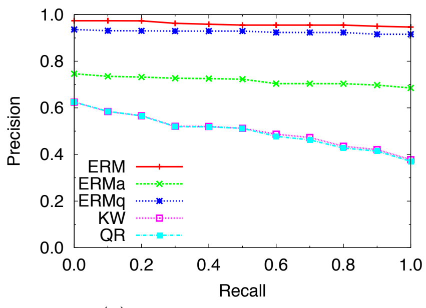
Figure 11: Runtime performance analysis



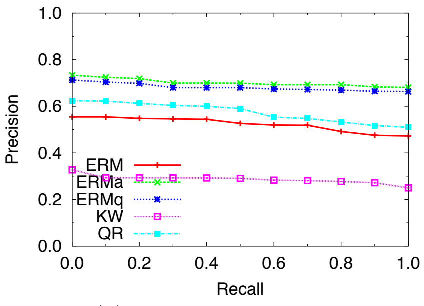
(a) S=Amazon T=IMdb



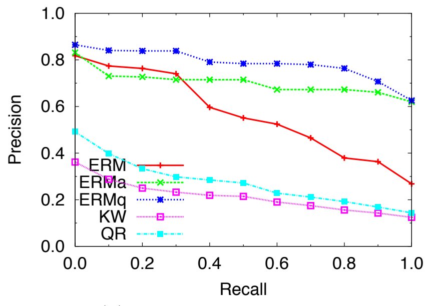
(b) S=Amazon T=DBpedia



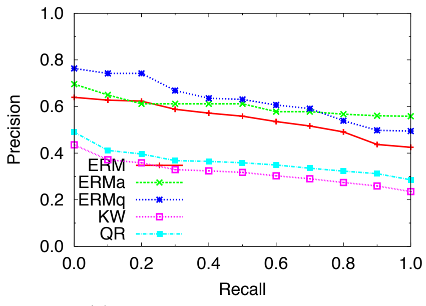
(c) S=IMdb T=Amazon



(d) S=IMdb T=DBpedia

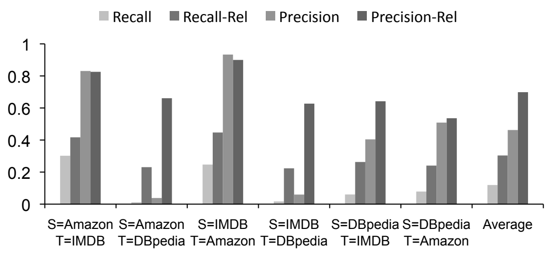


(e) S=DBpedia T=IMdb

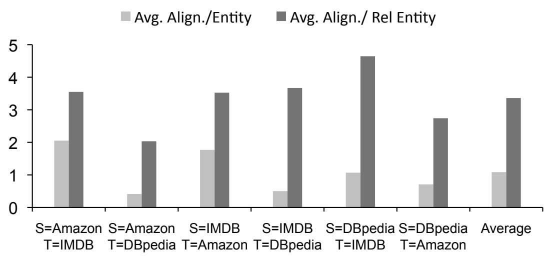


(f) S=DBpedia T=Amazon

Figure 8: Precision-recall curves for source dataset (S) and target dataset (T).



(a) Precision and recall of the mappings created on the fly between ERM and entities, respectively relevant entities (Rel).



(b) Average number of mappings created on the fly between ERM and entities, respectively relevant entities (Rel).

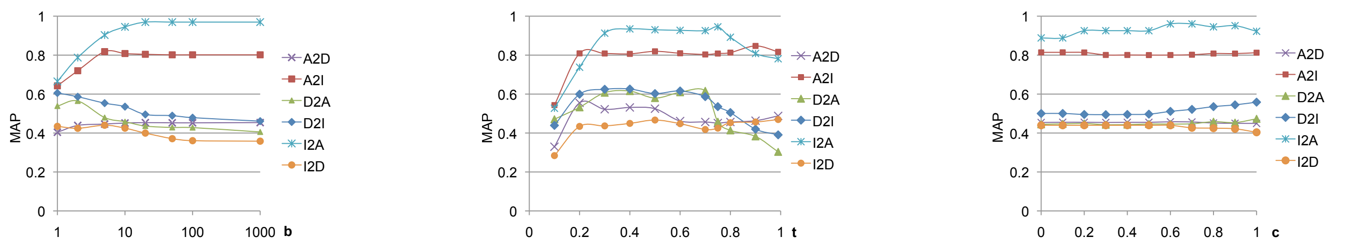
Figure 9: Evaluation of the mappings created on the fly.

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 [2, 5, 20] and the other one is structured query rewriting [3, 23, 10]. The latter type of approaches uses precomputed mappings, finds duplicates [23] or uses precomputed relaxations of the query constraints [10] to bridge differences in syntactical representation. The keyword search approaches rely on matches on the level of terms. Beside the pure 'bag-of-word' approaches [5, 20], a recent study showed that using a minimal structure by classifying attributes into *important* and *unimportant* fields improves keyword search for entities [2].

Our approach represents a novel combination which combines the flexibility of keyword search with the power of structured querying. 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 alignment tools like [13, 7]. We presented the general idea of our approach and preliminary results in [12].

Our work leverages several ideas that are have been proposed for IR tasks. In fact, the model underlying our approach originates from the concept of language models [17], which have been proposed for modeling resources and queries as multinomial distributions over the vocabulary terms, and for ranking based on the distance between the two models, e.g. using KL-divergence [22] or cross entropy [14] as measures. More precisely, the foundation of our work is established by Lavrenko et al.[14], who propose relevance-based language models to directly capture the relevance behind document and queries. Also, structure information has been exploited for constructing structured relevance models [15] (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. Thus, we build on well studied concepts and investigate them in a scenario different from the traditional IR settings. Instead of searching documents using keyword queries, we show how to use structured language models to process structured queries against structured data residing in external Web datasets. In this scenario, we also need to take structure mismatches (i.e. differences at the schema level) into account and thus, propose on the fly integration to deal with this problem.



(a) Field boosting b (b) Alignment threshold t (c) ERM pruning c

Figure 10: Parameter sensitivity analysis. The legend 'A2D' stands for source dataset=Amazon (A), target dataset=DBpedia (D), 'I2A' stands for source=IMdb (I), target=Amazon, etc.

The proposed technique is in principle similar to existing work on schema matching, e.g. [9], 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 we have not seen them applied to the schema mapping problem before. We consider this as a promising approach for embedding the pay-as-you-go data integration paradigm [16] into the search process.

6. CONCLUSION

We have proposed a novel approach for searching heterogeneous Web datasets using one single structured seed query that adheres to the vocabulary of just one of the datasets. We have introduced the entity relevance model which captures the structure and content of relevant results obtained for a seed query. The entity relevance model is used for matching and ranking results from external datasets, as well as for performing data integration on the fly. Our approach combines the flexibility of keyword search in the sense that no upfront integration is required, with the power of structured querying that comes from the use of the fine-grained structure of query and results. Extensive experiments conducted with real-world datasets show the effectiveness and feasibility of our approach.

Using our approach allows to take advantage of the structured data available numerously as Linked Data on the Web by incorporating these datasources into existing vertical search capabilities.

As future work, we will extend this approach to allow for more general queries representing graph patterns. The main idea will remain the same: building a relevance model from the seed query, querying external structured data using the seed query as keywords, and finally, mapping the structure of the results to the relevance model to rank them. However, results here are more complex, involving entities of different types that are possibly connected over long paths. We will employ existing techniques for keyword search on structured data (e.g. [19]) to retrieve these results, i.e., subgraphs in the data, which connect entities matching the query keywords. Also, we will extend our ideas for on-the-fly mapping and ranking to deal with the more complex structure of the queries and results.

7. ACKNOWLEDGEMENTS

We thank our colleagues Philipp Sorg and Günter Ladwig for helpful discussions. Also, we thank Julien Gaugaz and the L3S Research Center for providing us their versions of the IMdb and Amazon datasets. This work was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen project (grant 01IA08005K).

8. REFERENCES

[1] W3C Recommendation 15 January 2008, SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[2] R. Blanco, P. Mika, and S. Vigna. Effective and Efficient Entity Search in RDF data. In *ISWC*, pages 83–97, 2011.

[3] A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *IJCAI*, pages 16–21, 2003.

[4] S. Chaudhuri, B.-C. Chen, V. Ganti, and R. Kaushik. Example-driven Design of Efficient Record Matching Queries. In *VLDB*, pages 327–338, 2007.

[5] G. Cheng and Y. Qu. Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. *Int. J. Semantic Web Inf. Syst.*, 5(3):49–70, 2009.

[6] C. Cleverdon. The CRANFIELD Tests on Index Langauge Devices. *Aslib*, 1967.

[7] J. David. AROMA results for OAEI 2009. In *Ontology Matching Workshop, ISWC*, 2009.

[8] A. Doan and A. Y. Halevy. Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine*, 26(1):83–94, 2005.

[9] S. Duan, A. Fokoue, and K. Srinivas. One size does not fit all: Customizing Ontology Alignment Using User Feedback. In *ISWC*, pages 177–192, 2010.

[10] S. Elbassuoni, M. Ramanath, and G. Weikum. Query Relaxation for Entity-Relationship Search. In *ESWC*, pages 62–76, 2011.

[11] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[12] D. Herzig and T. Tran. One Query To Bind Them All. In *COLD2011*, CEUR Workshop Proceedings, Vol. 782, 2011.

[13] W. Hu and Y. Qu. Falcon-AO: A practical ontology matching system. *J. Web Sem.*, 6(3):237–239, 2008.

[14] V. Lavrenko and W. B. Croft. Relevance-based language models. In *SIGIR*, pages 120–127, 2001.

[15] V. Lavrenko, X. Yi, and J. Allan. Information retrieval on empty fields. In *HLT-NAACL*, pages 89–96, 2007.

[16] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[17] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR*, pages 275–281, 1998.

[18] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc Object Retrieval in the Web of Data. In *WWW*, pages 771–780, 2010.

[19] 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 *ICDE*, pages 405–416, 2009.

[20] H. Wang, Q. Liu, T. Penin, L. Fu, L. Zhang, T. Tran, Y. Yu, and Y. Pan. Semplore: A scalable IR approach to search the Web of Data. *J. Web Sem.*, 7(3):177–188, 2009.

[21] H. Wang, T. Tran, P. Haase, T. Penin, Q. Liu1, L. Fu, , and Y. Yu. SearchWebDB: Searching the Billion Triples! In *Semantic Web Challenge, ISWC*, 2008. http://www.cs.vu.nl/~pmika/swc-2008/SearchWebDB-paper.pdf.

[22] C. Zhai and J. D. Lafferty. A Risk Minimization Framework for Information Retrieval. *Inf. Process. Manage.*, 42(1):31–55, 2006.

[23] X. Zhou, J. Gaugaz, W.-T. Balke, and W. Nejdl. Query Relaxation Using Malleable Schemas. In *SIGMOD Conference*, pages 545–556, 2007.