# Interface

**Keywords**

 ESTER: efficient search on text, entities, and relations

We present ESTER, a modular and highly ecient system for combined full-text and ontology search. ESTER builds on a query engine that supports two basic operations: prefix search and join. Both of these can be implemented very eciently with a compact index, yet in combination provide powerful querying capabilities. We show how ESTER can answer basic SPARQL graph- pattern queries on the ontology by reducing them to a small number of these two basic operations. ESTER further sup- ports a natural blend of such semantic queries with ordi- nary full-text queries. Moreover, the prefix search operation allows for a fully interactive and proactive user interface, which after every keystroke suggests to the user possible se- mantic interpretations of his or her query, and speculatively executes the most likely of these interpretations. As a proof of concept, we applied ESTER to the English Wikipedia, which contains about 3 million documents, com- bined with the recent YAGO ontology, which contains about 2.5 million facts. For a variety of complex queries, ESTER achieves worst-case [**query processing**](http://academic.research.microsoft.com/Keyword/33961/query-processing) times of a fraction of a second, on a single machine, with an index size of about 4 GB.

### [Efficient type-ahead search on relational data: a TASTIER approach](http://academic.research.microsoft.com/Publication/4951809/efficient-type-ahead-search-on-relational-data-a-tastier-approach)

Existing keyword-search systems in relational databases re- quire users to submit a complete query to compute answers. Often users feel “left in the dark” when,they have limited knowledge about the data, and have to use a try-and-see method,to modify queries and,find answers. In this pa- per we propose a novel approach to [**keyword search**](http://academic.research.microsoft.com/Keyword/21563/keyword-search) in the relational world, called Tastier .A Tastier system can bring instant gratification to users by supporting type-ahead search, which finds answers “on the fly” as the user types in query keywords. A main challenge is how,to achieve a high interactive speed for large amounts of data in multiple tables, so that a query can be answered efficiently within milliseconds. We propose efficient index structures and al- gorithms for finding relevant answers on-the-fly by joining tuples in the database. We devise a partition-based method to improve query performance by grouping relevant tuples and pruning irrelevant tuples efficiently

Expressive and flexible access to web-extracted data: a keyword-based structured query language

Formulating information needs with conventional structured query languages is difficult due to the sheer size of schema information available to the user. We address this challenge by proposing a new [**query language**](http://academic.research.microsoft.com/Keyword/33953/query-language) that blends [**keyword search**](http://academic.research.microsoft.com/Keyword/21563/keyword-search) with structured [**query processing**](http://academic.research.microsoft.com/Keyword/33961/query-processing) over large information graphs with rich semantics. Our formalism for structured queries based on keywords combines the flexibility of[**keyword search**](http://academic.research.microsoft.com/Keyword/21563/keyword-search) with the expressiveness of structures queries. We propose a solution to the resulting disambiguation problem caused by introducing keywords as primitives in a structured query language. We show how expressions in our proposed language can be rewritten using the vocabulary of the web-extracted KB, and how different possible rewritings can be ranked based on their syntactic relationship to the keywords in the query as well as their semantic coherence in the underlying KB. An extensive [**experimental study**](http://academic.research.microsoft.com/Keyword/13358/experimental-study) demonstrates the efficiency and effectiveness of our approach. Additionally, we show how our [**query language**](http://academic.research.microsoft.com/Keyword/33953/query-language) fits into QUICK, an end-to-end [**information system**](http://academic.research.microsoft.com/Keyword/19930/information-system) that integrates web-extracted data graphs with full-text search. In this system, the rewritten query describes an arbitrary topic of interest for which corresponding entities, and documents relevant to the entities, are efficiently retrieved.

**NL**

While NLI systems which have a good performance require a customisation

(such as in the case of ORAKEL [1]), several systems have been developed

for which the customisation is not mandatory (e.g., PANTO [19], Querix [10],

AquaLog [13]), QuestIO [5], NLP-Reduce [10]). However, as is reported in [13]

the customisation usually improves the recall.

Semantic search via XML fragments: a high-precision approach to IR

In some IR applications, it is desirable to adopt a high precision [**search strategy**](http://academic.research.microsoft.com/Keyword/36522/search-strategy) to return a small set of documents that are highly focused and relevant to the user's information need. With these applications in mind, we investigate [**semantic search**](http://academic.research.microsoft.com/Keyword/36916/semantic-search)using the XML Fragments [**query language**](http://academic.research.microsoft.com/Keyword/33953/query-language) on text corpora automatically pre-processed to encode [**semantic information**](http://academic.research.microsoft.com/Keyword/36890/semantic-information)useful for retrieval. We identify three XML Fragment operations that can be applied to a query to conceptualize, restrict, or relate terms in the query. We demonstrate how these operations can be used to address four different query-time semantic needs: to specify target information type, to disambiguate keywords, to specify search term context, or to relate select terms in the query. We demonstrate the effectiveness of our [**semantic search**](http://academic.research.microsoft.com/Keyword/36916/semantic-search) technology through a series of experiments using the two applications in which we embed this technology and show that it yields significant improvement in precision in the search results.

ORAKEL

using a software called FrameMapper, where the linguistic argument structures,

such as verbs or nouns with their arguments, are mapped to the relations in

the ontology. While

In the ORAKEL system, the main task of the person in charge of customizing the system is to create a

domain-specific lexicon mapping subcategorization frames to relations specified in the domain ontology.

The lexicon engineer is

essentially responsible for specifying how certain natural language expressions map to predicates in the

knowledge base. For this purpose, we have designed an interface FrameMapper (compare Fig. 1) with access

to the knowledge base, which supports the lexicon engineer in specifying by graphical means the mapping

from language to relational predicates defined in the knowledge base. The result of the interaction of the

knowledge engineer is a domain lexicon specific for the application in question. The process of domain

adaptation is described in detail in Section 4, while the graphical user interface of FrameMapper is

described in Section 5.

Besides the domain-specific lexicon, ORAKEL also relies on a general lexicon which specifies the semantics

of closed-class words such as prepositions, determiners, question pronouns, numbers, etc. The semantics of

these closed-class words are actually domain independent and specified with respect to elementary or foundational

categories as given by foundational ontologies. In our ORAKEL system, we rely on the foundational

ontology DOLCE [37], which provides fundamental categories such as physical object, agentive physical object,

etc. as well as predicates and relations related to time and space.

**Both type of lexica are in fact a lexicalized grammar**

**which is used by ORAKEL for parsing but also for constructing the semantics of input questions.**

ORAKEL is an ontologybased

natural language system in two senses. First, the ontology for a certain knowledge base is used to guide

the lexicon construction process. On the one hand, parts of the lexicon are automatically generated from the

underlying ontology. But most importantly, on the other hand, the ontology is at the core of the whole lexicon

acquisition process in ORAKEL, which is performed by the lexicon engineer to adapt the system to some

domain and a particular knowledge base. Second, ORAKEL is ontology-based in the sense that it is a natural

language interface which relies on deduction to answer a user’s query.

As ORAKEL relies on a well-defined deduction process to answer a query, an important requirement is

that the user’s question is translated into logical form, in particular into a query which can be evaluated by

the underlying inference engine.

The challenge for natural language interfaces is thus the domain-specific interpretation of the user’s question in terms of relations and concepts defined in the schema or ontology of the knowledge base. Thus, parsers which create a generic logical form for a given input sentence will clearly not suffice for this purpose. The challenge is to construct a logical query consisting of domain-specific predicates which can be evaluated with respect to the knowledge base, returning the correct answer as a deduction process. Therefore, it is crucial that a natural language interface is adapted to every different knowledge base it is applied to.

These approaches exploit external lexical knowledge, for example in the form of lexical databases such as

WordNet [26], to account for syntactic variants. This is for example the case of the PRECISE [44] and AquaLog

[35] systems, which essentially rely on lexical matches to determine to which entities in the knowledge

base the words in the query refer to. At first sight, these approaches seem superior to an approach as presented

in this paper in which a lexicon needs to be explicitly created. Nevertheless, such approaches strongly depend

on the quality and coverage of the lexical resources used. Recent work by ourselves [19], in which an approach

based on lexical matching is explored, has in fact shown that one can rely less on lexical matching the more technical the domains get

In fact, we can not expect to have the complete lexical knowledge necessary for very

technical domains in general resources such as WordNet. Manually engineering a lexicon as in the ORAKEL

system described in this article certainly represents a considerable effort, but it allows to directly control the

quality and coverage of the lexicon for the specific application as the lexicon is represented declaratively and

can be directly updated.

**A Robust Ontology-Based Method for Translating Natural Language Queries to Conceptual Graphs**

1. Recognizing specified entities: this step recognizes entities specified by names in

a query. For instance, in the query “What is the capital of Mongolia?”, Mongolia

is a specified entity.

2. Recognizing unspecified entities: this step recognizes entities represented by

only words expressing entity types. For instance, in the example query “How

many counties are in Indiana?”, “counties” represents unspecified entities of the

type COUNTY.

3. Extracting relational phrases: this step finds out the phrases that represent relations

between the entities in a query. For example, in the query “What state is

Niagara Falls located in?”, “located in” is a phrase representing a relation between

Niagara Falls and a state, which is the queried entity

4. Determining the type of queried entities: this step determines the type of unknown

entities represented by interrogative words such as What or Which. For

example, in the query “What is WWE short for?”, the relation word “short for”

corresponds to the relation type HASALIAS, which requires the range entity type

ALIAS for the queried entity.

5. Unifying identical entities: this step groups the occurrences of the same entity

into one. For example, in the query “Who is the president of Bolivia?” there are

two identical entities represented by Who and “the president” to be grouped.

6. Discovering implicit relations: this step adds in relations that are not explicitly

expressed by words in a query. For example, in the query “What county is Modesto,

California in?”, there is an implicit relation between Modesto and California,

meaning the former is a sub-region of the latter.

7. Determining the types of relations: this step maps the extracted relational

phrases and discovered implicit relations to the appropriate relation types in the

ontology of discourse. For instance, in the example query “When was Microsoft

established?”, ESTABLISHMENTDATE is the suitable relation type for the relational

word “established” in this query about time.

8. Removing improper relations: this step checks and removes improper relations

constructed between entities in the previous steps. For example, in the query “What

city in Florida is Sea World in?”, there are three entities represented by “city”,

“Florida”, and “Sea World”. Without strict syntactic parsing, relations of the type

LOCATEDIN may connect both the first two entities and the third one. However, that

relation between “Florida” and “Sea World” is redundant in this case.

9. Constructing the final CG: this final step produces the CG corresponding to a

query with respect to the ontology and KB of discourse.

**USE GATE for many of these tasks**

**natural language query interface to structured information.**

QuestIO (Question-based In-

terface to Ontologies), which translates a Natural Language (NL) or a keyword-

based question into SPARQL, and returns the answer to the user after executing

the formal query against an ontology.

Although this approach uses very shallow

NLP, it is quite e\_cient for very small and domain-speci\_c ontologies. Also, it

performs quite well for the set of ill-formed and grammatically incorrect ques-

tions [5]. However, the trade-o  
 is that many grammatically correct questions

which do require more deep analysis would remain unanswered, or partially an-

swered. For example, if the question is What is the largest city in Nevada?,

QuestIO would be able to list cities in Nevada, but it would ignore the word

largest which is in this case crucial to deliver semantic meaning

**Natural Language Interfaces to Ontologies:**

**FREyA**

We present

our system FREyA, which combines syntactic parsing with the knowledge

encoded in ontologies in order to reduce the customisation e\_ort. If

the system fails to automatically derive an answer, it will generate clari

\_cation dialogs for the user. The user's selections are saved and used for training the system in order to improve its performance over time.

Our system is similar to Querix

in many aspects, with the main di  
erence that the primary goal of the dialog

in our system is not only to resolve ambiguities, but also to map question terms

to the relevant ontology concepts. Therefore, our system does not rely on the

vocabulary of the ontology, but tries to align it with that of the user.

Identi\_cation and veri\_cation of ontology concepts 🡪 USING user clarification and REeinforment learning

{ Generating SPARQL

{ Identi\_cation of the answer type and presenting the results to the user

## Relaxation

Query Relaxation for Entity-Relationship Search

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 triplepattern

queries 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 on any ranking function for

structured queries over RDF-data. 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.

# Facets

Reconciling expressive querying and exploratory search,

Query-based Faceted Search (QFS), is to de-

fine a semantic search that is (1) easy to use, (2) safe, and (3) expressive. Ease-

of-use and safeness are retained from existing faceted search systems by keeping

their general principles, as well as the visual aspect of their interface. Expres-

siveness is obtained by representing the current selection by a query rather than

by a set of items, and by representing navigation links by query transforma-

tions rather than by set operations (e.g., intersection, crossing). In this way,

the expressiveness of faceted search is determined by the expressiveness of the

query language, rather than by the combinatorics of user interface controls. In

this paper, the query language, named LISQL, generalizes existing semantic

faceted search systems, and covers most features of SPARQL. The use of queries

for representing selections in faceted search has other benefits than navigation

expressiveness. The current query is an intensional description of the current

selection that complements its extensional description (list of items). It informs

users in a precise and concise way about their exact position in the navigation

space. It can easily be copied and pasted, stored and retrieved later.

# Content

 An experimental study of the impact of information extraction accuracy on semantic search performance

Researchers have shown that various [**natural language processing**](http://academic.research.microsoft.com/Keyword/27138/natural-language-processing) techniques can be used in [**document analysis**](http://academic.research.microsoft.com/Keyword/10820/document-analysis) to impact search performance. For the most part, they examined how an analysis system with certain performance characteristics can be leveraged to improve document and/or passage search results. We have previously shown that semantic queries which utilize [**named entity**](http://academic.research.microsoft.com/Keyword/27043/named-entity) and relation information extracted from the corpus can lead to significant improvement in search performance. In this paper, we extend our previous efforts and examine how search performance degrades in the face of imperfect [**named entity**](http://academic.research.microsoft.com/Keyword/27043/named-entity) and relation extraction. Our study was carried out by developing [**gold standard**](http://academic.research.microsoft.com/Keyword/56035/gold-standard) annotated corpora and applying different error models to the [**gold standard**](http://academic.research.microsoft.com/Keyword/56035/gold-standard) annotations to simulate errors made by automatic recognizers. We identify automatic recognizer characteristics that make them more amenable to our search tasks, show that recognizer recall has more significant impact on [**semantic search**](http://academic.research.microsoft.com/Keyword/36916/semantic-search) performance than its precision, and demonstrate that significant improvement in both MAP and Exact Precision scores can be achieved by adopting automatic [**named entity**](http://academic.research.microsoft.com/Keyword/27043/named-entity)and relation recognizers with near state-of-the-art performance.

**Using WordNet to disambiguate word senses for text retrieval**

This paper describes an [**automatic indexing**](http://academic.research.microsoft.com/Keyword/2625/automatic-indexing) procedure that uses the “IS-A” relations contained within WordNet and the set of nouns contained in a text to select a sense for each plysemous noun in the text. The result of the indexing procedure is a vector in which some of the terms represent word senses instead of word stems. Retrieval experiments comparing the effectivenss of these sense-based vectors vs. stem-based vectors show the stem-based vectors to be superior overall, although the sense-based vectors do improve the performance of some queries. The overall degradation is due in large part to the difficulty of disambiguating senses in short query statements. An analysis of these results suggests two conclusions: the IS-A links define a generalization/specialization hierarchy that is not sufficient to reliably select the correct sense of a noun from the set of fine sense distinctions in WordNet; and missing correct matches because of incorrect sense resolution has a much more deleterious effect on retrieval performance than does making spurious matches.

Concept Based Retrieval in Classical IR Svstemsl

The character by character approach to IR has been

abandoned in favor of an approach based on the meaning of both the queries and

the texts containing the information to be sought. The concept space, locally derived

from a thesaurus, is used to represent a query as well as documents retrieved in

atomic concept units. Dependencies between the search terms are taken into

account. The meanings of the query and the retrieved documents (results of

Elementary Logical Conjuncts (ELGs) ) are compared.

**ConceptBased**

**Biomedical Text Retrieval**

Information Retrieval (IR) in the context of biomedical

databases has the following three major problems [3]: the

frequent use of (possibly non-standardized) acronyms, the

presence of homonyms (the same word referring to two or

more different entities) and synonyms (two or more words

referring to the same entity). How to deal with an abundant

number of lexical variants of the same term is a challenging

task in biomedical IR. In biomedical domain, using one or

two keywords to retrieve the corresponding concept is usually

not enough. Thus, concept-oriented retrieval techniques

have drawn much attention recently from researchers since

traditional keyword-based retrieval fails to meet the requirements

for both precision and recall.

For the concept detection, a set of concepts will be extracted

from the query first. This process can be described

as follows: (1) extract the full name of a biomedical entity

and its abbreviation by using BioNLP abbreviation extraction

function [4]; (2) take out all the stop words; (3) generate

a concept 1.

**Ontology as a Search-Tool: A Study of Real Users’ Query**

**Formulation With and Without Conceptual Support**

This study examines 16 real users’ use of an ontology as a search

tool. The users’ queries constructed with the help of a Concept-based Information

Retrieval Interface (CIRI) were compared to queries created independently

based on the same search task description. Also the effectiveness of the CIRI

queries was compared to the users’ unaided queries. The simulated search task

method was used to make the searching situations as close to real as possible.

Due to CIRI’s query expansion feature the number of search terms was remarkably

higher in ontology queries than in Direct interface queries. The search

results were evaluated with generalised precision and generalised relative recall

as well as precision based on personal assessments. The Direct interface queries

performed better in all methods of comparison.

Conceptual language models for domain-specific retrieval

In this paper we show that a concept language can be effectively used to improve full-text retrieval. In a two step process

that extends on relevance feedback and uses a conceptual representation as a pivot language we improve the query model

representing the information need of the user.

In the first step, the textual information need is translated into a conceptual representation. In a process we call conceptual

query modeling, feedback documents from an initial retrieval run are used for obtaining a conceptual query model. This

model represents the user’s information need at a different, higher conceptual level than the original query. The intuition

behind this step is that this conceptual representation gives an unambiguous representation of the information need. In contrast

to traditional textual relevance feedback, where the query refinement is biased towards terms occurring in the initial

query, this intermediate conceptual representation is less dependent on the original query words. On its own, this explicit

conceptual representation can be used to aid retrieval, for example by suggesting relevant concepts to the user (Meij & de

Rijke, 2007), or by matching it to a conceptual representation of the documents (Trieschnigg et al., 2009). In the second step,

however, we translate the conceptual query model back into a contribution to the textual query model. We hypothesize that,

since the textual representation of documents is more detailed than its conceptual representation,1 retrieving information

with a textual query representation translated from a conceptual form, results in better retrieval performance than strictly

matching with only concepts. Essential to these two translation steps is the estimation of a query model, both for terms and

for concepts. The textual query should be captured by a small set of specific concepts and the conceptual query model should

be translated to specific textual terms. To achieve this, we employ an expectation maximization algorithm inspired by parsimonious

language models (Hiemstra, Robertson, & Zaragoza, 2004).

Concept Search

In this paper we present a novel approach, called Concept

Search, which extends syntactic search, i.e., search based on the computation

of string similarity between words, with semantic search, i.e.,

search based on the computation of semantic relations between concepts.

The key idea of Concept Search is to operate on complex concepts and to

maximally exploit the semantic information available, reducing to syntactic

search only when necessary, i.e., when no semantic information is

available. The experimental results show that Concept Search performs

at least as well as syntactic search, improving the quality of results as a

function of the amount of available semantics

Problems tackled:

**Polysemy.**

**Synonymy.**

**Complex concepts.** Syntactic search engines fall short in taking into account

complex concepts formed by natural language phrases and in discriminating

among them. Consider, for instance, document *D*2 (in Figure 1). This document

describes two concepts: *a laptop computer* and *a coffee table*.

**Related concepts.** Syntactic search does not take into account concepts which

are semantically related to the query concepts. For instance, a user looking for

*carnivores* might not only be interested in documents which talk about carnivores

but also in those which talk about the various kinds of carnivores such as *dogs*

and *cats*.

*C-Search* reuses retrieval models (*Model* )

and data structures (*Data Structure*) of syntactic search with the only difference

in that now words (*W*) are substituted with concepts (*C*) and syntactic matching

of words (*WMatch*) is extended to semantic matching of concepts (*SMatch*).

Searching documents, in *C-Search*, is implemented using complex concepts expressed

in a propositional Description Logic (DL) language (i.e., a DL language

without roles). Complex concepts are computed by analyzing meaning of the

words and phrases.

The

conversion of words into concepts is performed as follows. First, we look up and

enumerate all meanings of the word in WordNet. Next, we perform word sense

filtering, i.e., we discard word senses which are not relevant in the given context.

In order to do this, we follow the approach presented in [22], which exploits POS

tagging information and WordNet lexical database for disambiguation of words

in short noun phrases

Complex concepts are computed by extracting phrases and by analyzing their

meaning. Noun phrases are translated into the logical conjunction of atomic

concepts corresponding to the words. For instance, the noun phrase *A little dog*

represents a concept, whose extension is the set of all dogs of a small size.

**Ontology-Based Query Expansion with Latently**

**Related Named Entities for Semantic Text Search**

However, the content of a document or a

query is mainly defined by both keywords and named entities occurring in

it. Named entities have ontological features, namely, their aliases, classes,

and identifiers, which are hidden from their textual appearance. Besides, the

meaning of a query may imply latent named entities that are related to the

apparent ones in the query. We propose an ontology-based generalized vector

space model to semantic text search. It exploits ontological features of

named entities and their latently related ones to reveal the semantics of

documents and queries. We also propose a framework to combine different

ontologies to take their complementary advantages for semantic annotation

and searching. Experiments on a benchmark dataset show better search

quality of our model to other ones.

**A Generalized Vector Space Model**

In [4], a generalized VSM was proposed so that a document or a query was represented

by a vector over a space of generalized terms each of which was either a

keyword or an NE triple. As usual, similarity of a document and a query was defined

by the cosine of the angle between their representing vectors. The work implemented

the model by developing a platform called S-Lucene modified from

Lucene4. The system automatically processed documents for NE-keyword-based

searching in the following steps:

1. Removing stop-words in the documents.

2. Recognizing and annotating named entities in the documents using KIM5.

3. Extending the documents with implied NE triples. That is, for each entity

named *n* possibly with class *c* and identifier *id* in the document, the triples

(*n*/\*/\*), (*\**/*c*/\*), (*n*/*c*/\*), (*alias*(*n*)/\*/\*), (*\**/*super*(*c*)/\*), (*n*/*super*(*c*)/\*),

(*alias*(*n*)/*c*/\*), (*alias*(*n*)/ *super*(*c*)/\*), and (*\**/\*/*id*) were added for the

document.

4. Indexing NE triples and keywords by S-Lucene.

Here *alias*(*n*) and *super*(*c*) respectively denote any alias of *n* and any super class

of *c* in the ontology and knowledge base of discourse.

A query was also automatically processed in the following steps:

1. Removing stop-words in the query.

2. Recognizing and annotating named entities in the query.

3. Representing each recognized entity named *n* possibly with class *c* and

identifier *id* by the most specific and available triple among (*n*/\*/\*), (*\**/*c*/\*),

(*n*/*c*/\*), and (\*/\*/*id*).

**Query Expansion**

Recognizing Relation Phrases: Relation phrases are prepositions, verbs,

and other phrases representing relations, such as *in*, *on*, *of*, *has*, *is*, *are*, *live*

*in*, *located in*, *was actress in*, *is author of*, *was born*. We implement relation

phrase recognition using the ANNIE tool of GATE ([8]).

2. Determining Relations: Each relation phrase recognized in step 1 is

mapped to a corresponding one in Ontology\_2 by a manually built dictionary.

For example, “*was actress in*” is mapped to *actedIn*, “*is author of*” is

mapped to *wrote*, and “*nationality is*” is mapped to *isCitizenOf*.

3. Recognizing Entities: Entity recognition is implemented by OCAT (Ontology-

based Corpus Annotation Tool) of GATE.

4. Determining Related Entities: Each entity that has a relation determined in

step 2 with an entity recognized in step 3 is added to the query. In the scope

of this paper, we consider to expand only queries having one relation each.

However, the method can be applied straightforwardly to queries with

more than one relation.

Our view of the semantic retrieval problem is very close to the latest proposals in KIM [20],

[28]. While **KIM** focuses on automatic population and annotation of documents, our work focuses

on the ranking algorithms for semantic search. Along with TAP [16], KIM is one of the

most complete proposals reported to date, to our knowledge, for building high-quality KBs, and

automatically annotating document collections at a large scale. In their latest account of progress

[20] a ranking model for retrieval is hinted at but has not been developed in detail and evaluated. In fact, KIM relies on a keyword-based IR engine for this purpose (indexing, retrieval and ranking).

Our work complements KIM with a ranking al gorithm specifically designed for an ontology-

based retrieval model, using a semantic indexing scheme based on annotation weighting

techniques.

**TAP** [16] presents a view of the Semantic Web where documents and concepts are nodes

alike in a semantic network, whereby the separation of contents and metadata is not as explicit as

we propose here. The two main problems addressed by TAP are a) the development of a distributed

query infrastructure for ontology data in the Semantic Web, and b) the presentation of query

execution results, augmenting query answers with data from surrounding nodes. These issues are

complementary to the ones addressed in this paper. However the expressive power of the TAP

query language is fairly limited compared to ontology query languages such as SPARQL [29],

RDQL [32], etc. The supported search capability is limited to keyword search within the “title

properties” of instances, and no ranking is provided.

The **Corese** system [17] is an ontology-based search engine for

the Semantic Web, which retrieves Web resources that are annotated

in RDF(S) via a query language based on RDF(S). It is the system

that is perhaps closest in spirit to our approach. In a first

phase, Corese translates annotations into conceptual graphs, it

then applies proper inference rules to augment the information

contained in the graphs, and finally evaluates a user query by projecting

it onto the annotation graphs. The Corese query language is

based on RDF, and it allows variables and operators.

**SHOE** [33] is one of the first attempts to semantically query the

Web. It provides the following: a tool for annotating Web pages, allowing users to add SHOE markup to a page by selecting ontologies,

classes, and properties from a list; a Web crawler, which

searches for Web pages with SHOE markup and stores the information

in a knowledge base (KB); an inference engine, which provides

new markups by means of inference rules (basically, Horn clauses);

and several query tools, which allow users to pose structured queries

against an ontology. One of the query tools allows users to

draw a graph in which nodes represent constant or variable instances,

and arcs represent relations. To answer the query, the system

retrieves subgraphs matching the user graph. The SHOE search

tool allows users to pose queries by first choosing an ontology from

a drop-down list and next choosing classes and properties from another

list. Finally, the system builds a conjunctive query, issues the

query to the KB, and presents the results in a tabular form.

**Avatar Semantic Search: A Database Approach to**

**Information Retrieval**

We present ***Avatar Semantic Search***, a prototype search engine

that exploits annotations in the context of classical keyword search.

The process of annotations is accomplished offline by using highprecision

information extraction techniques to extract facts, concepts,

and relationships from text. These facts and concepts are

represented and indexed in a structured data store. At runtime, keyword

queries are interpreted in the context of these extracted facts

and converted into one or more precise queries over the structured

store. In this demonstration we describe the overall architecture of

the ***Avatar Semantic Search*** engine. We also demonstrate the superiority

of the AVATAR approach over traditional keyword search

engines using Enron email data set and a blog corpus.

**Extraction and Representation *Avatar Semantic Search*** uses the

publicly available UIMA framework [7, 13] to compose and

execute text analysis programs (called annotators) [10]. The

UIMA framework allows us to define a document processing

workflow consisting of a chain of annotators. Documents are

fed in at one end of the workflow and the resulting annotations

become available at the other end. The annotations produced

through this process are persisted in an annotation store. We

have implemented the annotation store as a thin layer on top

of a commercial DBMS. While the task of building annotators

and persisting annotations involves several challenges, the details

are not relevant for this demonstration proposal.

**Interpretation.** Our primary focus will be the task of bridging

the gap between the keyword queries that users submit and

the structured queries that model their intent. ***Avatar Semantic***

***Search*** uses the notion of *keyword query interpretation* – a process

by which end-user keyword queries are automatically converted

into one or more precise queries called *interpretations*.

Each interpretation, when executed over the annotation store, produces a set of documents as result (e.g., given the schema of

Figure 1, Query q1 is a possible interpretation of the keyword

query ‘tom phone’).

The translation index is used by the semantic optimizer to retrieve

the possible “meanings” of an individual keyword. Given a

keyword, the index returns the (i) concepts in the schema, such as

types and paths, that match the keyword (based on keywords provided

by domain experts for types and paths in the schema), and

(ii) paths where the keyword appears as a data value in the data set.

Figure 3 shows the result of probing the translation index with

the keyword ‘tom’. Notice that in addition to the four value matches,

the translation index also returns a “Keyword” match. The keyword

match indicates that by default, ‘tom’ can be treated a plain

keyword without attaching a specific semantic meaning. This keyword

match is implicitly generated every time the translation index

is queried.

*Enumeration:* The concept of *j*ustification is used to formally

define the class of queries over the annotation store that represent

valid interpretations of a keyword query. In the enumeration

step, user keywords are matched against the translation

index as shown in Figure 3. The resulting matches are

used to generate every possible *justified* interpretation of a

keyword query.

*Pruning:* As we show in the demonstration, even for

annotation stores with a small number of types, the set of

possible justified interpretations can be extremely large. For

instance, the query ‘tom phone’ generates over 40 interpretations

in our implementation over the Enron data set. The

pruning step uses domain specific constraints over the annotation

schema to eliminate several of these interpretations.

An example constraint over the schema in Figure 2 models

the fact that there is exactly one author for every email message.

The pruning step uses this constraint to discard interpretations

that involve multiple author annotations.

\_ *Ranking:* After pruning, every remaining interpretation is

potentially of interest to a user who submitted the original

keyword query. The goal of ranking is to generate an ordering

of these interpretations so that the more likely interpretations

are ranked higher. Such an ordering is achieved

using the notion of *s*emantic value of an interpretation. As

an example, one of the measures used to compute semantic

value is the number of keywords in an interpretation that are

mapped to concepts (treating keyword ‘phone’ as the concept

PhoneNumber).

## Matching

# ****Ρ-Queries: enabling querying for semantic associations on the semantic web****

This paper presents the notion of Semantic Associations as

complex relationships between resource entities. These

relationships capture both a connectivity of entities as well as

similarity of entities based on a specific notion of similarity called

ñ-isomorphism. It formalizes these notions for the RDF data

model, by introducing a notion of a Property Sequence as a type.

In the context of a graph model such as that for RDF, Semantic

Associations amount to specific certain graph signatures.

Specifically, they refer to sequences (i.e. directed paths) here

called Property Sequences, between entities, networks of Property

Sequences (i.e. undirected paths), or subgraphs of ñ-isomorphic

Property Sequences.

The ability to query about the existence of such relationships is

fundamental to tasks in analytical domains such as national

security and business intelligence, where tasks often focus on

finding complex yet meaningful and obscured relationships

between entities. However, support for such queries is lacking in

contemporary query systems, including those for RDF.

This paper discusses how querying for Semantic Associations

might be enabled on the Semantic Web, through the use of an

operator ñ. It also discusses two approaches for processing ñ-

queries on available persistent RDF stores and memory resident

RDF data graphs, thereby building on current RDF query

languages.

The framework described in this section provides a formal basis

for Semantic Associations. It builds on the formalization for the

RDF data model given in [40], by including a notion of a

Property Sequence. A Property Sequence allows us to

capture paths in the RDF model and forms the basis for

formalizing Semantic Associations as binary relations on Property

Sequences. Secondly, we some complex queries called ρ-queries

for querying about Semantic Associations.

A Property Sequence PS is a finite sequence of properties

[P1, P2, P3, … Pn] where Pi is a property defined in an RDF

Schema RSj of a schema set RSS. The interpretation of PS is

given by:

**STRATEGIES FOR PROCESSING**

ρ**-QUERIES**

**Using Structrure Index**

Phase 1 captures the query, i.e. the resources and context (i.e.

schema set). In the second stage, the resources are classified i.e.,

the classes that entities belong to, within the given context, are

identified. This involves a query to the data store layer, which

exploits the rdf:typeOf statements to answer the query. Much of

the processing is done in the third phase where potential paths

involving the entities in the query are discussed by querying a

PathGuide (a combination of index structures that stores

information about paths that exist between resources classes).

There are two kinds of paths that are kept in the PathGuide. The

first kind of path is that which is obvious from the schema. The

second kind is those paths that exist at the data level but are not

evident at the schema level. This is because of the fact that the

RDF data model allows multiple classifications of entities.

**Using Graph Algorithms**

graph traversals algorithms can be applied. In the case of ρ-

pathAssociation we can search for paths between entities, and in

the case of a ρ-joinAssociation we check if the two entities belong

in the same connected component. One issue with this approach is

that that trying to find all paths between entities could easily lead

to an exponential time algorithm

However, [52] provides

promising fast algorithms for solving path problems which may

be employed for such computations. In particular, it offers nearlinear

time algorithms for computing a path expression

representing the set of all paths between nodes in a graph.

**OntoSearch**

The procedure of the OntoSearch system in handling search

queries is highlighted in Figure 1. Similar to that of a traditional

search engine, a user submits queries consisting

of keywords to the system1, wherein the corresponding semantic

annotation is not required. OntoSearch then returns

an initial list of documents obtained with a keyword based

search method. Since the documents are pre-annotated with

the ontological information, we also obtain a set of the associated

concepts based on the documents retrieved. Using

these concepts as the seeds to our semantic network based

domain ontology, the spreading activation theory (Anderson

1983) process infers the concepts that are semantically related

to the initial concept set. Finally, the conceptual relevance

scores, in terms of the concept activations in the domain

ontology, are used to re-rank the documents before presentation

to the user. The detailed algorithms are presented

in the following sections.

# Ranking

 Ranking Complex Relationships on the Semantic Web

[**information retrieval**](http://academic.research.microsoft.com/Keyword/19900/information-retrieval) over [**semantic metadata**](http://academic.research.microsoft.com/Keyword/36899/semantic-metadata) extracted from the Web has received an increasing amount of interest in both industry and academia. In particular, discovering complex and meaningful relationships among this metadata is an interesting and challenging research topic. Just as ranking of documents is a critical component of today's search engines, the ranking of complex relationships will be an important component in tomorrow's[**Semantic Web**](http://academic.research.microsoft.com/Keyword/36922/semantic-web) analytics engines. Building upon our recent work on specifying and discovering complex relationships in RDF data, called Semantic Associations, we present a flexible ranking approach which can be used to identify more interesting and relevant relationships in the Semantic Web. Additionally, we demonstrate our ranking scheme's effectiveness through an[**empirical evaluation**](http://academic.research.microsoft.com/Keyword/12217/empirical-evaluation) over a real-world dataset.

Ranking Semantic Associations

Our goal for supporting users in finding interesting

semantic associations is to rank the results of a

query involving two entities — e1:Person and

e9:Person in Figure 1, for example. The results of

the query are semantic associations indicating the

different ways in which these two people are related.

Due to the small-world phenomenon of every

human being able to reach anyone else via a short

social sequence, myriad paths could conceivably

connect these two entities, so we must rank the

paths in order of relevance.

Semantic Metrics:

**Context.** Userspecify context in terms of classes, e.g region 1 “scientific publication” and “computer science researcher.” Assume the user has specified that region 2 contain the classes “country” and “state.” The resulting

regions, 1 and 2, will now refer to the computer

science research and geographic domains, respectively.

For the associations at the top of Figure 1,

we’ll give regions 1 and 2 the weights 0.8 and 0.2,

respectively, which gives the bottom-most association

the highest rank because all of its entities

and relationships are in the region with the greatest

weight. The second-highest ranked association

would be the one at the top of the figure because

it includes an entity in region 1 and, unlike the

association in the middle, an entity in region 2.

**Subsumption**. Lower-ranked classes in an ontology’s

hierarchy are more specialized instances of

those higher up — that is, they convey more

detailed information and have more specific meanings.

**Trust.** Various entities and their relationships in a

semantic association originate from different

sources, some of which might be more trusted than

others. Thus, a user optionally assigns trust values

to extracted metadata depending on its source. For

the data set we used, we assigned default trust values

empirically.

**Statistical Metrics**

Statistical metrics are based on aspects of an

ontology, such as the number and connectivity of

entities and relationships.

**Rarity**. Given the increasing size of Semantic Web

test beds, many relationships and entities of the

same type exist. Although those that rarely occur

are sometimes more interesting,9 the opposite can

be true for other queries. In the context of money

laundering, for example, individuals often engage

in common-looking (not rare) transactions to

avoid detection.

**Popularity**. When we investigate the entities in an

association, we see that some of them have more

relationships than others. Somewhat similar to

Kleinberg’s Web page ranking algorithm11 and

Google’s PageRank12 algorithm, our approach

considers the number of incoming and outgoing

relationships among entities — specifically, the

number of edges — to be its popularity. In some

queries, associations with highly popular entities

are more relevant because they’re the hotspots in

the knowledge base. In other queries, however,

the user might want to rank popular entities

lower. Associations with a small number of connections

can be interesting, for example.

**Association length.** In some queries, a user might

be interested in more direct (that is, shorter) associations.

An editor in chief, for example, might

want to look for a (potential) conflict of interest in

a peer-review setting.

**SemRank: ranking complex relationship search results on the semantic web**

In this paper, we present an approach that ranks results based on how predictable a result might be for users. It is based on a [**relevance model**](http://academic.research.microsoft.com/Keyword/35083/relevance-model) SemRank, which is a rich blend of semantic and information-theoretic techniques with heuristics that supports the novel idea of modulative searches, where users may vary their search modes to effect changes in the ordering of results depending on their need.

**Predictability of results**

There is some related

work [26] being done in the area of ranking Semantic Networks.

However, this approach suffers from the same limitation as most

ranking approaches which have a fixed ranking scheme that

imposes a single type of ordering on results. That is that the

same query made in different contexts and for different

purposes, still yields the same ordering.

In some cases, we may need to boost results that are

considered unpredictable whereas in the latter case we may need

to reverse that ordering.

In context of a relationship search however, queries do not contain a

description as such, they may just identify the entities of interest.

Consequently, a relevance model that is based on how good of a

match a document is to a query does not apply and the

development of novel techniques is necessary. One promising

approach to dealing with this problem is based on using metrics

that somehow measure the predictability of the result that is

being returned. For example, we may choose to rank highest in

an investigative or discovery search, results that are less

predictable while in a conventional search the reverse ordering

more desirable.

In this paper, we focus on ranking the results of complex

relationship searches on the Semantic Web. We pursue

an approach that is based on a modulative relevance

model SemRank, that can easily (using a sliding bar) be

modulated or adjusted via the query interface. In this

way, a user can easily vary their search mode from a

Conventional search mode to a Discovery search

mode based on their need.

**Information Gain and** **-Path Semantic**

**Associations**

Based on this we can build a model for measuring the

information content of a semantic association by considering the

occurrence of edge as an event and RDF properties as its

outcomes. We begin with defining the notion for a property and

then extend it to a sequence of properties of a path.

To illustrate these concepts, for the property “purchased”

connecting &r1 and &r2 in Figure 2 above, ROC1 = {Student,

Passenger}, ROC2 = {Book, Ticket} with θ =

{[[purchased]]+, [[bidsFor]]+}, so that the size of θ = | [[

purchased ]]^ | + | [[ acquired ]]^ | + | [[bidsFor]]^ |. If

there are 20, 40, 80 instances of the properties purchased,

acquired and bidsFor respectively, with a total of 1000

property instances in the description base, then the specificity of

purchased is 0.02 while its θ-specificity is 0.143.

Exploiting Locality of Wikipedia Links in Entity Ranking

The global score of an entity page is derived by combining three separate scores:

a linkrank score, a category score, and a full-text similarity score.

Our approach utilises the

known categories and the link structure of Wikipedia, and more importantly,

exploits link co-occurrences to improve the effectiveness of entity

ranking.

**LinkRank score.** The linkrank function calculates a score for a page, based

on the number of links to this page, from the first N pages returned by the

search engine in response to the query. The parameter N has been kept to a

relatively small value mainly for performance purposes, since Wikipedia pages

contain many links that would need to be extracted. We carried out experiments

with different values of the parameter N, by varying it between 5 and 100 with

a step of 5, and found that N=20 was a good compromise between performance

and discovering more potentially good entities.

* Takes locality of links into account

**Category similarity score.** To calculate the category similarity score, we use

a very basic similarity function that computes the ratio of common categories

between the set of categories associated with the target page cat(t) and the set

of the union of the categories associated with the entity examples cat(E):

Consider the example of the Euro page shown in Fig. 1. Let us assume that

the topic is “European countries where I can pay with Euros”, and France,

Germany and Spain are three entity examples. We see that the 15 countries that

are members of the Eurozone are all listed in the same paragraph with the three

entity examples. In fact, there are other contexts in this page where those 15

countries also co-occur together. By contrast, although there are a few references

to the United Kingdom in the Euro page, it does not occur in the same context

as the three examples (except for the page itself).

Entity ranking using Wikipedia as a pivot

In this paper we investigate the task of Entity Ranking on the Web. Searchers looking for entities are arguably better served by presenting a ranked list of entities directly, rather than a list of [**web pages**](http://academic.research.microsoft.com/Keyword/45005/web-pages) with relevant but also potentially redundant information about these entities. Since entities are represented by their web homepages, a naive approach to entity ranking is to use standard text retrieval. Our experimental results clearly demonstrate that [**text retrieval**](http://academic.research.microsoft.com/Keyword/41733/text-retrieval) is effective at finding relevant pages, but performs poorly at finding entities. Our proposal is to use Wikipedia as a pivot for finding entities on the Web, allowing us to reduce the hard web entity ranking problem to easier problem of Wikipedia entity ranking. Wikipedia allows us to properly identify entities and some of their characteristics, and Wikipedia's elaborate category structure allows us to get a handle on the entity's type. Our main findings are the following. Our first finding is that, in principle, the problem of web entity ranking can be reduced to Wikipedia entity ranking. We found that the majority of entity ranking topics in our test collections can be answered using Wikipedia, and that with high precision relevant web entities corresponding to the Wikipedia entities can be found using Wikipedia's 'external links'. Our second finding is that we can exploit the structure of Wikipedia to improve entity ranking effectiveness. Entity types are valuable retrieval cues in Wikipedia. Automatically assigned entity types are effective, and almost as good as manually assigned types. Our third finding is that web entity retrieval can be significantly improved by using Wikipedia as a pivot. Both Wikipedia's external links and the enriched Wikipedia entities with additional links to homepages are significantly better at finding primary web homepages than [**anchor text**](http://academic.research.microsoft.com/Keyword/1552/anchor-text) retrieval, which in turn significantly improved over standard text retrieval

ObjectRank: Authority-Based Keyword Search in Databases

The ObjectRank system applies authority-based ranking to [**keyword search**](http://academic.research.microsoft.com/Keyword/21563/keyword-search) in databases modeled as labeled graphs. Conceptually, authority orig- inates at the nodes (objects) containing the key- words and flows to objects according to their se- mantic connections. Each node is ranked accord- ing to its authority with respect to the particular keywords. One can adjust the weight of global importance, the weight of each keyword of the query, the importance of a result actually con- taining the keywords versus being referenced by nodes containing them, and the volume of au- thority flow via each type of semantic connection. Novel performance challenges and opportunities are addressed. First, schemas impose constraints on the graph, which are exploited for performance purposes. Second, in order to address the issue of authority ranking with respect to the given key- words (as opposed to Google's global PageRank) we precompute single keyword ObjectRanks and combine them during run time. We conducted user surveys and a set of performance experiments on multiple real and synthetic datasets, to assess the semantic meaningfulness and performance of ObjectRank.

An alternative way to conceive the intuition behind ObjectRank

is to consider that authority/importance flows in

the database graph in the same fashion that [25] defined

authority-based search in arbitrary graphs. Initially the

“OLAP” authority is found at the objects that contain the

keyword “OLAP”. Then authority/importance flows, following

the rules in the authority transfer schema graph, until

an equilibrium is established that specifies that a paper

is authoritative if it is referenced by authoritative papers, is

written by authority authors and appears in authority conferences.

Vice versa, authors and conferences obtain their

authority from their papers. Notice that the amount of authority

flow from, say, paper to cited paper or from paper

to author or from author to paper, is arbitrarily set by a domain

expert and reflects the semantics of the domain. For

example, common sense says that in the bibliography domain

a paper obtains very little authority (or even none) by

referring to authoritative papers. On the contrary it obtains

a lot of authority by being referred by authoritative papers.

Our DBLP demo offers to the user more than one authority

flow settings, in order to accommodate multiple user

profiles/requirements.

 EntityAuthority: Semantically Enriched Graph-Based Authority Propagation

This paper pursues the recently emerging paradigm of searching for entities that are embedded in Web pages. We utilize information- extraction techniques to identify entity candidates in documents, map them onto entries in a richly structured ontology, and derive a generalized data graph that encompasses Web pages, entities, and ontological concepts and relationships. We exploit this combina- tion of pages and entities for a novel kind of search-result ranking, coined EntityAuthority, in order to improve the quality of keyword queries that return either pages or entities. To this end, we utilize the mutual reinforcement between authoritative pages and impor- tant entities. This resembles the HITS method for Web-graph[**link analysis**](http://academic.research.microsoft.com/Keyword/23014/link-analysis) and recently proposed ObjectRank methods, but our ap- proach operates on a much richer, typed graph structure with dif- ferent kinds of nodes and also differs in the underlying mathemat- ical denitions. Preliminary experiments with topic-specic slices of Wikipedia demonstrate the effectiveness of our approach on cer- tain classes of queries.

## ObjectLevel Ranking: Bringing Order to Web Objects

Traditional PageRank model is no longer valid for object

popularity calculation because of the existence of heteroge-

neous relationships between objects. This paper introduces

*PopRank*, a domain-independent object-level link analysis

model to rank the objects within a speci¯c domain. Speci¯-

cally we assign a popularity propagation factor to each type

of object relationship, study how di®erent popularity propa-

gation factors for these heterogeneous relationships could af-

fect the popularity ranking, and propose e±cient approaches

to automatically decide these factors. Our experiments are

done using 1 million CS papers, and the experimental re-

sults show that *PopRank* can achieve signi¯cantly better

ranking results than naively applying PageRank on the ob-

ject graph

*PopRank* extends the PageRank model by adding a *pop-*

*ularity propagation factor (PPF)* to each link pointing to

an object, and uses di®erent propagation factors for links of

di®erent types of relationships. For example, for the links

pointing to a paper object, we need three propagation fac-

tors (*°*3, *°*2, and *°*1 in Figure 1) for the three di®erent types

of relationships: cited-by, authored-by, and published-by,

respectively. However manually assigning these factors to

make the popularity ranking reasonable is extremely chal-

lenging. With a huge link graph, it is very hard for us to

tell which types of links are more important and even harder

to quantify their exact importance. Fortunately it is always

easier for us to collect some partial ranking of the objects

from domain experts. For example, as a researcher, we know

the order of the top conferences or journals within our ¯eld,

and we may also know which papers are more popular. We

propose a learning based approach to automatically learn

the popularity propagation factors for di®erent types of links

using the partial ranking of the objects given by domain ex-

perts. The simulated annealing algorithm is used to explore

the search space of all possible combinations of propagation

factors and to iteratively reduce the di®erence between the

partial ranking from the domain experts and that from our

learned model.

**Using Naming Authority to Rank Data and Ontologies for Web Search**

n such environments, the authority of data sources is an important signal that the [**ranking algorithm**](http://academic.research.microsoft.com/Keyword/34360/ranking-algorithm) has to take into account. This paper presents algorithms for prioritising data returned by queries over web datasets expressed in RDF. We introduce the notion of naming authority which provides a correspondence between identifiers and the sources which can speak authoritatively for these identifiers. Our algorithm uses the original PageRank method to assign authority values to data sources based on a naming authority graph, and then propagates the authority values to identifiers referenced in the sources.

Hierarchical Link Analysis for Ranking Web Data

. Previous works have shown how PageR- ank can be adapted to achieve entity ranking. In this paper, we propose to exploit locality on the [**Web of Data**](http://academic.research.microsoft.com/Keyword/44998/web-of-data) by taking a layered approach, similar to hierarchical PageRank approaches. We provide justications for a two-layer model of the Web of Data, and introduce DING (Dataset Ranking) a novel ranking methodology based on this two-layer model. DING uses links between datasets to compute dataset ranks and com- bines the resulting values with semantic-dependent entity ranking strate- gies. We quantify the eectiveness of the approach with other link-based algorithms on large datasets coming from the Sindice search engine. The evaluation which includes a [**user study**](http://academic.research.microsoft.com/Keyword/43748/user-study) indicates that the resulting rank is better than the other approaches

**Weighted Link Analysis.** When working with more heterogeneous links, stan-

dard approaches do not provide accurate results since links of di  
erent types can

have various impact on the ranking computation. In [5, 6], the authors extend

PageRank to consider di  
erent types of relations between entities. PopRank [7],

an object-level link analysis, proposes a machine learning approach to assign a

\popularity propagation factor" to each type of relation. ObjectRank [8] applies

authority-based ranking to keyword search in databases. However, these works

do not consider the features of links such as their specicity and cardinality to

assign weights in an unsupervised fashion. Furthermore, these approaches are

too complex to apply on web-scale since they will require multiple times the

current processing power of current web search engines. A major task is to bring

down this computational power requirement.

**Semantic Web Link Analysis**. SemRank [15] proposes a method to rank

semantic relations using information-theory techniques but is solely focussed

on ranking and retrieval of relationships. The Swoogle search engine [1] is the

rst one to propose OntoRank, an adaptation of PageRank for Semantic Web

resources. In ReconRank [2], a link analysis is applied at query time for com-

puting the popularity of resources and documents. The above algorithms only

consider the individual web resources, disregarding the semantics and structure

of the Web of Data. They are therefore costly on a web-scale and likely provide

sub-optimal results. We are aware of one recent study [4] that has analysed the

e  
ectiveness of PageRank on the domain level for ranking Semantic Web re-

sources. However, their approach disregards the link structure between entities

within a domain, and does not consider weighted relations.

**Finding and Ranking Knowledge on the Semantic Web**

OntoRank

Swoogle’s OntoRank is based on the rational surfer model which emulates an agent’s

navigation behavior at the document level. Like the random surfer model, an agent

either follows a link in an SWD to another or jumps to a new random SWD with a

constant probability 1 ¡ d. It is ‘rational’ because it emulates agents’ navigation on

the Semantic Web, i.e., agents follow links in a SWD with non-uniform probability

according to link semantics. When encountering an SWD ®, agents will (transitively)

import the “official” ontologies that define the classes and properties referenced by ®.

Let link(®; l; ¯) be the semantic link from an SWD ® to another SWD ¯ with tag l;

linkto(®) be a set of SWDs link directly to the SWD ®; weight(l) be a user specified

navigation preference on semantic links with type l, i.e., TM and EX; OTC(®) be a

set of SWDs that (transitively) IM or EX ® as ontology; f(x; y) and wPR(x) be two

intermediate functions.

OntoRank is computed in two steps: (i) iteratively compute the rank, wPR(®), of

each SWD ® until it converges (equations 1 and 2); and (ii) transitively pass an SWD’s

rank to all ontologies it imported (equation 3).

TripleRank: Ranking Semantic Web Data by Tensor Decomposition

 Existing methods for graph- based authority ranking lack support for fine-grained latent coherence between resources and predicates (i.e. support for link semantics in the[**linked data**](http://academic.research.microsoft.com/Keyword/23031/linked-data) model). In this paper, we present TripleRank, a novel approach for faceted authority rank- ing in the context of RDF knowledge bases. TripleRank captures the additional latent semantics of [**Semantic Web**](http://academic.research.microsoft.com/Keyword/36922/semantic-web) data by means of statistical methods in order to produce richer descriptions of the available data. We model the [**Semantic Web**](http://academic.research.microsoft.com/Keyword/36922/semantic-web) by a 3-dimensional tensor that enables the seamless representation of arbitrary semantic links. For the analysis of that model, we apply the PARAFAC decompo- sition, which can be seen as a multi-modal counterpart to Web authority ranking with HITS. The result are groupings of resources and predicates that characterize their authority and navigational (hub) properties with respect to identified topics. We have applied TripleRank to multiple data sets from the [**linked open data**](http://academic.research.microsoft.com/Keyword/60190/linked-open-data) com- munity and gathered encouraging feedback in a [**user evaluation**](http://academic.research.microsoft.com/Keyword/43702/user-evaluation) where TripleRank results have been exploited in a faceted browsing scenario.

**Effective and efficient entity**

**search in rdf data,**

Triple stores have long provided RDF storage as well as data access using expressive, formal query languages such as SPARQL. The new end users of the Semantic Web, however, are mostly unaware of SPARQL and overwhelmingly prefer imprecise, informal keyword queries for searching over data. At the same time, the amount of data on the Semantic Web is approaching the limits of the architectures that provide support for the full expressiveness of SPARQL. These factors combined have led to an increased interest in semantic search, i.e. access to RDF data using Information Retrieval methods. In this work, we propose a method for effective and efficient entity search over RDF data. We describe an adaptation of the BM25F ranking function for RDF data, and demonstrate that it outperforms other state-of-the-art methods in ranking RDF resources. We also propose a set of new index structures for efficient retrieval and ranking of results. We implement these results using the open-source MG4J framework.

In our view of[**Information Retrieval**](http://academic.research.microsoft.com/Keyword/19900/information-retrieval) on the Semantic Web, a [**search engine**](http://academic.research.microsoft.com/Keyword/36510/search-engine) returns documents rather than, or in addition to, exact values in response to user queries. For this purpose, our approach includes an ontology-based scheme for the semi- [**automatic annotation**](http://academic.research.microsoft.com/Keyword/2608/automatic-annotation) of documents, and a retrieval system. The [**retrieval model**](http://academic.research.microsoft.com/Keyword/35520/retrieval-model) is based on an adaptation of the classic vector-space model, including an annotation weighting algorithm, and a ranking algorithm. [**Semantic search**](http://academic.research.microsoft.com/Keyword/36916/semantic-search) is combined with conventional keyword-based retrieval to achieve tolerance to [**knowledge base**](http://academic.research.microsoft.com/Keyword/21646/knowledge-base) incompleteness. Experiments are shown where our approach is tested on corpora of significant scale, showing clear improvements with respect to key- word-based search.

An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval

**Effective Keyword Search in Relational Databases**

Even though the major

RDBMSs have provided full-text search capabilities, they still

require users to have knowledge of the database schemas and use

a structured query language to search information. This search

model is complicated for most ordinary users. Inspired by the big

success of information retrieval (IR) style keyword search on the

web, keyword search in relational databases has recently emerged

as a new research topic. The differences between text databases

and relational databases result in three new challenges: (1)

Answers needed by users are not limited to individual tuples, but

results assembled from joining tuples from multiple tables are

used to form answers in the form of tuple trees. (2) A single score

for each answer (i.e. a tuple tree) is needed to estimate its

relevance to a given query. These scores are used to rank the most

relevant answers as high as possible. (3) Relational databases

have much richer structures than text databases. Existing IR

strategies are inadequate in ranking relational outputs. In this

paper, we propose a novel IR ranking strategy for effective

keyword search.

A similarity value between

a given query and a document is computed to rank documents.

However, the basic text information unit stored in a relational

database is a text column value, while the basic unit of answers

needed by users is a tuple tree, which is assembled by joining

multiple tuples, each of which may contain zero, one or multiple

text column values (**each text column value is considered as a**

**document**). A

The weighting method in Hristidis et al. [11] considers each text

column as a **collection**, and uses the standard IR weighting

method as shown in Formula 2 to compute a weight for each term

*k* in each document *Di*. Then, as shown in Formula 4, each weight

is normalized (divided by *size(T)*, i.e. the number of tuples in *T*).

The weights of the term in all documents are summed to obtain

the term weight in the super-document *T*. Formula 4 identifies and

deals with a new factor, *size(T)*, that affects similarity. However,

more factors need to be considered.

**Four Normalizations**

*Tuple Tree Size Normalization*

Document Length Normalization Reconsidered

*Document Frequency Normalization*

*Inter-Document Weight Normalization*

An Adaptation of the Vector-Space Model for

Ontology-Based Information Retrieval

The annotations are used by the retrieval and ranking module, as will be explained in Section IV.

The ranking algorithm is based on an adaptation of the classic vector-space model [31]. In the

classic vector-space model, keywords appearing in a document are assigned weights reflecting

that some words are better at discriminating between documents than others. Similarly, in our

system, annotations are assigned a weight that reflects how relevant the instance is considered to

be for the document meaning. Weights are computed automatically by an adaptation of the TFIDF

algorithm [31], based on the frequency of occurrence of the instances in each document.

The RDQL query is executed against the knowledge base, which returns a list of instance

tuples that satisfy the query. This step of the process is purely Boolean (i.e. based on an exact

match), so that the returned instances must strictly hold all the conditions in the formal query. Finally, the documents that are annotated with the instances returned in the previous step are

retrieved, ranked, and presented to the user.

**A Generalized Vector Space Model**

In [4], a generalized VSM was proposed so that a document or a query was represented

by a vector over a space of generalized terms each of which was either a

keyword or an NE triple. As usual, similarity of a document and a query was defined

by the cosine of the angle between their representing vectors. The work implemented

the model by developing a platform called S-Lucene modified from

Lucene4. The system automatically processed documents for NE-keyword-based

searching in the following steps:

1. Removing stop-words in the documents.

2. Recognizing and annotating named entities in the documents using KIM5.

3. Extending the documents with implied NE triples. That is, for each entity

named *n* possibly with class *c* and identifier *id* in the document, the triples

(*n*/\*/\*), (*\**/*c*/\*), (*n*/*c*/\*), (*alias*(*n*)/\*/\*), (*\**/*super*(*c*)/\*), (*n*/*super*(*c*)/\*),

(*alias*(*n*)/*c*/\*), (*alias*(*n*)/ *super*(*c*)/\*), and (*\**/\*/*id*) were added for the

document.

4. Indexing NE triples and keywords by S-Lucene.

Here *alias*(*n*) and *super*(*c*) respectively denote any alias of *n* and any super class

of *c* in the ontology and knowledge base of discourse.

A query was also automatically processed in the following steps:

1. Removing stop-words in the query.

2. Recognizing and annotating named entities in the query.

3. Representing each recognized entity named *n* possibly with class *c* and

identifier *id* by the most specific and available triple among (*n*/\*/\*), (*\**/*c*/\*),

(*n*/*c*/\*), and (\*/\*/*id*).

**Naga**

In this paper, we propose NAGA, a new semantic search

engine. NAGA builds on a knowledge base, which is organized as

a graph with typed edges, and consists of millions of entities and

relationships extracted from Web-based corpora. A graph-based

query language enables the formulation of queries with additional

semantic information. We introduce a novel scoring model,

based on the principles of generative language models, which

formalizes several notions such as confidence, informativeness

and compactness and uses them to rank query results. We

demonstrate NAGA’s superior result quality over state-of-the-art

search engines and question answering systems.

A good ranking model for answer graphs should satisfy the

following desiderata:

1) Confident answers (i.e. answers containing facts with

high extraction confidence from authoritative pages)

should be ranked higher.

2) Informative answers should be ranked higher. For example,

when asking a query Albert Einstein isA $z the

answer Albert Einstein isA physicist should rank higher

than the answer Albert Einstein isA politician, because

Einstein is rather known as a physicist than as a politician.

Similarly, for a query such as $y isA physicist, the

answers about world class physicists should rank higher

than those about hobby physicists.

3) Compact answers should be preferred, i.e. direct connections

rather than loose connections between entities are

preferable. For example, for the query How are Einstein

and Bohr related? the answer about both having won the

Nobel Prize should rank higher than the answer that Tom

Cruise connects Einstein and Bohr by being a vegetarian like Einstein, and by being born in the year in which

Bohr died

Our approach is inspired

by existing work on language models (LM) for information

retrieval (IR) on document collections [31], [19], but it is

adapted and extended to the new domain of knowledge graphs.

In this setting, the basic units are not words, but facts or fact

templates. Our graphs and queries can be seen as sets of facts

or fact templates respectively. A candidate result graph in our

setting corresponds to a document in the standard IR setting

Assume probabilistic independence between the query’s fact templates

Define the likelihood of a query fact given an answer graph as

a mixture of two distributions, ˜ P(qi|g) and ˜ P(qi)

P(qi|g) = \_ · Pconf (qi|g) + (1 − \_) · Pinfo(qi|g)

In summary, confidence and informativeness are two complementary

components of our model. The confidence expresses

how certain we are about a specific fact – independent

of the query and independent of how popular the fact is on the

Web. The informativeness captures how useful the fact is for

a given query. This depends also on how visible the fact is on

the Web. In this spirit, our definition of informativeness differs

from the information theoretic one, which would consider

less frequent facts more informative.

The compactness of answers

is implicitly captured by their likelihood given the query. This

is because the likelihood of an answer graph is the product

over the probabilities of its component facts. Therefore, the

more facts in an answer graph the lower its likelihood and

thus its compactness.

For example, for the query Margaret Thatcher connect Indra

Gandhi the answer graph stating that they are both primeministers,

is more compact than the answer that they are both

prime-ministers of English-speaking countries.

**Searching RDF Graphs with SPARQL and Keywords**

LM for this triple pattern is a probability

distribution over these triples (with smoothing by giving a small amount of probability mass to all other triples).

The probability of a given triple in the LM of a triple pattern can be viewed as the probability that the user

is interested in this particular triple as an answer to her question.

## As discussed above, some triples are more

## informative than others, because they refer to important directors, important movies, etc. We can incorporate

## this aspect into the LM by using statistical weights for different triples in order to construct a non-uniform

## distribution. To this end, we consider the witnesses of a given triple: how often, on the Web or in the news, do

## we see this triple. Alternatively, if we construct the RDF data collection by automatic information extraction

## from Web sources, the witnesses of a triple are the distinct sources from which we have extracted the triple.

## In our implementation, we issued keywords queries for each triple against a major search engine and used the

## reported result sizes as estimates for witness count

Web Object Retrieval

. In this paper, we propose a [**paradigm shift**](http://academic.research.microsoft.com/Keyword/29989/paradigm-shift) to enable searching at the object level. In traditional [**information retrieval**](http://academic.research.microsoft.com/Keyword/19900/information-retrieval) models, documents are taken as the retrieval units and the content of a document is considered reliable. However, this reliability assumption is no longer valid in the object retrieval context when multiple copies of information about the same object typically exist. These copies may be inconsistent because of diversity of Web site qualities and the limited performance of current [**information extraction**](http://academic.research.microsoft.com/Keyword/19851/information-extraction) techniques. If we simply combine the noisy and inaccurate attribute information extracted from different sources, we may not be able to achieve satisfactory retrieval performance. In this paper, we propose several language models for Web object retrieval, namely an unstructured object retrieval model, a structured object retrieval model, and a [**hybrid model**](http://academic.research.microsoft.com/Keyword/18752/hybrid-model) with both structured and unstructured retrieval features. We test these models on a paper [**search engine**](http://academic.research.microsoft.com/Keyword/36510/search-engine) and compare their performances. We conclude that the [**hybrid model**](http://academic.research.microsoft.com/Keyword/18752/hybrid-model) is the superior by taking into account the extraction errors at varying levels.

Relevance Feedback between Web Search and the Semantic Web

We investigate the possibility of using structured

data to improve search over unstructured documents.

In particular, we use relevance feedback to

create a ‘virtuous cycle’ between structured data

from the Semantic Web and web-pages from the

hypertext Web. Previous approaches have generally

considered searching over the Semantic Web

and hypertext Web to be entirely disparate, indexing

and searching over different domains. Our

novel approach is to use relevance feedback from

hypertext Web results to improve Semantic Web

search, and results from the Semantic Web to improve

the retrieval of hypertext Web data. In both

cases, our evaluation is based on certain kinds of

informational queries (abstract concepts, people,

and places) selected from a real-life query log and

checked by human judges. We show our relevance

model-based system is better than the performance

of real-world search engines for both hypertext and

Semantic Web search, and we also investigate Semantic

Web inference and pseudo-relevance feedback.

USING LANGUAGE AND RELEVANCE MODEL

We can compare both structured Semantic

Web and unstructured hypertext data by considering both

to be an unstructured ‘bags of words,’ albeit weighted by either

frequency in unstructured text, weights given to RDF

structure, or both. Semantic Web data is ‘flattened’ into a

‘document’ of terms derived from both the text and URIs in

the RDF graph.

**Learning to Rank Relational Objects and Its Application to**

**Web Search**

Learning to rank is a new statistical learning technology on

creating a ranking model for sorting objects. The technology

has been successfully applied to web search, and is becoming

one of the key machineries for building search engines. Exist-

ing approaches to learning to rank, however, did not consider

the cases in which there exists relationship between the ob-

jects to be ranked, despite of the fact that such situations are

very common in practice. For example, in web search, given

a query certain relationships usually exist among the the

retrieved documents, e.g., URL hierarchy, similarity, etc.,

and sometimes it is necessary to utilize the information in

ranking of the documents. This paper addresses the issue

and formulates it as a novel learning problem, referred to

as, `learning to rank relational objects'. In the new learning

task, the ranking model is de¯ned as a function of not only

the contents (features) of objects but also the relations be-

tween objects. The paper further focuses on one setting of

the learning problem in which the way of using relation in-

formation is predetermined. It formalizes the learning task

as an optimization problem in the setting. The paper then

proposes a new method to perform the optimization task,

particularly an implementation based on SVM. Experimen-

tal results show that the proposed method outperforms the

baseline methods for two ranking tasks (Pseudo Relevance

Feedback and Topic Distillation) in web search, indicating

that the proposed method can indeed make e®ective use of

relation information and content information in ranking.

# Discussion

Extensive experimental work on five test collections in two domains shows that our

approach gives significant improvements in terms of recall, initial precision and mean

average precision with respect to a baseline without relevance feedback. On one test collection,

it is also able to outperform a text-based pseudo-relevance feedback approach

based on relevance models. On the other test collections it performs similarly to relevance

models. Overall, conceptual language models have the added advantage of offering query

and browsing suggestions in the form of conceptual annotations. In addition, the internal

structure of the meta-language can be exploited to add related terms.

The introduction of concept languages was initially driven by a need to facilitate search and navigation of the collection

(Roberts, 1984; Joyce & Needham, 1958). Concepts were defined to unambiguously and precisely represent the content of

documents. Today, most of these early retrieval systems have been replaced by full-text search systems which have been

shown to be at least as effective (Cleverdon, Mills, & Keen, 1966). Since full-text search systems do not require a manually

curated concept language, they are far less labour-intensive. Despite the effectiveness of full-text search, full-text indexing

terms (which typically comprise of all the terms used in the documents in a given collection) can be more ambiguous or less

expressive than concepts. Not surprisingly then, information retrieval (IR) researchers continue to study ways of incorporating

information from concept languages to address problems in textual query representations. For example, a textual query

may be mapped to one or more concepts in a thesaurus and expanded with their synonymous terms (Voorhees, 1994). Results

of such approaches, however, have been mixed at best.

**Ontology as a Search-Tool: A Study of Real Users’ Query**

**Formulation With and Without Conceptual Support**

This study examines 16 real users’ use of an ontology as a search

tool. The users’ queries constructed with the help of a Concept-based Information

Retrieval Interface (CIRI) were compared to queries created independently

based on the same search task description. Also the effectiveness of the CIRI

queries was compared to the users’ unaided queries. The simulated search task

method was used to make the searching situations as close to real as possible.

Due to CIRI’s query expansion feature the number of search terms was remarkably

higher in ontology queries than in Direct interface queries. The search

results were evaluated with generalised precision and generalised relative recall

as well as precision based on personal assessments. The Direct interface queries

performed better in all methods of comparison.