Query Relaxation for Entity-Relationship Search

Shady Elbassuoni
Maya Ramanath
Gerhard Weikum

MPI–I–2010–5-008 December 2010

Authors' Addresses

Shady Elbassuoni
Max-Planck Institute for Informatics
Campus E1.4
66123 Saarbrücken
Germany

Maya Ramanath
Max-Planck Institute for Informatics
Campus E1.4
66123 Saarbrücken
Germany

Gerhard Weikum
Max-Planck Institute for Informatics
Campus E1.4
66123 Saarbrücken
Germany

Abstract

Entity-relationship-structured data is becoming more important on the Web. For example, large knowledge bases have been automatically constructed by information extraction from Wikipedia and other Web sources. Entities and relationships can be represented by subject-property-object triples in the RDF model, and can then be precisely searched by structured query languages like SPARQL. Because of their Boolean-match semantics, such queries often return too few or even no results. To improve recall, it is thus desirable to support users by *automatically relaxing* or reformulating queries in such a way that the intention of the original user query is preserved while returning a sufficient number of ranked results.

In this paper we describe comprehensive methods to relax SPARQL-like triple-pattern queries, possibly augmented with keywords, in a fully automated manner. Our framework produces a set of relaxations by means of statistical language models for structured RDF data and queries. The query processing algorithms merge the results of different relaxations into a unified result list, with ranking based again on language models. Our experimental evaluation, with two different datasets about movies and books, shows the effectiveness of the automatically generated relaxations and the improved quality of query results based on assessments collected on the Amazon Mechanical Turk platform.

Keywords

rdf, sparql, relaxation, entity, relationship, search

Contents

1 Introduction

1.1 Motivation

There is a trend towards viewing Web or digital-library information in an entity-centric manner: what is the relevant information about a given sports club, a movie star, a politician, a company, a city, a poem, etc. Moreover, when querying the Web, news, or blogs, we like the search results to be organized on a per-entity basis. Prominent examples of this kind of search are entitycube.research.microsoft.com or google.com/squared/. Additionally, services that contribute towards more semantic search are large knowledge repositories, including both handcrafted ones such as freebase.com as well as automatically constructed ones such as trueknowledge.com or dbpedia.org. These have been enabled by knowledge-sharing communities such as Wikipedia and by advances in information extraction (e.g., [9, 31, 71, 90, 77]).

One way of representing entity-centric information, along with structured relationships between entities, is the Semantic-Web data model RDF. An RDF collection consists of a set of subject-property-object (SPO) triples. Each triple is a pair of entities with a named relationship. A small example about books is shown in Table 1.1.

When triples are extracted from Web pages, they can be associated with weights, reflecting extraction confidence, "witness" count (number of different pages where the triple was seen), etc. Moreover, since text documents can not be completely cast into structured data, keeping the *textual context* from which the triples were extracted can be a valuable asset for search. This way, we can combine structured querying for RDF triples with keyword or phrase conditions about the associated context.

RDF data of this kind can be queried using a conjunction of *triple patterns*, where a triple pattern is a triple with variables and the same variable in different patterns denotes a join condition. For example, searching for Pulitzer-prize winning science fiction authors from the USA could be phrased as:

Subject (S)	**Property (P)**	**Object (O)**
Carl_Sagan	wrote	Contact
Carl_Sagan	type	American_Writer
Carl_Sagan	type	Astronomer
Carl_Sagan	bornIn	USA
Carl_Sagan	wonAward	Pulitzer_Prize
Contact	type	novel
Contact	hasGenre	Science_Fiction
Contact	hasTag	aliens
Contact	hasTag	philosopy
Jon_Krakauer	wrote	Into_the_Wild
Into_the_Wild	type	biography
Into_the_Wild	hasTag	adventure
Into_the_Wild	hasTag	wilderness
Jon _Krakauer	hasBestseller	Into_Thin_Air
Jon _Krakauer	citizenOf	USA

Table 1.1: RDF triples

?a wrote ?b ; ?b hasGenre Science_Fiction ;
?a wonAward Pulitzer_Prize ; ?a bornIn USA

This query contains a conjunction (denoted by ";") of four triple patterns where ?a and ?b denote variables that should match authors and their books respectively.

While the use of triple patterns enables users to formulate their queries in a precise manner, it is possible that the queries are overly constrained and lead to unsatisfactory recall. For example, this query would return very few results even on large book collections, and only one - Carl Sagan - for our example data. However, if the system were able to automatically reformulate one or more conditions in the query, say, replacing bornIn with citizenOf, the system would potentially return a larger number of ranked results.

Additionally, each triple pattern can be associated with *keywords* in order to query the associated context. For example, consider a query asking for non-fiction best-seller books about mountaineering: ?a wrote ?b{*non-fiction, best-seller, mountaineering*}

This query contains a single triple pattern augmented with 3 keywords. In this case, the system has to return triples that match the structured part of the query, and additionally whose context match the keywords in the query. Again, relaxing keyword conditions (for example, by matching a subset of

the keywords in the query), could potentially improve recall.

Note that the query expressions considered here should be seen as an API, not as an end-user interface. As SPARQL is gaining momentum for the Linking-Open-Data initiative and other RDF sources on the "Web of Data", it serves as an API for building semantically rich application on top of such a query language.

1.2 Query Relaxation Problem

This paper introduces the *query relaxation* problem: automatically broadening or reformulating triple-pattern queries – which may be augmented with keywords – to retrieve more results without unduly sacrificing precision. We can view this problem as the entity-relationship-oriented counterpart of query expansion in the traditional keyword-search setting. Automatically expanding queries in a robust way so that they would not suffer from topic drifts (e.g., overly broad generalizations) is a difficult problem [15].

The problem of query relaxation for triple-pattern queries has been considered in limited form in [89, 49, 32, 48] and our previous work [33]. Each of these prior approaches focused on very specific aspects of the problem, and only two of them [89, 33] conducted experimental studies on the effectiveness of their proposals. These techniques are discussed in more detail in Section 2.

Our Approach. This paper develops a comprehensive set of query relaxation techniques, where the relaxation candidates can be derived from both the RDF data itself as well as from external ontological and textual sources. Our framework is based on statistical language models (LMs) and provides a principled basis for generating relaxed query candidates. Moreover, we develop a model for holistically ranking results of both the original query and different relaxations into a unified result list.

Our query relaxation framework consists of the following four types of relaxations:

- Entities (subject, object) and relations (property) specified in a triple pattern are relaxed by substituting with related entities and relations. For example, bornIn could be substituted with citizenOf or livesIn and Pulitzer_Prize could be replaced by Hugo_Award or Booker_Prize.

- Entities and relations specified in a triple pattern could be substituted with variables. For example, Pulitzer_Prize could be replaced by ?p to

cover arbitrary awards or wonAward could be replaced by ?r, allowing for matches such as nominatedFor and shortlistedFor.

- Triple patterns from the entire query could be either removed or made optional. For example, the triple pattern ?b hasGenre Science_Fiction could be removed entirely, thus increasing the number of authors returned.

- Keywords associated with triple patterns may be dropped entirely, or only a subset of them considered. Additionally, expansion terms could be also considered for each keyword.

1.3 Contributions

The technical contributions of this paper are the following:

- We develop a novel, comprehensive framework for different kinds of query relaxation, in an RDF setting, based on language modeling techniques.

- Our framework can incorporate external sources such as ontologies and text documents to generate candidate relaxations.

- We harness the statistical LMs and relaxation weights for computing, in a principled manner, query-result rankings.

- We evaluate our model and techniques with two datasets – movie data from imdb.com and book information from the online community librarything.com, and show that our methods provide very good results in terms of NDCG.

The rest of the paper is organized as follows. We start by giving some technical background in Section 3. Our relaxation framework is described in Section 4. In Section 5 we describe our result-ranking model. We report on our evaluation in Section 6 and discuss related work in Section 2.

2 Related Work

One of the problems addressed in this paper is that of relaxing entities and relations with similar ones. This is somewhat related to both record linkage [63], and ontology matching [75]. However, the latter problems deal with finding identical concepts (whether column names, ontological concepts, entities, etc.), while we are concerned with finding a match which is *close in spirit* to a given entity or relation. When a given entity or relation does not have a candidate which is "close enough", we identify this and offer a substitution with a variable as the best relaxation candidate. Other kinds of query reformulations, such as spelling corrections, or using alternative words to express the same concept (e.g., 'Big Apple" for "New York"), etc. directly benefit from the research on record linkage, entity disambiguation and ontology matching, but are not the focus of this work.

Query reformulation in general has been studied in other contexts such as keyword queries [24] (more generally called query expansion), XML [5, 58], SQL [20, 89] as well as RDF [49, 32, 48]. While we can make use of query expansion strategies for keyword conditions, we deal primarily with triple-pattern queries over RDF data. The problem of query relaxation for triple-pattern queries is quite different from an XML or SQL setting, given that RDF provides graph-structured, non-schematic data. XML is limited to tree-structured data and relaxations on the navigational predicates (which are not found in triple pattern queries). And relational approaches expect a database schema and would not support variable substitutions.

For RDF triple-pattern queries, relaxation has been addressed to some extent in [89, 49, 32, 48, 33]. This prior work can be classified based on several criteria as described below. Note that except for [89] and our previous work in [33], none of the other papers report on experimental studies.

Scope of relaxations. With the exception of [32, 49], the types of relaxations considered in previous papers are limited. For example, [89] considers relaxations of relations only, while [48, 33] consider both entity and relation relaxations. The work in [33], in particular, considers a very limited form

of relaxation – replacing entities or relations specified in the triple patterns with variables. Our approach, on the other hand, considers a comprehensive set of relaxations and in contrast to most other previous approaches, weights the relaxed query in terms of the *quality* of the relaxation, rather than the number of relaxations that the query contains.

Relaxation framework. While each of the proposals mentioned generates multiple relaxed query candidates, the method in which they do so differ. While [32, 49, 48] make use of rule-based rewriting, the work in [89] and our own work make use of the data itself to determine appropriate relaxation candidates. Even though rule-based rewriting of queries provides the system developer with more control over the quality of rewriting, it potentially implies a lot of human input before-hand in order to formulate these rules. The human input could be in the form of designing the RDFS [49] for the data, or in an analysis of user and domain preferences [32]. In contrast, our approach is entirely automatic and does not require human-input, while still allowing for the incorporation of other sources, such as RDFS, in the relaxation process.

Result ranking. Our approach towards result ranking is the only one that takes a holistic view of both the original and relaxed query results. This allows us to rank results based on both the *relevance* of the result itself, as well as the *closeness* of the relaxed query to the original query. The "block-wise" ranking adopted by previous work – that is, results for the original query are listed first, followed by results of the first relaxation and so on – is only one strategy for ranking, among others, that can be supported by our ranking model.

3 Background

In this section, we describe the basic setting and some of the terminology used in the rest of this paper.

3.1 Knowledge Base

A knowledge base KB of entities and relations is a set of *triples*, where a triple is of the form $\langle e1, r, e2 \rangle$ with entities $e1$, $e2$ and relation r (or $\langle s, p, o \rangle$ with subject s, object o, and property p in RDF terminology). An example of such a triple is:

Carl_Sagan wrote Contact

Each triple t is associated with a set of keywords which captures the context from which the triple was extracted. Furthermore, each triple-keyword pair, denoted $\langle t, w_i \rangle$ is associated with a witness count $c(t; w_i)$, which indicates the number of times the triple t was extracted from the corpus (e.g., the Web) with the corresponding keyword w_i as part of its context. In addition, the witness count for just the triple t, denoted $c(t)$ is also stored. The witness count gives a measure of "importance" of the triple in the corpus.

3.2 Query

A query consists of *triple patterns* where a triple pattern is a triple with at least 1 variable. For example, the query "science-fiction books written by Carl Sagan" can be expressed as:

Carl_Sagan wrote ?b; ?b hasGenre Science_Fiction

consisting of 2 triple patterns. In addition, each triple pattern may be augmented with one or more keywords, which we refer to as a keyword-augmented triple pattern. For example, the query "Books by Carl Sagan

about the nuclear war" can be formulated using the single keyword-augmented triple pattern:

Carl_Sagan wrote ?b{*nuclear war*}

Given a query with k triple patterns, the result of the query is the set of all k-tuples that are isomorphic to the query when binding the query variables with matching entities and relations in KB. For example, the result for the first query includes the 2-tuple

Carl_Sagan wrote Contact; Contact hasGenre Science_Fiction

and a result for the second query includes the tuple

Carl_Sagan wrote A_Path_Where_No_Man_Thought

4 Relaxing Entities and Relations

As mentioned in the introduction, we are interested in 4 types of relaxations: i) replacing a constant (corresponding to an entity or a relation) in one or more triple patterns of the query with another constant which still reflects the user's intention, ii) replacing a constant in one or more triple patterns with a variable, iii) removing a triple pattern altogether, and, iv) relaxing keywords by considering only subsets of them. In this section, we introduce a framework which seamlessly integrates all four types of relaxations. Note that, even though we refer only to entities in the following, the same applies to relations as well.

Finding similar entities. For each entity $\mathsf{E_i}$ in the knowledge base KB, we construct a document $D(\mathsf{E_i})$ (the exact method of doing so will be described in Section 4.1). For each document $D(\mathsf{E_i})$, let $LM(\mathsf{E_i})$ be its language model. The similarity between two entities $\mathsf{E_i}$ and $\mathsf{E_j}$ is now computed as the distance between the LMs of the corresponding documents. Specifically, we use the square-root of the Jensen-Shannon divergence between two probability distributions (that is, $LM(\mathsf{E_i})$ and $LM(\mathsf{E_j})$, in this case), which is a metric. And so, for an entity of interest E, we can compute a ranked list of similar entities.

Replacing entities with variables. We interpret replacing an entity in a triple pattern with a variable as being equivalent to replacing that entity with *any* entity in the knowledge base.

We first construct a special document for the entire knowledge base, $D(\mathsf{KB})$. Let E be the entity of interest (i.e., the entity in the triple pattern to be replaced with a variable). Let $D(\mathsf{E})$ be its document. Now, we construct a document corresponding to "any" entity other than E as: $D(\mathsf{ANY}) = D(\mathsf{KB}) - D(\mathsf{E})$ (i.e., remove the contents of $D(\mathsf{E})$ from $D(\mathsf{KB})$). The simi-

larity between the entity E and "any" entity ANY is computed as the distance between the LMs of their corresponding documents.

In the ranked list of potential replacements for entity E, a variable replacement is now simply another candidate. In other words, the candidates beyond a certain rank are so dissimilar from the given entity, that they may as well be ignored and represented by a single variable.

Removing triple patterns. So far, we gave a high-level description of our relaxation technique for individual entities (or relations) in a triple pattern, without treating the triple pattern holistically. In a given query containing multiple triple patterns, a large number of relaxed queries can be generated by systematically substituting each constant with other constants or variables. We now consider the case when a triple pattern in the query contains only variables. In a post-processing step, we can now choose one of the following options. First, the triple pattern can be made optional. Second, the triple pattern can be removed from the query. To illustrate the two cases, consider the following examples.

Example 1: Consider the query asking for married couples who have acted in the same movie:

?a1 actedIn ?m; ?a2 actedIn ?m; ?a1 marriedTo ?a2

and a relaxation:

?a1 actedIn ?m; ?a2 ?r ?m; ?a1 marriedTo ?a2

Even though the second triple pattern contains only variables, retaining this pattern in the query still gives the user potentially valuable information – that ?a2 was related some how to the movie ?m. Hence, instead of removing this triple pattern from the query, it is only made optional – that is, a result may or may not have a triple which matches this triple pattern.

Example 2: Consider the query asking for movies which James Cameron produced, directed as well as acted in:

Cameron produced ?m; Cameron directed ?m; Cameron actedIn ?m

and a relaxation:

Cameron produced ?m; Cameron directed ?m; ?x ?r ?m

In this case, the last triple pattern matches any random fact about the movie ?m. This does not give any valuable information to the user as in the previous case and can be removed.

Relaxing keywords. Each keyword in the query can be associated with expansion terms [24] such as {*biography*} for {*non-fiction*} and {*mountain climbing*} for {*mountaineering*}, etc. Given a set of keywords, in principle, we would like to construct all possible subsets of the keywords and generate a relaxed query containing each of these subsets. For example, consider the query:

?a wrote ?b{*non-fiction, best-seller, mountaineering*}

We can now generate a set of 8 queries (including the original) each of which associates the triple ?a wrote ?b with a subset of terms {*non-fiction, best-seller, mountaineering*}.

However, this kind of relaxation is equivalent to interpreting the keyword matching as AND-ish (match as many keywords as possible). And so, instead of explicitly generating relaxed queries, we incorporate the AND-ish interpretation in our ranking model.

4.1 Constructing Documents and LMs

LibraryThing		**IMDB**	
Egypt	**Non-fiction**	**Academy_Award_for_Best_Actor**	**Thriller**
Ancient_Egypt	Politics	BAFTA_Award_ for_Best_Actor	Crime
Mummies	American_History	Golden_Globe_Award_ for_Best_Actor_Drama	Horror
Egyptian	Sociology	*var*	Action
Cairo	Essays	Golden_Globe_Award_ for_Best_Actor_Musical_or_Comedy	Mystery
Egyptology	History	New_York_Film_Critics_ Circle_Award_for_Best_Actor	*var*

Table 4.1: Example entities and their top-5 relaxations

We now describe how to construct documents for entities and relations and how to estimate their corresponding LMs.

Sources of information. The document for an entity should "describe" the entity. There are at least three different sources of information which we can leverage in order to construct such a document. First, we have the knowledge base itself – this is also the primary source of information in our case since we are processing our queries on the knowledge base. Second, we

could make use of external textual sources – for example, we could extract contextual snippets or keywords from text/web documents from which the triples were extracted. Third, we could also utilize external ontologies such as Wordnet, in order to find terms which are semantically close to the entity.

In this paper, our main focus is on utilizing the knowledge base as the information source and hence, we describe our techniques in this context and perform experiments using these techniques. But, our framework can be easily extended to incorporate other information sources and we briefly describe how this can be done at the end of this section.

4.1.1 Documents and LMs for entities

Let E be the entity of interest and $D(\mathsf{E})$ be its document, which is constructed as the set of all triples in which E occurs either as a subject or an object. That is,

$$D(\mathsf{E}) = \{\langle \mathsf{E\ r\ o} \rangle : \langle \mathsf{E\ r\ o} \rangle \in \mathsf{KB}\} \cup \{\langle \mathsf{s\ r\ E} \rangle : \langle \mathsf{s\ r\ E} \rangle \in \mathsf{KB}\}$$

We now need to define the set of terms over which the LM is estimated. We define two kinds of terms: i) "unigrams" U, corresponding to all entities in KB, and, ii) "bigrams" B, corresponding to all entity-relation pairs. That is,

$$U = \{\mathsf{e} : \langle \mathsf{e\ r\ o} \rangle \in \mathsf{KB} || \langle \mathsf{s\ r\ e} \rangle \in \mathsf{KB}\}$$

$$B = \{(\mathsf{er}) : \langle \mathsf{e\ r\ o} \rangle \in \mathsf{KB}\} \cup \{(\mathsf{re}) : \langle \mathsf{s\ r\ e} \rangle \in \mathsf{KB}\}$$

Example: The entity Woody_Allen would have a document consisting of triples Woody_Allen directed Manhattan, Woody_Allen directed Match_Point, Woody_Allen actedIn Scoop, Woody_Allen type Director, Federico_Fellini influences Woody_Allen, etc. The terms in the document would include Scoop, Match_Point, (type,Director), (Federico_Fellini,influences), etc.

Note that the bi-grams usually occur exactly once per entity, but it is still important to capture this information. When we compare the LMs of two entities, we would like identical relationships to be recognized. For example, if for a given entity, we have the bigram (hasWonAward, Academy_Award), we can then distinguish the case where a candidate entity has the term (hasWonAward, Academy_Award) and the term (nominatedFor, Academy_Award). This distinction cannot be made if only unigrams are considered.

Estimating the LM. The LM corresponding to document $D(\mathsf{E})$ is now a mixture model of two LMs: P_U, corresponding to the unigram LM and P_B,

the bigram LM. That is,

$$P_{\mathsf{E}}(w) = \mu P_U(w) + (1-\mu) P_B(w)$$

where μ controls the influence of each component. The unigram and bigram LMs are estimated in the standard way with linear interpolation smoothing from the corpus. That is,

$$P_U(w) = \alpha \frac{c(w; D(\mathsf{E}))}{\Sigma_{w' \in U} c(w'; D(\mathsf{E}))} + (1-\alpha) \frac{c(w; D(\mathsf{KB}))}{\Sigma_{w' \in U} c(w'; D(\mathsf{KB}))}$$

where $w \in U$, $c(w; D(\mathsf{E}))$ and $c(w; D(\mathsf{KB}))$ are the frequencies of occurrences of w in $D(\mathsf{E})$ and $D(\mathsf{KB})$ respectively and α is the smoothing parameter. The bigram LM is estimated in an analogous manner.

4.1.2 Documents and LMs for relations

Let R be the relation of interest and let $D(\mathsf{R})$ be its document, which is constructed as the set of all triples in which R occurs. That is,

$$D(\mathsf{R}) = \{\langle \mathsf{s}' \ \mathsf{R} \ \mathsf{o}' \rangle : \langle \mathsf{s}' \ \mathsf{R} \ \mathsf{o}' \rangle \in \mathsf{KB}\}$$

As with the case of entities, we again define two kinds of terms – "unigrams" and "bigrams". Unigrams correspond to the set of all entities in KB. But, we make a distinction here between entities that occur as subjects and those that occur as objects, since the relation is directional (note that there could be entities that occur as both). That is,

$$S = \{\mathsf{s} : \langle \mathsf{s} \ \mathsf{r} \ \mathsf{o} \rangle \in \mathsf{KB}\}$$

$$O = \{\mathsf{o} : \langle \mathsf{s} \ \mathsf{r} \ \mathsf{o} \rangle \in \mathsf{KB}\}$$

$$B = \{(\mathsf{so}) : \langle \mathsf{s} \ \mathsf{r} \ \mathsf{o} \rangle \in \mathsf{KB}\}$$

Example: Given the relation directed, $D(\mathsf{directed})$ would consist of all triples containing that relation, including, James_Cameron directed Aliens, Woody_Allen directed Manhattan, Woody_Allen directed Match_Point, Sam_Mendes directed American_Beauty, etc. The terms in the document would include James_Cameron, Manhattan, Woody_Allen, (James_Cameron, Aliens), (Sam_Mendes, American_Beauty), etc.

Estimating the LM. The LM of $D(\mathsf{R})$ is a mixture model of three LMs: P_S, corresponding to the unigram LM of terms in S, P_O, corresponding to the unigram LM of terms in O and P_B, corresponding to the bigram LM. That is,

$$P_R(w) = \mu_s P_S(w) + \mu_o P_O(w) + (1 - \mu_s - \mu_o) P_B(w)$$

where μ_s, μ_o control the influence of each component. The unigram and bigram LMs are estimated in the standard way with linear interpolation smoothing from the corpus. That is,

$$P_S(w) = \alpha \frac{c(w; D(\mathsf{R}))}{\Sigma_{w' \in S} c(w'; D(\mathsf{R}))} + (1 - \alpha) \frac{c(w; D(\mathsf{KB}))}{\Sigma_{w' \in S} c(w'; D(\mathsf{KB}))}$$

where $w \in S$, $c(w; D(\mathsf{R}))$ and $c(w; D(\mathsf{KB}))$ are the frequencies of occurrences of w in $D(\mathsf{R})$ and $D(\mathsf{KB})$ respectively, and α is a smoothing parameter. The other unigram LM and the bigram LM are estimated in an analogous manner.

4.1.3 Generating the candidate list of relaxations

As previously mentioned, we make use of the square root of the JS-divergence as the similarity score between two entities (or relations). Given probability distributions P and Q, the JS-divergence between them is defined as follows,

$$JS(P||Q) = KL(P||M) + KL(Q||M)$$

where, given two probability distributions R and S, the KL-divergence is defined as,

$$KL(R||S) = \Sigma_j R(j) \log \frac{R(j)}{S(j)}$$

and

$$M = \frac{1}{2}(P + Q)$$

4.2 Examples

Tables 4.1 and 4.2 show example entities and relations from the IMDB and LibraryThing datasets and their top-5 relaxations derived from these datasets, using the techniques described above. The entry *var* represents the variable candidate. As previously explained, a variable substitution indicates that there were no other *specific* candidates which had a high similarity to the given entity or relation. For example, in Table 4.2, the commentedOn relation

has only one specific candidate relaxation above the variable relaxation – hasFriend. Note that the two relations are relations between people - a person X could comment on something a person Y wrote, or a person X could have a friend Y - whereas the remaining relations are not relations between people. When generating relaxed queries using these individual relaxations, we ignore all candidates which occur after the variable. The process of generating relaxed queries will be explained in Section 5.

LibraryThing		**IMDB**	
wrote	**commentedOn**	**directed**	**bornIn**
hasBook	hasFriend	actedIn	livesIn
hasTagged	*var*	created	originatesFrom
var	hasBook	produced	*var*
hasTag	hasTag	*var*	diedIn
hasLibraryThingPage	hasTagged	type	isCitizenOf

Table 4.2: Example relations and their top-5 relaxations

4.3 Using other information sources

The core of our technique lies in constructing the document for an entity E or relation R and estimating its LM. And so, given an information source, it is sufficient to describe: i) how the document is constructed, ii) what the terms are, and, iii) how the LM is estimated. In this paper, we have described these three steps when the information source is the knowledge base of RDF triples. It is easy to extend the same method for other sources. For example, for the case of entities, we could make use of a keyword context or the text documents from which an entity or a triple was extracted. Then a document for an entity will be the set of all keywords or a union of all text snippets associated with it. The terms can be any combination of n-grams and the LM is computed using well-known techniques from the IR literature (see for example, entity LM estimation in [68, 72, 65, 37, 83], in the context of entity ranking).

Once individual LMs have been estimated for an entity from each information source, a straight-forward method to combine them into a single LM is to use a mixture model of all LMs. The parameters of the mixture model can be set based on the importance of each source. Note that this method does not preclude having different subsets of sources for different entities.

5 Relaxing Queries and Ranking Results

Q: ?x directed ?m; ?m hasWonPrize Academy_Award; ?m hasGenre Action		
L_1	L_2	L_3
?x directed ?m : 0.0	?m hasWonPrize Academy_Award : 0.0	?m hasGenre Action : 0.0
?x actedIn ?m : 0.643	?m hasWonPrize Golden_Globe : 0.624	?m hasGenre Adventure : 0.602
?x created ?m : 0.647	?m hasWonPrize BAFTA_Award : 0.659	?m hasGenre Thriller : 0.612
?x produced ?m : 0.662	?m ?r Academy_Award : 0.778	?m hasGenre Crime : 0.653

Table 5.1: Top-3 relaxation lists for the triple patterns for an example query

Q: ?x directed ?m; ?m hasWonPrize Academy_Award; ?m hasGenre Action	
Relaxed Queries	**score**
?x directed ?m;?m hasWonPrize Academy_Award;?m hasGenre Adventure	0.602
?x directed ?m;?m hasWonPrize Academy_Award;?m hasGenre Thriller	0.612
?x directed ?m;?m hasWonPrize Golden_Globe;?m hasGenre Action	0.624
?x actedIn ?m;?m hasWonPrize Academy_Award;?m hasGenre Action	0.643
?x created ?m;?m hasWonPrize Academy_Award;?m hasGenre Action	0.647

Table 5.2: Top-5 relaxed queries for an example query. The relaxed entities/relations are underlined.

We have so far described techniques to construct candidate lists of relaxations for entities and relations. In this section we describe how we generate relaxed queries and how results for the original and relaxed queries are ranked.

5.1 Generating Relaxed Queries

Let $Q_0 = q_1 q_2 ... q_n$ be the query, where q_i is a triple pattern. Let the set of relaxed queries be $R = \{Q_1, Q_2, ..., Q_r\}$, where Q_j is a query with one or more of its triple patterns relaxed. A triple pattern q_i is relaxed by relaxing at least one of its constants. Let s_j be the relaxation score of Q_j, computed by adding up the scores of all entity and relation relaxations in Q_j (recall that an individual entity or relation score is the square root of the JS-divergence between the LMs of the relaxed entity/relation and the original entity/relation). Clearly, the score s_0 for the original query Q_0 is 0 and the queries can be ordered in ascending order of s_js.

Example: Consider the query asking for Academy award winning action movies and their directors. Table 5.1 shows the lists L_1, L_2 and L_3 containing the top-3 closest relaxations for each triple pattern along with their scores. Table 5.2 shows the top-5 relaxed queries along with their scores.

5.2 Ranking Model

Our ranking model is based on our previous work in [33] and is summarized in this subsection. Our technique follows the language modeling approach in the context of keyword queries on a document corpus of [88]. In a nutshell, a query LM P_Q for the generation of query Q and a result LM P_G for the generation of result G (consisting of a tuple of triples) are estimated. The results are then ranked in increasing order of the KL-divergence between the query LM and the result LM.

5.2.1 Query LM

Let $Q_0 = \{q_1^0, ..., q_n^0\}$ be a query with n triple patterns where q_i^0 is a triple pattern. Let $Q_1, ..., Q_r$ be the set of relaxed queries and let q_i^j denote the i^{th} triple pattern of query Q_j.

In order for the query and result LMs to be comparable, they both have to be probability distributions over triples. To this end, the query LM is estimated as follows. Let $\hat{q}_i^j$ be the set of triples which match the triple pattern q_i^j. Let $T = \{t_1, ..., t_n\}$ be an n-tuple where t_i is a triple and $c(t_i)$ is the witness count of triple t_i. The query LM is a probability distribution P_Q over all possible n-tuples of triples. Assuming independence between triples, the probability of T is:

$$P_Q(T) = \prod_i P_Q(t_i) \tag{5.1}$$

where $P_Q(t_i)$ is estimated as:

$$P_Q(t_i) = \lambda_0 P_Q^0(t_i) + \lambda_1 P_Q^1(t_i) + ... + \lambda_r P_Q^r(t_i) \tag{5.2}$$

where $P_Q^j(t_i)$ is the probability of triple t_i in the query LM of the (relaxed) query Q_j, λ_j controls the impact of each relaxed query and $\sum \lambda_j = 1$. If λ_0 is set to 1 and the remaining λ_j's are set to 0, we get the exact match case where no relaxed query is taken into account. $P_Q^j(t_i)$ is computed as follows:

$$P_Q^j(t_i) = \begin{cases} \frac{c(t_i)}{\sum_{t \in \hat{q}_i^j} c(t)} & \text{if } t_i \in \hat{q}_i^j \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

One of the contributions of this paper is on strategies for setting the values of the λ_j's. This is discussed in Subsection 5.3.

As an example, consider the small knowledge base shown in Table 5.3. It shows the triples as well as their witness counts. Now consider a query with two patterns:

Carl_Sagan wrote ?b; ?b hasGenre ?g

asking for Carl Sagan's books and their genres. The query is factorized into two patterns, as shown in Table 5.4, and the matches of each pattern are retrieved. The probabilities of the matching triples are then calculated based on their witness counts and the sum of witness counts of all matching triples. The probabilities computed in this way are then used in Equation 5.1 to calculate 2-tuple probabilities in the query model[1].

5.2.2 Query LMs for keyword-augmented queries

We mentioned earlier that relaxing keyword conditions was equivalent in our case to treating the keyword condition as AND-ish. Our ranking model explicitly takes this into account. And so, we do not do any relaxed-query generation.

Let $Q_j = \{q_1^j, ..., q_n^j\}$ denote a query with n triple patterns. Furthermore, let q_i^j be a keyword-augmented pattern. That is, $q_i^j = q[w_1, ..., w_m]$ where q is a purely-structured triple pattern and w_k is an associated keyword. Analogous to the previous case, we need to compute the probability $P_Q(T)$ which is computed according to equations 5.1 and 5.2. However, in order to estimate $P_Q^j(t_i)$, we need to take into account the keywords associated with triple pattern q_i^j. That is, the probability of the triple t, given the context $w_1, ..., w_m$,

[1]Note that the set of all 2-tuples is derived from the cross-product of the sets $\{t_1, t_2, t_3\}$ and $\{t_6, t_7, t_8, t_9, t_{10}\}$

#	**Triple** (t_i)	$c(t_i)$
t_1.	Carl_Sagan wrote Contact	200
t_2.	Carl_Sagan wrote Other_Worlds	50
t_3.	Carl_Sagan wrote Cosmos	250
t_4.	John_Krakauer wrote Into_the_Wild	300
t_5.	John_Krakauer wrote Into_Thin_Air	200
t_6.	Contact hasGenre Science_Fiction	100
t_7.	Other_Worlds hasGenre Nonfiction	50
t_8.	Cosmos hasGenre Science	200
t_9.	Into_the_Wild hasGenre Biography	150
t_{10}.	Into_Thin_Air hasGenre Nonfiction	100

Table 5.3: Small knowledge base with triples and witness counts

$\hat{q_1}$	$P_Q(t)$	$\hat{q_2}$	$P_Q(t)$
t_1	200/500	t_6	100/600
t_2	50/500	t_7	50/600
t_3	250/500	t_8	200/600
		t_9	150/600
		t_{10}	100/600

Table 5.4: Query LM estimation for a query with two patterns: q_1 = Carl_Sagan wrote ?b and q_2 = ?b hasGenre ?g

where $t \in \hat{q}_i^j$. Note that $\hat{q}_i^j$ matches only the structured part of the triple pattern q_i^j. Assuming independence between keywords, we estimate the triple probability as:

$$P_Q^j(t_i|w_1, ..., w_m) = \prod_{k=1}^{m} [\alpha P_Q^j(t_i|w_k) + (1-\alpha)P(t_i)] \tag{5.4}$$

where $P_Q(t_i|w_k)$ is the probability of t_i given the single-term w_k, $P(t_i)$ is a smoothing component, and the parameter α controls the influence of smoothing. Now, in order to estimate the first component of Equation 5.4, we have:

$$P_Q^j(t_i|w_k) = \begin{cases} \frac{c(t_i;w_k)}{\sum_{t \in \hat{q}_i^j} c(t;w_k)} & \text{if } t_i \in \hat{q}_i \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

With the use of Equation 5.4 to compute the triple probabilities, we no longer need to explicitly relax keywords in the query. For example, for the triple pattern: ?a wrote ?b{*non-fiction, best-seller, mountaineering*} , we do not

need to generate the following relaxations: ?a wrote ?b{*non-fiction, best-seller*} or ?a wrote ?b{*best-seller, mountaineering*}, etc. This is because the smoothing component automatically takes care of the case when one or more keywords are not present with the triple

5.2.3 Result LM

For query Q with n triple patterns, let G be a result. Now, the LM of G, denoted P_G, is estimated over all n-tuples as:

$$P_G(T) = \beta P(T|G) + (1 - \beta)P(T|KB)$$

where β is the smoothing parameter. If G is the tuple T (note that G has to be an n-tuple), then $P(T|G) = 1$, otherwise, $P(T|G) = 0$.

For smoothing, independence between triples is assumed:

$$P(T|KB) = \prod_{i=1}^{n} P(t_i|KB)$$

where $P(t_i|KB)$ is estimated given the entire knowledge base:

$$P(t_i|KB) = \frac{c(t_i)}{\sum_{t \in KB} c(t)}$$

5.2.4 Ranking

Given the query and results LMs, the KL-divergence between the query LM P_Q of query Q and a result LM P_G of result G is computed as follows:

$$KL(Q||G) = \sum_i P_Q(T_i) log \frac{P_Q(T_i)}{P_G(T_i)}$$

The results are then returned to the user in ascending order of KL-divergence.

5.3 Setting Weights for Relaxations

An important and non-trivial aspect in the result ranking is the setting of weights of each relaxed query. Ideally these weights would be learned by taking into account relevance judgments of users for original or relaxed queries, user preferences, etc. Moreover, these weights should be learned on a query-to-query basis, which means that large query logs with a variety of queries

are required. Since we do not have access to such query logs, we instead make the following simplifying assumption regarding user preferences and experiment with two specific weighting schemes.

Recall from Equation 5.2, that the ranking model requires the following computation:

$$P_Q(t_i) = \lambda_0 P_Q^0(t_i) + \lambda_1 P_Q^1(t_i) + ... + \lambda_r P_Q^r(t_i)$$

where Q_j corresponds to the j^{th} relaxed query, and $P_Q^j(t_i)$ is the probability of triple t_i in the query LM of Q_j. Let q_i be the triple pattern in Q_j for which t_i is a match.

We now have the following two weighting schemes depending on how the results should be displayed to the user.

Incremental weighting. In this weighting scheme, we assume that the user is interested in seeing results *in order*. That is, all ranked matches of the original query first, followed by all ranked matches of the first relaxed query, then those of the second relaxed query, etc. That is, the results are presented "block-wise".

In order to ensure this, we need to set the λ_i's by examining the scores of the highest scoring and lowest scoring triple matches to a given triple pattern. For example, consider a query with a single triple pattern: ?x hasGenre Science_Fiction. Suppose a relaxation to this query is ?x hasGenre Fantasy. If we want to ensure that all matches of the original query are displayed before the first match of the relaxed query, we first examine the triple with the lowest score for the original query and the highest score for the relaxed query[2]. Let these scores be s_{low}^0 and s_{high}^1, respectively. We now need to ensure that $\lambda_0 * s_{low}^0 > \lambda_1 * s_{high}^1$.

Adaptive weighting. In this weighting scheme, we assume that the user is interested in seeing the results in a holistic manner. That is, a match to a lower ranked (relaxed) query can appear before a match to a higher ranked query. For example, as before, let the query ask for books of genre science fiction and a relaxation asks for books of genre fantasy. The assumption now is that the user would rather see a "famous" book of genre fantasy, rather than an obscure book of genre science fiction. And so, a "mixing" of results is allowed.

[2]Note that although not explained in this paper, the score has a direct correlation with the probability of the triple in the query model.

Let α_i be the score of the relaxed triple pattern q_i. That is, α_i is the sum of scores of each individual entity/relation relaxations in q_i. Now the weight λ_i for $P_Q^j(t_i)$ is set as follows:

$$\lambda_i = \frac{1 - \alpha_i}{\Sigma_{j=0}^{r}(1 - \alpha_j)}$$

This weighting scheme basically gives higher weights to the matches of relaxed triple patterns which are closer to their corresponding original triple patterns. However, matches for a lower ranked query with sufficiently high scores can be ranked above matches for a higher ranked query.

Note that both incremental as well as adaptive weighting are only two ways in which we can present results to the user. Additional schemes can include a mixture of both schemes for instance, or any other variations. Our ranking model is general enough and can support any number of such fine-grained result presentation schemes.

6 Experimental Evaluation

We evaluated the effectiveness of our relaxation techniques in 2 experiments. The first one evaluated the quality of individual entity and relation relaxations. It also evaluated the quality of the relaxed queries overall. The second experiment evaluated the quality of the final query results obtained from both original and relaxed queries. The complete set of evaluation queries used, relevance assessments collected as well as an online demonstration of our system can be found at `http://www.mpii.de/~elbass/demo/demo.html`.

6.1 Setup

All experiments were conducted over two datasets using the Amazon Mechanical Turk service[1]. The first dataset was derived from the LibaryThing community, which is an online catalog and forum about books. The second dataset was derived from a subset of the Internet Movie Database (IMDB). The data from both sources was automatically parsed and converted into RDF triples. In addition, each triple was also augmented with keywords derived from the data source it was extracted from. In particular, for the IMDB dataset, all the terms in the plots, tag-lines and keywords fields were extracted, stemmed and stored with each triple. For the LibraryThing dataset, since we did not have enough textual information about the entities present, we retrieved the books' Amazon descriptions and the authors' Wikipedia pages and used them as textual context for the triples. Finally, to estimate witness counts for the triples, we relied on the Web corpus. We issued queries to a major search engine, with the subject and object of the triple as keywords, and set the witness count to the number of hits returned. The witness counts for the triple-keyword pairs were estimated in a similar fashion. Table 6.1 gives an overview of the datasets.

Due to the lack of an RDF query benchmark, we constructed *40* evaluation queries for each dataset and converted them into structured triple-

[1] `http://aws.amazon.com/mturk/`

#entities	Example entity types	#triples	Example relations
LibraryThing Dataset			
48,000	book, author user, tag	700,000	wrote, hasFriend hasTag, type
IMDB Dataset			
59,000	movie, actor director, producer, country , language	600,000	actedIn, directed won, marriedTo, produced, hasGenre

Table 6.1: Overview of the datasets

LibraryThing	
Information need	**Query** Q
A Science-Fiction book made into a Film	?b type Science_Fiction; ?b hasTag Film
A children's writer who wrote a book that won the Booker Prize	?w type Children_Writer; ?w wrote ?b; ?b hasTag Booker
A book with tag British Literature and whose author won a Nobel prize	?w wrote ?b{*nobel prize*}; ?b hasTag British_Literature
A book with tag Film and has something to do with a civil war	?w wrote ?b{*civil war*}; ?b hasTag Film
IMDB	
Information need	**Query** Q
A movie with genre Comedy that won the Academy Award	?m hasGenre Comedy; ?m won Academy_Award
An actor from New York that won the Academy Award for Best Actor	?a won Academy_Award_for_Best_Actor; ?a originatesFrom New_York
A director that won the Academy Award for Best Director and directed a movie based on a true story	?d directed ?m{*true story*}; ?d won Academy_Award_for_Best_Director
A singer that acted in a movie about school friends	?a actedIn ?m{*school friends*}; ?a type singer

Table 6.2: Subset of the evaluation queries and the triple-pattern queries corresponding to the information need

pattern queries. In addition, we constructed *15* keyword-augmented queries for each dataset, where one or more triple-pattern was augmented with one

or more keyword. The number of triple patterns in the constructed queries ranged from 1 to 4. Some example queries are shown in Table 6.2.

6.2 Quality of Relaxations

To evaluate the quality of individual entity and relation relaxations, we extracted all unique entities and relations occurring in all evaluation queries. The total numbers of entities and relations are given in Table 6.3. For each entity, the top-5 relaxations were retrieved, excluding the variable relaxation. We presented the entity and each relaxation to 6 evaluators and asked them to assess how closely related the two are on a 3-point scale: 2 corresponding to "closely related", 1 corresponding to "related" and 0 corresponding to "unrelated". The same was done for each relation.

To evaluate the quality of relaxed queries overall, we generated the top-5 relaxed queries for each evaluation query. The relaxed queries were ranked in ascending order of their scores, which were computed as described in Section 5.1. We asked 6 evaluators to assess how close a relaxed query is to the original one on a 4-point scale: 3 corresponding to "very-close", 2 to "close", 1 to "not so close" and 0 corresponding to "unrelated".

Table 6.3 shows the results obtained for entity, relation and query relaxations. For a given entity, the average rating for each relaxation was first computed and then this rating was averaged over the top-5 relaxations for that entity. A similar computation was performed for relations and query relaxations. The second row shows the average rating over all relaxed entities, relations and queries. The third row shows the *Pearson correlation* between the average rating and the JS-divergence. We achieved a strong *negative* correlation for all relaxations which shows that the smaller the score of the relaxation (closer the relaxation is to the original), the higher the rating assigned by the evaluators. The fourth row shows the average rating for the top relaxation.

The fifth and sixth rows in Table 6.3 show the average rating for relaxations that ranked above and below the variable relaxation respectively. Recall that, for each entity or relation, a possible entry in the relaxation candidate-list is a variable as described in Section 4. For those relaxations that ranked above a variable (i.e., whose JS-divergence is less than that of a variable), the average rating was more than 1.29 for both entities and relations, indicating how close these relaxations are to the original entity or relation. For those relaxations that ranked below a variable, the average rating was less than 1.1 for entities and 0.8 for relations. This shows that the evaluators, in effect, agreed with the ranking of the variable relaxation.

Row	Metric	Entities (3-pt scale)	Relations (3-pt scale)	Queries (4-pt scale)
1	**No. of items**	87	15	80
2	**Avg. rating**	1.228	0.863	1.89
3	**Correlation**	-0.251	-0.431	-0.119
4	**Avg. rating for top relaxation**	1.323	1.058	1.94
5	**Avg. rating above variable**	*1.295*	*1.292*	-
6	**Avg. rating below variable**	1.007	0.781	-

Table 6.3: Results for entity, relation and query relaxations

6.3 Quality of Query Results

We compared our relaxation framework, with its two weighting scheme variants, *Adaptive* and *Incremental* (see Section 5.3), against a baseline approach outlined in [33]. The latter simply replaces a constant in the original query with a variable to generate a relaxed query. The weight of the relaxed triple pattern is determined based on the number of constants replaced. For all 3 methods, we set the weight of an original triple pattern to the same value, to ensure that exact matches would rank on top, and thus make the differences rely solely on the quality and weights of the relaxations.

We pooled the top-10 results from all 3 approaches and presented them to 6 evaluators in no particular order. The evaluators were required to assess the results on a 4 point scale: 3 corresponding to "highly relevant", 2 corresponding to "relevant", 1 corresponding to "somewhat relevant", and 0 corresponding to "irrelevant". To measure the ranking quality of each technique, we used the Discounted Cumulative Gain (DCG) [51], which is a measure that takes into consideration the rank of relevant results and allows the incorporation of different relevance levels. DCG is defined as follows

$$DCG(i) = \begin{cases} G(1) & \text{if i = 1} \\ DCG(i-1) + G(i)/log(i) & \text{otherwise} \end{cases}$$

where i is the rank of the result within the result set, and $G(i)$ is the relevance

level of the result. We set $G(i)$ to a value between 0 and 3 depending on the evaluator's assessment.

For each result, we averaged the ratings given by all evaluators and used this as the relevance level for the result. Dividing the obtained DCG by the DCG of the ideal ranking we obtained a *Normalized DCG (NDCG)* which accounts for the variance in performance among the weighting schemes.

The results of our user evaluation are shown in Tables 6.4 and 6.5. The reported NDCG values were averaged over all evaluation queries. Both variations of our framework (the first two columns) *significantly* outperformed the baseline approach, with a one-tailed paired t-test (p-value ≤ 0.01). Furthermore, we computed the average rating over all results for each technique as shown in the second and fourth rows (Avg. Rating).

For purely-structured queries (Table 6.4), the Adaptive approach had over 8% improvement in NDCG over the baseline for Librarything and over 5% for IMDB. The Incremental approach had improvements over 15% for Librarything and 7% for IMDB. For keyword-augmented queries (Table 6.5), the improvement is much more evident. The Adaptive approach outperformed the baseline one with over 33% gain in NDCG for Librarything and 21% for IMDB. The Incremental approach had improvements in NDCG of over 13% for Librarything and over 8% for IMDB.

Librarything			
	Adaptive	**Incremental**	**Baseline**
NDCG	**0.868**	**0.920**	0.799
Avg. Rating	**2.062**	**2.192**	1.827
IMDB			
	Adaptive	**Incremental**	**Baseline**
NDCG	**0.880**	**0.900**	0.838
Avg. Rating	**1.874**	**1.928**	1.792

Table 6.4: Results for purely-structured queries

Note that in the case of keyword-augmented queries, the Adaptive approach was preferred by evaluators to the Incremental one because the keywords play a key role in determining the relevance of a result. Since the Adaptive scheme does not enforce a strict ordering of structure relaxations, it allows results which better match the keyword context (while matching a lower-ranked structure relaxation) to be ranked higher.

Finally, in Tables 6.6 and 6.7 we show two example evaluation queries and the top-3 results returned by each relaxation approach. Next to each result,

Librarything			
	Adaptive	**Incremental**	**Baseline**
NDCG	**0.757**	**0.640**	0.566
Avg. Rating	**1.985**	**1.349**	0.969
IMDB			
	Adaptive	**Incremental**	**Baseline**
NDCG	**0.841**	**0.755**	0.623
Avg. Rating	**1.602**	**1.464**	0.922

Table 6.5: Results for keyword-augmented queries

we show the average rating given by the evaluators. The relaxed constants are underlined.

The query in Table 6.6 asks for science fiction books that have tag Film. There is only *one* one such result which is ranked as the top result by all 3 approaches. Since the Adaptive approach ranks the whole set of approximate results, it allows for more diversity in terms of relaxations. And so, the Adaptive approach returns the more famous and iconic movies, Blade and Star_Wars as the top results compared to The_Last_Unicorn and The_Mists_Of_Avalon returned by the Incremental scheme.

The query in Table 6.7 shows an example of a keyword-augmented query which asks for books with the tag Film about a civil war. The top-3 results returned by the Adaptive approach are all books that were turned into movies. In addition, all of them are about civil wars. The results returned by the Incremental weighting were also books that were turned into movies, however only one of them is about a civil war (the first). For the baseline approach, where the tag Film in the second triple pattern was relaxed into a variable, the results returned are all books about civil wars, or wars in general, however none of them had a tag related to films.

Rank	Result	Rating
Q: ?b type Science_Fiction**;** ?b hasTag Film		
Adaptive		
1	Star_Trek_Insurrection type Science_Fiction; Star_Trek_Insurrection hasTag Film	2.50
2	Blade type Science_Fiction; Blade hasTag Movies	2.83
3	Star_Wars type Science_Fiction; Star_Wars hasTag Made_Into_Movie	2.00
Incremental		
1	Star_Trek_Insurrection type Science_Fiction; Star_Trek_Insurrection hasTag Film	2.50
2	The_Last_Unicorn type Science_Fiction; The_Last_Unicorn hasTag Made_Into_Movie/tv	2.50
3	The_Mists_of_Avalon type Science_Fiction; The_Mists_of_Avalon hasTag Made_Into_Movie/tv	2.17
Baseline		
1	Star_Trek_Insurrection type Science_Fiction; Star_Trek_Insurrection hasTag Film	2.50
2	Helter_Skelter type History; Helter_Skelter hasTag Film	0.83
3	Fear_and_Loathing_in_Las_Vegas type History; Fear_and_Loathing_in_Las_Vegas hasTag Film	1.83

Table 6.6: Top-ranked results for the example query "A science-fiction book that has tag Film"

Rank	Result	Rating
Q: ?w wrote ?b**{*civil war*};** ?b hasTag Film		
Adaptive		
1	Margaret_Mitchell wrote Gone_with_the_Wind ; Gone_with_the_Wind hasTag Made_into_a_Movie	2.57
2	Ernest_Hemingway wrote For_Whom_the_Bell_Tolls ; For_Whom_the_Bell_Tolls hasTag Made_into_Movie/tv	2.64
3	Charles_Frazier wrote Cold_Mountain ; Cold_Mountain hasTag Made_into_Movie	2.83
Incremental		
1	Ernest_Hemingway wrote For_Whom_the_Bell_Tolls ; For_Whom_the_Bell_Tolls hasTag Made_into_Movie/tv	2.64
2	Aldous_Huxley wrote Brave_New_World ; Brave_New_World hasTag Made_Into_Movie/tv	2.12
3	George_Orwell wrote Nineteen_Eighty-four ; Nineteen_Eighty-four hasTag Made_Into_Movie/tv	1.78
Baseline		
1	J_Michael_Straczynski wrote Civil_War ; Civil_War hasTag American	1.78
2	Ernest_Hemingway wrote For_Whom_the_Bell_Tolls ; For_Whom_the_Bell_Tolls hasTag Europe	2.36
3	Ernest_Hemingway wrote A_Farewell_to_Arms ; A_Farewell_to_Arms hasTag World_War_One	2.00

Table 6.7: Top-ranked results for the example query "A book that has tag Film and has something to do with a civil war"

7 Conclusion

We proposed a comprehensive and extensible framework for query relaxation for entity-relationship search. Our framework makes use of language models as its foundation and can incorporate a variety of information sources on entities and relations. We showed how to use an RDF knowledge base to generate high quality relaxations. Furthermore, we showed how different weighting schemes can be used to rank results. Finally, we showed the effectiveness of our techniques through a comprehensive user evaluation. We believe that our contributions are of great importance for an extended-SPARQL API that could underlie the emerging "Web-of-Data" applications such as Linking-Open-Data across heterogeneous RDF sources.

Bibliography

[1] Freebase.

[2] O. Alonso and H. Zaragoza. Exploiting semantic annotations in information retrieval: Esair '08. *SIGIR Forum*, 42(1), 2008.

[3] S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, M. Holstege, J. Melton, M. Rys, and J. Shanmugasundaram. Xquery and xpath full text 1.0, 2008.

[4] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman. Structure and content scoring for xml. In *VLDB*, 2005.

[5] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. Flexpath: Flexible structure and full-text querying for xml. In *SIGMOD*, 2004.

[6] S. Amer-Yahia, L. V. S. Lakshmanan, and C. Yu. Socialscope: Enabling information discovery on social content sites. In *CIDR*, 2009.

[7] S. Amer-Yahia and M. Lalmas. Xml search: languages, inex and scoring. *SIGMOD Record*, 35(4), 2006.

[8] K. Anyanwu, A. Maduko, and A. P. Sheth. Sparq2l: towards support for subgraph extraction queries in rdf databases. In *WWW*, 2007.

[9] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

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

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

[12] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[13] H. Bast, A. Chitea, F. Suchanek, and I. Weber. Ester: Efficient search on text, entities and relations. In *SIGIR*, 2007.

[14] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE*, 2002.

[15] B. Billerbeck and J. Zobel. When query expansion fails. In *SIGIR*, 2003.

[16] H. E. Blok, V. Mihajlovic, G. Ramírez, T. Westerveld, D. Hiemstra, and A. P. de Vries. The tijah xml information retrieval system. In *SIGIR*, 2006.

[17] M. J. Cafarella, C. Re, D. Suciu, and O. Etzioni. Structured querying of web text data: A technical challenge. In *CIDR*, 2007.

[18] S. Chakrabarti. Breaking through the syntax barrier: Searching with entities and relations. In *ECML*, 2004.

[19] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *WWW*, 2007.

[20] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. on Database Syst.*, 31(3), 2006.

[21] T. Cheng, X. Yan, and K. C.-C. Chang. Entityrank: Searching entities directly and holistically. In *VLDB*, 2007.

[22] N. Choi, I.-Y. Song, and H. Han. A survey on ontology mapping. *Sigmod Record*, 35(3), 2006.

[23] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD*, 1998.

[24] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Pearson Education, 2009.

[25] W. B. Croft and H. S. (Eds.). Special issue on database and information retrieval integration. *VLDB J.*, 17(1), 2008.

[26] W. B. Croft, D. Metzler, and T. Strohman. *Search Engines - Information Retrieval in Practice*. Addison-Wesley, 2009.

[27] A. de Vries et al. Overview of entity ranking track. In *INEX*, 2007.

[28] P. DeRose, X. Chai, B. J. Gao, W. Shen, A. Doan, P. Bohannon, and X. Zhu. Building community wikipedias: A machine-human partnership approach. In *ICDE*, 2008.

[29] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. In *VLDB*, 2008.

[30] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin. Finding top-k min-cost connected trees in databases. In *ICDE*, 2007.

[31] A. Doan, L. Gravano, R. Ramakrishnan, and S. V. (Editors). Special issue on managing information extraction. *ACM SIGMOD Record*, 37(4), 2008.

[32] P. Dolog, H. Stuckenschmidt, H. Wache, and J. Diederich. Relaxing rdf queries based on user and domain preferences. *Journal of Intell. Inf. Sys.*, 2008.

[33] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum. Language-model-based ranking for queries on RDF-graphs. In *CIKM*, 2009.

[34] S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching rdf graphs with SPARQL and keywords. *IEEE Data Engineering Bulletin*, 33(1), 2010.

[35] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12), 2008.

[36] R. Fagin et al. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[37] H. Fang and C. Zhai. Probabilistic models for expert finding. In *ECIR*, 2007.

[38] D. S. W. Fei Wu. Automatically refining the wikipedia infobox ontology. In *WWW*, 2008.

[39] Freebase: A social database about things you know and love. http://www.freebase.com.

[40] K. Golenberg, B. Kimelfeld, and Y. Sagiv. Keyword proximity search in complex data graphs. In *SIGMOD*, 2008.

[41] J. Graupmann, R. Schenkel, and G. Weikum. The spheresearch engine for unified ranked retrieval of heterogeneous xml and web documents. In *VLDB*, 2005.

[42] C. Gueret, E. Oren, S. Schlobach, and M. Schut. An evolutionary perspective on approximate rdf query answering. In *Proc. of the Intl. Conf. on Scalable Uncertainty Management*, 2008.

[43] H. He, H. Wang, J. Yang, and P. S. Yu. Blinks: ranked keyword searches on graphs. In *SIGMOD*, 2007.

[44] H. He, H. Wang, J. Yang, and P. S. Yu. Blinks: ranked keyword searches on graphs. In *SIGMOD Conference*, 2007.

[45] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, Enschede, 2001.

[46] D. Hiemstra. Statistical language models for intelligent XML retrieval. In *Intelligent Search on XML Data*, 2003.

[47] V. Hristidis, H. Hwang, and Y. Papakonstantinou. Authority-based keyword search in databases. *TODS*, 33(1), 2008.

[48] H. Huang, C. Liu, and X. Zhou. Computing relaxed answers on rdf databases. In *WISE*, 2008.

[49] C. Hurtado, A. Poulovassilis, and P. Wood. Query relaxation in rdf. *Journal on Data Semantics*, 2008.

[50] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4), 2008.

[51] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR*, 2000.

[52] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB*, 2005.

[53] G. Kasneci, M. Ramanath, M. Sozio, F. M. Suchanek, and G. Weikum. Star: Steiner tree approximation in relationship-graphs. In *ICDE*, 2009.

[54] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. In *ICDE*, 2008.

[55] B. Kimelfeld and Y. Sagiv. Efficiently enumerating results of keyword search over data graphs. *Inf. Syst.*, 33(4-5), 2008.

[56] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: Similarity measures and algorithms (tutorial). In *SIGMOD*, 2006.

[57] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, 2001.

[58] D. Lee. *Query Relaxation for XML Model*. PhD thesis, UCLA, 2002.

[59] G. Li, B. Ooi, J. Feng, J. Wang, and L. Zhou. Ease: an effective 3-in-1 keyword search method for unstructured, semistructured and structured data. In *SIGMOD*, 2008.

[60] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In *SIGMOD*, 2008.

[61] X. Liu and W. B. Croft. Statistical language modeling for information retrieval. In *Annual Review of Information Science and Technology 39*, 2004.

[62] P. Mika. Anatomy of a searchmonkey. Nodalities Magazine, www.talis.com/nodalities/pdf/nodalities_issue4.pdf, 2008.

[63] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Morgan & Claypool, 2010.

[64] T. Neumann and G. Weikum. RDF-3X: a RISC-style engine for RDF. *Proceedings of the VLDB Endowment*, 1(1):647–659, 2008.

[65] Z. Nie, Y. Ma, S. Shi, J.-R. Wen, and W. Ma. Web object retrieval. In *WWW*, 2007.

[66] Z. Nie, Y. Ma, S. Shi, J.-R. Wen, and W.-Y. Ma. Web object retrieval. In *WWW*, 2007.

[67] Z. Nie, J.-R. Wen, and W.-Y. Ma. Object-level vertical search. In *CIDR*, 2007.

[68] D. Petkova and W. Croft. Hierarchical language models for expert finding in enterprise corpora. *Int. J. on AI Tools*, 17(1), 2008.

[69] H. Raghavan, J. Allan, and A. McCallum. An exploration of entity models, collective classification and relation description. In *Proc. of LinkKDD*, 2004.

[70] W3c: Resource description framework (rdf). www.w3.org/RDF/.

[71] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 2(1), 2008.

[72] P. Serdyukov and D. Hiemstra. Modeling documents as mixtures of persons for expert finding. In *ECIR*, 2008.

[73] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM*, 1999.

[74] W3c: Sparql query language for rdf. www.w3.org/TR/rdf-sparql-query/.

[75] S. Staab and R. Studer. *Handbook on Ontologies (International Handbooks on Information Systems)*. SpringerVerlag, 2004.

[76] F. Suchanek, M. Sozio, and G. Weikum. A self-organizing framework for information extraction. In *WWW*, 2009.

[77] F. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW*, 2009.

[78] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. *J. Web Sem.*, 6(3), 2008.

[79] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *J. Web Sem.*, 6(3), 2008.

[80] S. Tata and G. M. Lohman. Sqak: doing more with keywords. In *SIGMOD*, 2008.

[81] M. Theobald et al. Efficient and self-tuning incremental query expansion for top-k query processing. In *SIGIR*, 2005.

[82] A. Trotman and B. Sigurbjörnsson. Narrowed extended xpath i (nexi). In *INEX*, 2004.

[83] D. Vallet and H. Zaragoza. Inferring the most important types of a query: a semantic approach. In *SIGIR*, 2008.

[84] D. Vallet and H. Zaragoza. Inferring the most important types of a query: a semantic approach. In *SIGIR*, 2008.

[85] G. Weikum, G. Kasneci, M. Ramanath, and F. Suchanek. Combining database and information-retrieval techniques for knowledge discovery. *Communications of the ACM*, 52, 2009.

[86] C. Zhai. Statistical language models for information retrieval: A critical review. *Foundations and Trends in IR*, 2(3), 2008.

[87] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2), 2004.

[88] C. Zhai and J. Lafferty. A risk minimization framework for information retrieval. *Inf. Process. Manage.*, 42(1), 2006.

[89] X. Zhou, J. Gaugaz, W.-T. Balke, and W. Nejdl. Query relaxation using malleable schemas. In *SIGMOD*, 2007.

[90] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*, 2009.

Below you find a list of the most recent research reports of the Max-Planck-Institut für Informatik. Most of them are accessible via WWW using the URL `http://www.mpi-inf.mpg.de/reports`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
– Library and Publications –
Campus E 1 4

D-66123 Saarbrücken

E-mail: `library@mpi-inf.mpg.de`

.

MPI-I-2008-5-003	G. de Melo, F. Suchanek, A. Pease	Integrating YAGO into the Suggested Upper Merged Ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single Phase Construction of Optimal DAG-structured QEPs
MPI-I-2008-5-001	F. Suchanek, G. Kasneci, M. Ramanath, M. Sozio, G. Weikum	STAR: Steiner Tree Approximation in Relationship-Graphs
MPI-I-2008-1-001	D. Ajwani	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for Finite Domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : A Large Ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A Time Machine for Text Search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: Searching and Ranking Knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global Stochastic Optimization for Robust and Accurate Human Motion Capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global Illumination using Photon Ray Splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU Marching Cubes on Shader Model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A Higher-Order Structure Tensor
MPI-I-2007-4-004	C. Stoll	A Volumetric Approach to Interactive Shape Editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A Nonlinear Viseme Model for Triphone-Based Speech Synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of Smooth Maps with Mean Value Coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered Stochastic Optimization for Object Recognition and Pose Estimation
MPI-I-2007-2-001	A. Podelski, S. Wagner	A Method and a Tool for Automatic Veriication of Region Stability for Hybrid Systems
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-Time Reordering in a Sweep-line Algorithm for Algebraic Curves Intersecting in a Common Point
MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - A Core of Semantic Knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A Neighborhood-Based Approach for Clustering of Linked Document Collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment Texture Editing in Monocular Video Sequences based on Color-Coded Printing Patterns

MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: Index-access Optimized Top-k Query Processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-Aware Global df Estimation in Distributed Information Retrieval Systems
MPI-I-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean Value Coordinates for Arbitrary Spherical Polygons and Polyhedra in R^3
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and Annealing Particle Filters: Mathematics and a Recipe for Applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture Modeling and Animation by Imitation
MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving Non-local Denoising of Static and Time-varying Range Data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced Dynamic Reflectometry for Relightable Free-Viewpoint Video
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven Laplacian Mesh Deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On Fast Construction of Spatial Hierarchies for Ray Tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A Framework for Natural Animation of Digitized Models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU Point List Generation through Histogram Pyramids
MPI-I-2006-4-001	A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel	Design and Evaluation of Backward Compatible High Dynamic Range Video Compression
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On Verifying Complex Properties using Symbolic Shape Analysis
MPI-I-2006-1-007	H. Bast, I. Weber, C.W. Mortensen	Output-Sensitive Autocompletion Search
MPI-I-2006-1-006	M. Kerber	Division-Free Computation of Subresultants Using Bezout Matrices
MPI-I-2006-1-005	A. Eigenwillig, L. Kettner, N. Wolpert	Snap Rounding of Bézier Curves
MPI-I-2006-1-004	S. Funke, S. Laue, R. Naujoks, L. Zvi	Power Assignment Problems in Wireless Communication
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated Retraining Methods for Document Classification and their Parameter Tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An Emperical Model for Heterogeneous Translucent Objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric Calibration of High Dynamic Range Cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A. Magnor, H. Seidel	Joint Motion and Reflectance Capture for Creating Relightable 3D Videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and Design of Discrete Normals and Curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse Meshing of Uncertain and Noisy Surface Scattered Data
MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2005-2-004	Y. Kazakov	A Framework of Refutational Theorem Proving for Saturation-Based Decision Procedures
MPI-I-2005-2-003	H.d. Nivelle	Using Resolution as a Decision Procedure
MPI-I-2005-2-002	P. Maier, W. Charatonik, L. Georgieva	Bounded Model Checking of Pointer Programs
MPI-I-2005-2-001	J. Hoffmann, C. Gomes, B. Selman	Bottleneck Behavior in CNF Formulas
MPI-I-2005-1-008	C. Gotsman, K. Kaligosi, K. Mehlhorn, D. Michail, E. Pyrga	Cycle Bases of Graphs and Sampled Manifolds
MPI-I-2005-1-007	I. Katriel, M. Kutz	A Faster Algorithm for Computing a Longest Common Increasing Subsequence
MPI-I-2005-1-003	S. Baswana, K. Telikepalli	Improved Algorithms for All-Pairs Approximate Shortest Paths in Weighted Graphs
MPI-I-2005-1-002	I. Katriel, M. Kutz, M. Skutella	Reachability Substitutes for Planar Digraphs