Interpreting Keyword Queries over Web Knowledge Bases

Jeffrey Pound Alexander K. Hudek Ihab F. Ilyas[†] Grant Weddell
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada

[†]Qatar Computing Research Institute (QCRI)

{jpound,akhudek,gweddell}@cs.uwaterloo.ca, [†]ikaldas@qf.org.qa

ABSTRACT

Many keyword queries issued to Web search engines target information about real world entities, and interpreting these queries over Web knowledge bases can often enable the search system to provide exact answers to queries. Equally important is the problem of detecting when the reference knowledge base is not capable of answering the keyword query, due to lack of domain coverage.

In this work we present an approach to computing structured representations of keyword queries over a reference knowledge base. We mine frequent query structures from a Web query log and map these structures into a reference knowledge base. Our approach exploits coarse linguistic structure in keyword queries, and combines it with rich structured query representations of information needs.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Query formulation, Retrieval models*

1. INTRODUCTION

As the amount of structured data on the Web continues to grow, the ability to exploit this data for keyword search becomes increasingly important. Efforts such as DBpedia[1], Freebase[2], and Linked Data[3] have produced large heterogeneous *knowledge bases* that encode great amounts of information about many real world entities. Web search queries seeking information about these entities could be better served by interpreting the query over a knowledge base in order to provide an exact answer to the query, or to enhance document retrieval by understanding the entities described by the query.

For example, consider a user searching for *"songs by jimi hendrix."* While a text search may retrieve relevant documents to the query terms, it leaves the user with the task of scouring textual results in search of the information they are seeking. This user's query could be better served by returning a list of particular songs by the musician Jimi Hendrix (precise answers to the query), possibly along with information about each song (e.g., structured facts from a knowledge base or documents resulting from a text search for the particular song). Similarly, the keyword query *"author of moby dick"* could be better answered if a search system understood that "moby dick" describes a particular book, "author of" describes a relationship, and the *query intent* is to find the unspecified entity that has an "author of" relationship to "moby dick."

We refer to the process of interpreting keyword queries over a knowledge base as *semantic query understanding*. This problem has the following characteristics that pose difficult technical challenges.

- **Ambiguity** Keyword queries tend to be short, ambiguous, and underspecified. For a given keyword query there may be multiple possible ways to interpret the underlying query intent. Semantic query understanding systems need to accurately interpret entity-based keyword queries when the underlying reference knowledge base contains the relevant information.
- **Representation & Coverage** Not all keyword queries have an entity focus (so called, *entity-based keyword queries*). A query such as *"corporate tax laws buyout"* may intentionally be seeking documents that mention the given query terms, and an attempt to represent the query in terms of knowledge base entities may produce an incorrect interpretation and harm the quality of results. Similarly, some entity-based queries may seek information that does not exist in a given reference knowledge base. A high accuracy semantic query understanding system will answer queries only when the query intent can be represented and evaluated over the reference knowledge base.
- **Scale** Web knowledge bases tend to be very large and heterogenous, meaning approaches cannot be engineered to exploit domain specific regularities or depend on small fixed schemas with complete data. Semantic query understanding techniques must scale to large heterogeneous Web knowledge bases.

1.1 Motivation

Query understanding is a broad phrase used to describe many techniques applied to keyword queries in order to

[1] http://dbpedia.org
[2] http://www.freebase.com
[3] http://linkeddata.org

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.
Copyright 2012 ACM 978-1-4503-1156-4/12/10 ...$15.00.

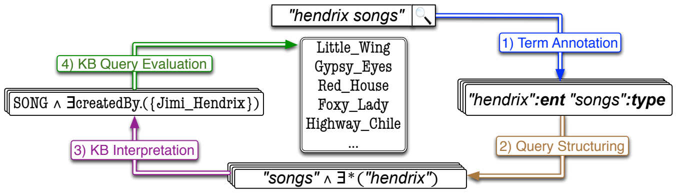

Figure 1: Overview of the query understanding process.

represent the underlying information need in a way that a search system can exploit. Some popular approaches include topic classification, in which the topic of a query is determined using a statistical classifier. Topic classification aids a search system by giving it context, or by allowing the system to select the appropriate data source to search. For example, the query

$$\text{“\textit{songs by jimi hendrix}”} \qquad (1)$$

may be classified as a music query. However query classification does not give any insight into precisely what the query is looking for within the domain of music.

Another query understanding approach that has received recent attention from researchers is term annotation. Term annotation labels individual query terms with annotations that describe the role these terms play with respect to an underlying information system. Following our example query, the terms *“jimi hendrix”* may be annotated as a *name*, and the term *“songs”* annotated as an entity *type*. While query annotations can aid a search system in understanding the meaning of individual terms, the annotations do not describe the underlying structure of the query as a single coherent query interpretation. Continuing the example, we want to model the latent query structure that expresses the query intention as finding entities of type *song* that are created by an entity named *“jimi hendrix.”* (as opposed to finding, for example, information about the person named *“jimi hendrix”* who is known to have written songs, or a song named *“jimi hendrix”*).

The goal of *semantic query understanding* is to compute a formal interpretation of an ambiguous information need. Figure 1 shows our proposed semantic query understanding process for a variation of our running example information need. Our example keyword query can be structurally understood as finding all *entities* of a *type* described as “songs,” that have a “by” *relationship* to an entity described as “jimi hendrix”. This is given as the following conjunctive query.

$$q(x)\text{:-}\exists y.\texttt{SONG}(x) \wedge \mathit{createdBy}(x,y) \wedge y = \texttt{Jimi\_Hendrix} \quad (2)$$

Where the predicates SONG and *createdBy*, and the entity Jimi_Hendrix are part of a Web knowledge base. The difficulty in mapping a keyword query to a formal interpretation lies in the inherent ambiguity in keyword queries, and the many possible mappings query terms can have over very large reference data collections. For example, the term *“hendrix”* matches 68 data items in our data set, and the term *“songs”* matches 4,219 items, and the term *“by”* matches 7,179 items (see Section 5 for details on the data set used for our experiments). This yields a space of over two billion possible conjunctive queries constructed by mapping each query term to a syntactically similar predicate. This does not count the possibility of additional predicates existing in the underlying conjunctive query that are not explicitly represented by a term in the keyword query. There is also great variation in how queries are expressed; many Web queries are very short and underspecified. For example, the query *“hendrix songs”* shown in Figure 1 shares the same logical structure as example query 1, even though the *createdBy* relation is not explicitly represented by any query terms. It is the job of the query understanding system to infer the existence of such latent relations.

In this work we propose a method for interpreting keyword queries over Web knowledge bases. We use an annotated Web search query log as training data to learn a mapping between keyword queries and their underlying semantic structures. Because of the very high cost in creating training data, we design an approach that maps high level representations of keyword queries to schema-level representations of structured queries, greatly reducing the amount of training data needed to accurately learn these mappings. Our approach integrates state-of-the-art techniques in natural language processing with top-k search over structured data and knowledge base query processing, bridging the gap between language models for statistical representations of keyword queries and database formalisms for structured representations of information needs.

1.2 Contributions

In this work, we make the following contributions.

- We propose a novel method for interpreting keyword queries over Web knowledge bases that computes probable semantic structures based on a statistical model learned from real user queries, and maps them into a knowledge base.
- We show an encoding of the semantic annotation problem as a parameter estimation problem for a conditional random field, and propose a method of structuring annotated queries that uses a high level representation of both keyword queries and the target knowledge base query structure. This high level representation allows us to learn a mapping from *semantic summaries* of keyword queries to *structured query templates* by exploiting the redundancy of the summaries and templates shared by different queries in a query log.
- We present the results of an analysis of a real Web search log, giving insight into the types of entity-based queries asked by real users. Our analysis shows the importance of addressing entity-based queries due to the large number of these queries issued by users. The analysis also establishes relationships between linguistic structure and semantic structure, and gives insight into the types of structures that repeatedly occur in keyword queries.

We demonstrate the viability of our proposed approach with an experimental evaluation of both the effectiveness and efficiency of a prototype implementation of the system.

1.3 Outline

The remainder of the paper is organized as follows. Section 2 gives an overview of the keyword query understanding problem, outlines the basis of our formalism for modeling query intentions, and gives an overview of our approach to semantic query understanding. The details of our approach are presented in Section 3. Section 4 describes the results of our analysis of a Web query log. In Section 5 we empirically evaluate our proposal. We review relevant work in Section 6 and conclude in Section 7.

2. DATA AND PROBLEM DEFINITIONS

2.1 Representing Knowledge and Queries

We adopt a formal model of knowledge representation as a collection of assertions based on the OWL2 EL Profile[4] extended with unary *entity* sets (allowing an entity to represent a class containing only itself), and with attributes that map to values in a concrete domain such as strings, numbers, or dates.

Definition 1. **(Concept)** A *concept* C is defined by the following grammar, where A is an entity type (a.k.a, entity class), R is a relation, R^- is the inverse of relation R, e is an entity, f is an attribute, k is a constant, and $\sqcap$ denotes concept conjunction (intersection)

$$C \quad ::= \quad A \mid \exists R(C) \mid \exists R^-(C) \mid C_1 \sqcap C_2 \mid f = k \mid \{e\}.$$

Definition 2. **(Knowledge base)** Facts about entities are represented as assertions. An *assertion* has one of the following forms:

$$A(e) \quad | \quad R(e_1, e_2) \quad | \quad f(e, k)$$

This first construct asserts that an entity e has type A, for example: $\texttt{GUITARIST}(\texttt{Jimi\_Hendrix})$. The second construct asserts that a relation R holds between two entities, for example: $\mathit{createdBy}(\texttt{Foxy\_Lady}, \texttt{Jimi\_Hendrix})$. This also implies the inverse $\mathit{createdBy}^-(\texttt{Jimi\_Hendrix}, \texttt{Foxy\_Lady})$. The last construct asserts that an entity has a particular value for a given attribute. This is a special case of a relation assertion, where the entity relates to a value. For example, $\mathit{dateOfBirth}(\texttt{Jimi\_Hendrix}, \text{1942-11-27})$. The assertion $\mathit{subClassOf}(A_1, A_2)$ is used to encode transitive class hierarchies among types.

A *knowledge base* $\mathcal{K}$ is represented as a set of assertions. We write $\mathcal{K} \models C$ to denote that concept C is consistent with $\mathcal{K}$ and $\mathcal{K} \models C(e)$ to denote that the knowledge base $\mathcal{K}$ entails that entity e can be inferred to be an instance of C.

We will ultimately use concepts in the given KB formalism to represent query intentions using the following definition of a search query over a knowledge base.

Definition 3. **(Concept Search Query)** A *concept search query* is given by a concept C expressed using Definition 1. The answer to the query C over a given knowledge base $\mathcal{K}$ is the set of entities inferred to be instances of C

$$\mathit{answer}(C, \mathcal{K}) = \{e \mid \mathcal{K} \models C(e)\}.$$

Our example query from Equation 2 can be modeled as the following: $\texttt{SONG} \sqcap \exists \mathit{createdBy}(\{\texttt{Jimi\_Hendrix}\})$.

2.2 The Query Understanding Problem

We can abstract a concept search query C as a conjunctive query λ^C consisting of u unary predicates and b binary predicates with a single distinguished variable. (This equivalence is straightforward, and can be seen in the relationship between assertions and concept expressions defined in Section 2.1.) Let $\mathcal{K}_\mathcal{P}$ denote the set of predicates occurring in a knowledge base $\mathcal{K}$, and $\lambda^C_\mathcal{P}$ denote the set of predicates (unary and binary) occurring in a conjunctive query λ^C. Let $\mathcal{L}(P)$ denote the set of textual labels that can be used to describe a predicate P. The space of possible queries λ^C for a given keyword query $Q = \langle q_1 q_2 \dots q_n \rangle$ is given by the following expression.

$$\begin{aligned} \lambda^C(x) \quad &= \quad \bigwedge_{i=1}^{u} P_i(x_j) \wedge \bigwedge_{i=1}^{b} P_i(x_j, x_k) \;\; \text{s.t.} && (3) \\ & \forall P \in \lambda^C_P, P \in \mathcal{K}_P && (4) \\ & \forall q \in Q, \exists P \in \lambda^C_P, q \in \mathcal{L}(P) && (5) \\ & \mathcal{K} \models C && (6) \end{aligned}$$

Equation 3 defines a space of all possible conjunctive queries with u unary predicates and b binary predicates. Equation 4 constrains all predicates to be part of the given knowledge base, ensuring the query is safe. Equation 5 ensures coverage of all query terms and Equation 6 ensures that the query is consistent with the knowledge base. Ideally we expect that $\mathcal{L}(P)$ contains all possible representations of P, including synonyms and linguistic variations of terms (e.g., lemmatized forms). In practice, we approximate this by relaxing the constraint $q \in \mathcal{L}(P)$ to allow fuzzy and partial string matching.

The *query understanding problem* is to find the most probable concept search query C, subject to the constraints in Equation 3, for a given keyword query Q:

$$C = \text{argmax}_{C'} \;\; \Pr(C'|Q) \qquad (7)$$

such that C represents the intention of Q with respect to a knowledge base $\mathcal{K}$. In the rest of this paper, we propose methods for estimating the quantity in Equation 7 by learning from a Web query log.

2.3 Solution Overview

Estimating the distribution in Equation 7 directly would require a large amount of labeled training examples. The space of possible keyword queries mapping to possible KB queries is very large, and estimating such a mapping directly is not feasible when training data is limited. Because of the manual effort required in creating labeled training data, we need to design an approach that can maximize the utility of a small collection of training examples.

Our semantic query understanding approach is summarized as a sequence of four steps, as illustrated in Figure 1.

1. **Keyword Query Annotation** Queries are first annotated with the semantic constructs from a knowledge representation language (i.e., entity, type, attribute, value, relation). We use part-of-speech tags as features that suggest probable semantic constructs for each query term. The mapping from part-of-speech tags to semantic constructs is learned from an annotated query log.
2. **Keyword Query Structuring** Annotated queries are structured by computing the most probable structured query templates given the annotations as a semantic summary of the query contents. This relationship between annotations and query structures is

[4] http://www.w3.org/TR/owl-profiles/#OWL_2_EL

learned from an annotated query log. Learning a mapping directly from keywords to structured queries would require large amounts of training examples. By learning the mapping from semantic summaries to query templates, we take advantage of the redundancy in the training data caused by many queries sharing the same summaries and templates.

3. **Knowledge Base Mapping** Semantically annotated keyword queries can be combined with a structured query template to form a structured representation of the keyword query known as a *structured keyword query*. We extend an existing approach [18] to map structured keyword queries into concept search queries.
4. **Knowledge Base Query Evaluation** Concept search queries are then executed over the knowledge base to find entities and values described by the query. This process performs query-time inference, exploiting the semantics encoded in the knowledge base to compute query answers using a custom knowledge base engine (discussion of the knowledge base engine is beyond the scope of this paper, however any database system that can implement the semantics defined in Section 2.1 could be substituted).

The main focus of this paper is on the first two steps, we build on existing methods to address the final two steps. Note that at steps 1 and 2 of the process, we are not yet concerned with how query terms map to particular knowledge base items as would be done in traditional keyword search over graphs. Inferring probable structures of the query terms first will allow us to constrain the possible mappings into the knowledge base. This can improve performance by fixing the structure of possible mappings into the knowledge base, and improve effectiveness by only allowing structures that have a high probability of being representative of query intentions. Inferring query structures also gives us the ability to know precisely what piece of information is being requested, much like a projection in structured query languages.

3. STRUCTURING KEYWORD QUERIES

As a first step to modeling the semantics of a keyword query, we map keyword sequences to structures representing their intents, known as *structured keyword queries*, addressing steps 1 and 2 of Figure 1. A structured keyword query describes a query in two ways: it breaks it into *segments* that represent particular semantic constructs, and it describes how these constructs relate to each other. In order to form structured keyword queries from a given keyword query, we will have to address these two problems.

3.1 Query Segmentation & Annotation

We define a keyword query Q to be a sequence of query terms $Q = q_1, q_2, \ldots, q_n$ over a vocabulary $\mathcal{V}$. A semantically annotated keyword query assigns *semantic constructs* from a schema language to sequences of keyword query terms.

Definition 4. **(Semantically Annotated Query)** Given a vocabulary $\mathcal{V}$, a keyword query Q and the knowledge representation language defined in Definition 1, a *semantically annotated keyword query AQ* (or simply *annotated query* when clear from context) is a sequence of keyword phrase-semantic construct pairs

$$AQ = \langle q_1 q_2 \ldots q_i \rangle{:}a_1 \; \langle q_{i+1} \ldots q_j \rangle{:}a_2 \; \ldots \langle q_{j+1} \ldots q_n \rangle{:}a_n$$

where each $q_i \in \mathcal{V}$ and each $a_i \in \{ent, type, rel, attr, val\}$.

A semantically annotated version of our example query is given by the following.

"songs":type *"by"*:rel *"jimi hendrix"*:ent

An algorithm that annotates keyword queries with semantic constructs must solve both the *segmentation problem* (the problem of determining the boundaries that separate multi-term phrases) and the annotation problem. We want to compute the probability of an annotated query given a keyword query, $\Pr(AQ|Q)$.

Research has shown that part-of-speech (POS) tagging can be accurately performed over keyword queries [2]. Our approach to annotating queries exploits query terms, their POS tags, and sequential relationships between terms and tags to concurrently infer a segmentation and semantic annotation of a part-of-speech annotated keyword query. To do this, we use a conditional random field (CRF) [12], a state-of-the-art machine learning method for labeling sequence data. As a baseline method for comparison, we also try a technique that directly classifies terms independently with semantic constructs. We then segment the query by joining any adjacent terms sharing the same semantic construct.

Baseline term classification Our baseline term classification aims to exploit the relationship between part-of-speech tags and semantic constructs by using a term's part-of-speech as a proxy for the term. Intuitively, there is a relationship between semantic constructs (e.g., entities, relations) and the parts-of-speech used in describing instances of those constructs. For example, entities are generally expressed using proper nouns like "Jimi Hendrix" or "New York." Relations are often described by prepositions, such as "in" or "by." The mapping between parts-of-speech and semantic constructs however, is not always so clear. Relations can sometimes be described by parts-of-speech other than prepositions (e.g., the noun "birthplace"); nouns can often describe many different semantic constructs, such as types, relations, and attributes; and perhaps the most challenging side of using part-of-speech tags to infer semantic constructs is that many entities, types, and relations are made up of multi-word phrases that can contain many different parts-of-speech (e.g., the relation "has won prize," or the type "Chancellors of Germany").

We model an annotated query log as set of triples $L = \{\langle Q, \pi, \sigma \rangle\}_i$ where Q is a keyword query, π is a function mapping query terms to POS tags, and σ is a function mapping query terms to semantic constructs. To classify terms via their part of speech tags, we use the standard naïve Bayes classification method. We perform naïve Bayes classification by directly estimating the joint probability distribution of POS tags and their semantic constructs from the query log. The conditional probability of a particular semantic construct C given a POS tag P is then the frequency of that query term's POS tag, P, mapping to C, versus the frequency of P mapping to *any* semantic construct, $\Pr(\mathrm{C}|\mathrm{P}) = \Pr(\mathrm{C}, \mathrm{P}) / \Pr(\mathrm{P})$ which is estimated from the query log by the following frequencies.

$$\Pr(\mathrm{C}|\mathrm{P}) = \frac{|\{\langle Q, \pi, \sigma \rangle \in L \; s.t. \exists q \in Q, \pi(q) = \mathrm{P}, \sigma(q) = \mathrm{C}\}|}{|\{\langle Q, \pi, \sigma \rangle \in L \; s.t. \exists q \in Q, \pi(q) = \mathrm{P}\}|}$$

With a distribution over semantic constructs given the part-of-speech of a query term, we can estimate the probability of assignments of semantic constructs to individual part-of-speech tagged query terms for a whole query. Assuming independence of query terms for tractability, this yields the following equation.

$$\Pr(AQ|Q) = \Pi_{q:\mathrm{C}\in AQ}\ \Pr(\mathrm{C}|\ \pi(q))$$

CRF feature design A CRF is an undirected probabilistic graphical model that models sequential data. Given a trained model and an input sequence, a CRF enables the computation of the most probable, or k most probable labelings according to the model. Specifically, a labeling is an assignment of state labels $y_1, \ldots, y_n$, to an input sequence $x_1, \ldots, x_n$, where each y_i corresponds to a state in the model and each x_i corresponds to a feature vector.

We base our CRF model on the design for noun-phrase detection proposed by Sha and Pereira [21] since our problem shares similarities. For input position x_i corresponding to query term q_i, we define a feature vector containing the following features: all query terms and POS tags in a size five window around position x_i; all query term bigrams in a size three window; all POS tag bigrams in a size five window; and all POS tag trigrams in a size five window. We include the actual query terms as features to allow important repetitive terms to be captured (e.g., "in" describing a relation such as *"restaurants in barcelona"*), but discard any generated feature that appears only once to avoid overfitting the particular terms in the training data.

We deviate from the model of Sha and Pereira in label design. The labels must encode both the semantic constructs we want to annotate as well as the boundaries between multi-term semantic constructs. We create two output labels for every semantic construct in our chosen knowledge representation language: a "begin" (B) and a "continue" (C) label. This is the minimal encoding that allows us to identify segmentation boundaries as well as semantic constructs. To generate training data, we label each multi-term phrase in the training data with the begin and continue labels corresponding the phrase's semantic construct. For example, the correct labeling of our running example is the following.

"songs":type-B *"by"*:rel-B *"jimi"*:ent-B *"hendrix"*:ent-C

Here, the query term *"hendrix"* is annotated as a continuation of the entity starting with *"jimi"*, yielding the following annotated query.

"songs":type *"by"*:rel *"jimi hendrix"*:ent

Unlike our baseline term classification approach, the CRF model can distinguish between multiple instances of the same semantic construct occurring in succession. We train our model using the Broyden-Fletche-Goldfarb-Shanno (BFGS) algorithm, a type of hill climbing approach for solving nonlinear optimization problems. To avoid overfitting we use L2 regularization. The probability $\Pr(AQ|Q)$ is then given directly by the CRF model.

3.2 Structuring Annotated Queries

An annotated query reveals part of the latent structure of an entity-based keyword query by indicating the semantic role represented by various parts of the query. However query annotation alone does not describe how these various recognized semantic constructs interact to model the underlying query intention. In our running example query, we know (after annotation) that the query contains a type, a relation, and an entity. However there is still ambiguity in what the query is seeking. Is it ultimately describing entities of the given type that are related to the given entity? Or is it seeking information about the given entity within the context of the given type?

To illustrate the ambiguity in query structure, consider the following two queries: *"john smith dentist"* and *"new york restaurants"*. Both queries contain an entity followed by a type. The first query seeks information about the given entity (*"john smith"*), with a type (*"dentist"*) given as context to disambiguate among possible interpretations of the entity. Whereas the second query is seeking instances of the given type (*"restaurants"*), within the context of the given entity (*"new york"*).

To model high level query structure, we follow the logical connectives from the knowledge representation language.

Definition 5. **(Structured Query Template)** A *structured query template* T is a schema-level description of a concept search query, expressed in the following grammar

$$T \quad ::= \quad node \ \mid \ edge(T) \ \mid \ T_1 \sqcap T_2$$

where $node \in \{ent, type, val\}$ and $edge \in \{rel, attr\}$.

A structured query template describes the overall graph structure of the query as well as the node and edge types of the query predicates. For example, the structured query template for our example query is $type \sqcap rel(ent)$.

We want to estimate the probability of a query template given a semantically annotated keyword query, $\Pr(T|AQ)$. We assume access to an annotated query log containing both semantically annotated queries and their structured query templates, $L = \{(AQ_i, T_i)\}$.

Our structuring approach directly estimates the probability of a structured template given an annotated query by aggregating over all queries in the training log that share the same high-level summary of semantic annotations.

Definition 6. **(Semantic Summary)** Given an annotated query $AQ = \langle q_1{:}a_1, q_2{:}a_2, \ldots, q_n{:}a_n\rangle$, a *semantic summary* is an ordered list of semantic constructs occurring in AQ, and is given by the function $S : AQ \rightarrow \mathrm{C}^n$ s.t. $S(AQ) = \langle a_1, a_2, \ldots, a_n\rangle$

For example, $S($*"songs"*:type *"by"*:rel *"jimi hendrix"*:ent$) = \langle type, rel, ent\rangle$.

We directly estimate $\Pr(T|AQ)$ from labeled training examples in our query log from the definition of conditional probabilities, using the semantic summary as a high level representation of the annotated query.

$$\Pr(T|AQ) = \frac{|\{\langle AQ', T'\rangle \in L \ s.t.T = T', S(AQ) = S(AQ')\}|}{|\{\langle AQ', T'\rangle \in L \ s.t.S(AQ) = S(AQ')\}|}$$

The probability of a query template given an annotated query is estimated by the proportion of queries with the same semantic summary that are structured using that template, versus the total number of queries with the same semantic summary and *any* structuring.

3.3 From Structures to KB Interpretations

Combing an annotated query with a structured query template yields a structured version of the keyword query. Combining the annotated running example query with its template gives the following query *"songs"*$\sqcap$ *"by"*(*"jimi hendrix"*) by swapping the annotated constructs into their respective positions in the template. We allow relations and attributes

POS	Ex.	Query	Token	Sem	Freq.
Proper noun	waterloo	77.1%	28.7%	ent	99.1%
				type	0.9%
Noun	musician	42.6%	15.9%	type	49.5%
				ent	42.7%
				attr	6.8%
				rel	0.9%
Plural noun	songs	13.6%	5.1%	type	68.0%
				ent	20.0%
				attr	8.0%
				rel	4.0%
Adjective	big	11.6%	4.3%	ent	60.0%
				type	40.0%
Preposition	in	7.8%	2.9%	rel	55.6%
				ent	44.4%
Number	2008	6.6%	2.5%	ent	53.8%
				val	46.2%
URI	yahoo.ca	5.0%	1.9%	ent	100%
Verb	run	4.7%	1.7%	ent	100%
Determiner	the	3.1%	1.2%	ent	100%
Gerund	winning	2.7%	1.0%	rel	66.7%
				type	33.3%

Figure 2: The ten most frequently occurring part-of-speech tags among entity-based queries, with the distribution of how frequently that tag mapped to various semantic constructs.

to be unmapped (e.g., see Figure 1), however we do not consider mappings in which entities or types are unmapped.

This type of expression is called a *structured keyword query*, and methods to map these expressions into Web knowledge bases have been developed [18]. The method proposed in [18] first generates a list of possible KB items for each keyword phrase in the query. These items are sorted by syntactic similarity, forming ordered input lists. The inputs are then processed using a variation of a top-k threshold algorithm.

In this work we extend the the method proposed in [18] in order to support the class of query structures discovered in our query log analysis. Our variation of structured keyword queries is given by the following grammar.

$$SKQ \ ::= \ k \ | \ k(SKQ) \ | \ *(SKQ) \ | \ SKQ_1 \sqcap SKQ_2$$

Where k is a sequence of keywords and $\sqcap$ denotes conjunction. This grammar follows the knowledge representation language defined in Section 2.1. We extend the original structured keyword query framework by adding the construct $*(SKQ)$ which allows nested queries with unknown relations. For this construct we must consider all possible relations in the knowledge base. We accomplish this by modifying the top-k algorithm used in [18] to dynamically add all relations that have non-zero knowledge base support to any KB item in adjacent input lists. We also add support to the underlying KB formalism to account for inverse relations and an explicit model of attributes.

Given a structured keyword query SKQ, the KB mapping system will return the most probable (or the top-k most probable) concept search query $C = \text{argmax}_{C'} \ \Pr(C'|SKQ)$. The probability of a structured keyword query (represented as a semantically annotated query and a query template $SKQ = \langle AQ, T \rangle$) is given by $\Pr(SKQ|Q) = \Pr(T|AQ) \cdot \Pr(AQ|Q)$. We can compute the first term using the techniques discussed in Section 3.1 and the second term using the method presented in Section 3.2. Our overall goal is to approximate the probability distribution given in Equation 7. Given a keyword query Q, the concept search query C representing the intent of Q is estimated as:

Template	Frequency
ent	44.9%
$type \sqcap rel(ent)$	12.8%
$ent_0 \sqcap rel(ent_1)$	7.7%
$ent \sqcap type$	5.8%
$type$	5.8%
$attr(ent)$	3.8%
$ent_1 \sqcap rel(ent_0)$	3.2%
$rel(ent)$	1.9%
$ent_0 \sqcap rel(ent_1, rel(ent_2))$	1.3%
$type_1 \sqcap rel(type_0)$	1.3%

Figure 3: The ten most frequently occurring templates among entity-based queries.

$$C = \text{argmax}_{C'} \ \Pr(C'|SKQ) \cdot \Pr(SKQ|Q) \qquad (8)$$

for some structured keyword query SKQ. Our produced concept search queries satisfy the problem definition in Equation 3. The queries are safe since all predicates are retrieved from the KB, they cover all query terms by only considering complete mappings into the KB, and they are guaranteed consistent as we evaluate resulting concept search queries and only include those that are non-empty.

4. ANALYSIS OF A WEB QUERY LOG

We analyzed a sample of keyword queries available from the Yahoo WebScope program [25]. We inspected queries keeping only those that have a *semantic construct* as the primary query intention, following the classification proposed in [19]. We annotated 258 queries with the part-of-speech tags and semantic constructs. We did not consider misspelled, non-English, or other queries that were not clearly understandable. Among the annotated queries, 156 queries had some semantic construct as their primary intention (*entity-based queries*). That is approximately 60% of queries having a semantic construct as the primary intent of the query. This is consistent with the analysis done in [19] which was performed over a different Web query log and reported 58% of queries having a semantic construct as the primary intent.

Our part-of-speech tag set is based on the analysis in [2], which is a reduced tag set designed for annotating Web queries. Figure 2 shows the distribution of part-of-speech tags over query tokens as well as the percentage of entity-based queries that contain that part-of-speech. The figure also illustrates the distribution of terms with the given part-of-speech over semantic constructs. This distribution captures the relationship between part-of-speech tags and semantic constructs, which plays a key role in our annotation methods described in Section 3.1. Not surprisingly, there is a very strong correlation between proper nouns and entities. Interestingly, plural nouns are a strong indicator of a term representing a *type*, while regular nouns are split between denoting *type* and *entity* labels.

From the 156 entity-based queries, we also annotated the structured query template underlying our interpretation of the intent of the query. In the cases where there were multiple possible structurings, we annotated all of them and count them as individual queries when computing the frequencies for the equations described in Section 3.

Figure 3 shows the ten most frequent structured query templates in our training query log, along with the percentage of queries exhibiting that structure (over all entity-based queries). Many queries tend to repeat the same structures.

5. EXPERIMENTAL EVALUATION

5.1 System Implementations

We implemented all of the techniques described in Section 3. We use CRF$_{++}$[5] to learn the CRF models and use the tool for mapping structured keyword queries into a KB described in [18] with the modifications described in Section 3.3. We consider the top-10 structures (unless otherwise specified) and union the results of the best KB mapping for each structure. For unary queries (not having multiple terms to perform disambiguation) we employ a heuristic requiring 0.95 syntactic similarity for an interpretation to qualify.

We create two system configurations, the first using the naïve Bayes (NB) term classifier for query annotation and the second using the CRF approach (CRF). Both systems use the direct approach (Drct) to structuring (Section 3.2) and the same KB mapping tool described in Section 3.3 to map the computed structured keyword queries into the knowledge base and evaluate the resulting concept search queries. We omit structures containing more than one unknown relation to avoid the exponential blow-up caused by matching over all possible pairs of relations. We have found this does not have a significant impact on the quality of results. We implement a simple POS-tagger that builds on the Brown and Wall Street Journal POS tagged lexicon. Our tagger assigns the most frequent POS tag for each known word (suitably mapped to our reduced tag set as described in Section 4), and assigns proper noun to unseen tokens (the most frequent tag in our query log). We find this to be sufficiently accurate for our purposes. Building and evaluating a more sophisticated POS tagger is an orthogonal problem beyond the scope of this paper.

For comparison, we also implement two alternative ways of mapping keyword queries into knowledge base entities. The first method implements traditional keyword search over graphs, with a number of heuristics taken from the literature. The process of keyword search over graphs is to find all nodes and edges that match query terms, then search among these *seed* matches to see if they can be connected. A result graph is a subgraph of the data graph that spans all query terms. This can be viewed as an instance of the Steiner tree problem. Our implementation uses a breadth-first search approach to finding Steiner trees. We use the same Lucene index and graph database used by the KB mapping tool. Seed nodes are processed in order of syntactic similarity to the query term they match. The graph search will generate a given parameter number of results and return the k highest scoring. The parameter is set to 10,000 in the effectiveness experiments and is set to 10 in the performance experiments. Scoring is based on the same syntactic scoring (3-gram distance) used by the KB mapping tool, normalized by the answer graph size to promote more compact answers.

The heuristics used in our graph search implementation include traversal of outgoing edges only, incoming edges only, bi-directional edge traversal [10], search depth limiting as recommended in [24] (we use a depth of three), and a last heuristic that forces *type* nodes to be leaves in answer graphs. Note that this system returns subgraphs of the knowledge base, and cannot precisely interpret queries in order to find *specific answers*. In our evaluation, we give this system the benefit of considering any answer graph as correct if it contains the answer anywhere in the graph.

The second comparison approach builds a text representation of each node in the knowledge base, then performs traditional keyword search over the text representations. We use Lucene to create an index over all labels in a one hop radius around each node (including edge labels).

5.2 Data and Workload

We use YAGO [22] as our knowledge base, a high quality fact collection with a rich type hierarchy over entities and a part of the Linked Data Web. Our training data is the Yahoo query log described in Section 4. To evaluate our approaches, we use both queries from the query log and a hand-crafted workload designed with specific properties we wish to evaluate. The query workload consists of 96 queries, half of which have an underlying query intention that *can* be modeled with the data in YAGO, and half taken from the Yahoo log that describe data *not* occurring in YAGO. The half that have YAGO answers, called *positive queries*, are created to exhibit the 12 most frequent structures found in our analysis, accounting for over 90% of all entity-based queries. We create four examples of each of these structures. The second half are entity-based queries taken from the Yahoo log for which we have manually verified that no interpretation exists in YAGO. We refer to these queries as *negative queries*.

The gold standard result for positive queries is defined by manually constructing a structured query over YAGO, and the correct answer for negative queries is defined as the empty set. This query workload explicitly exercises both of the problems of interpreting queries when possible, and recognizing when an interpretation is not possible. We manually create the queries to ensure all structures are represented, and to control for KB coverage. Finding examples of queries with all of the desired structures in a Web query log, such that they also have intentions existing in YAGO would require a huge manual effort, and possibly a larger query sample. Existing benchmarks also do not capture all of the structures we want to evaluate while controlling for KB coverage. The positive queries from the workload are available online[6].

5.3 Effectiveness Results

To evaluate effectiveness we conducted a direct evaluation of the query annotations, an evaluation of the computed KB interpretations, and an end-to-end evaluation of the full system in finding exact answers to keyword queries.

Precision is defined as the fraction of returned results that are correct. Recall is defined as the fraction of all possible correct results that get returned. MRR measures the (reciprocal) average rank where the (first) correct answer occurs. Given a set of queries $\mathcal{Q}$ and a function $rank(i)$ that returns the rank of the first correct answer for query i, MRR is given by the following.

[5] http://crfpp.sourceforge.net/

[6] http://cs.uwaterloo.ca/∼jpound/cikm2012/workload.txt

	Avg. Query Recall			Avg. Token Recall		
System	k=1	k=5	k=10	k=1	k=5	k=10
CRF	0.461	0.703	0.832	0.626	0.797	0.923
NB	0.432	0.500	0.547	0.616	0.711	0.736

Figure 4: Average number of queries with the correctly annotated query occurring in the top-k annotations (query recall), and average number of correctly annotated tokens for the best annotation occurring in the top-k (token recall).

$$\text{MRR}(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|}\sum_{i}^{|\mathcal{Q}|}\frac{1}{rank(i)}$$

For all measures, we count a value of 0 if a system does not return any result for a positive query or if any result is returned for a negative query.

5.3.1 Annotation Accuracy

Figure 4 shows the results of both the NB and CRF annotation approaches. Results are averages using 10-fold cross validation over the 156 entity-based queries from the Yahoo query log. We measure average query recall, the frequency in which the top-k annotations of a query generates the correct complete annotation. The figure also shows average token recall for the best annotation in a query's top-k annotations. This is the fraction of query terms annotated correctly for the best annotation produced for each query. This is relevant because our structuring and KB mapping tools are tolerant to errors, meaning an incorrect segmentation or annotation can still produce a correct concept search query. While both approaches perform similarly at k=1, the superiority of the CRF approach is clear as we consider the top-5 to top-10 annotations.

5.3.2 Query Interpretation

There are four possible outcomes when interpreting a query. The first is that the query is interpreted over the KB, and the interpretation is correct (true positive); the second is that the query is interpreted but the interpretation is incorrect (false positive); the third is that the query does not get interpreted though an interpretation does exist in the data (false negative); and the last is that the query does not get interpreted because no interpretation exists in the data (true negative). Figure 5 shows the confusion matrices for these outcomes for both the CRF and NB approaches when considering the top-10 structures over the workload outlined in Section 5.2. The figure shows that the CRF approach is superior with respect to both types of error, and exhibits a higher overall true positive rate (correct interpretations) than the NB approach. Overall the CRF-based approach produces a correct interpretation or recognizes an uninterpretable query almost 93% of the time. We found the CRF's ranked list of query interpretations to have an MRR of 0.698 on the positive query set. This means that the correct interpretation is found generally in the first or second position on average. The NB approach had an MRR of 0.589, also slightly better than the second position on average.

5.3.3 Keyword Query Answering

We used the workload outlined in Section 5.2 to evaluate the ability of the proposed approaches to interpret and answer keyword queries. Figure 6 shows the results of each of the systems, as well the best run for each measure by *any* of the graph search system's configurations. The superior CRF-based annotator paired with the the semantic summary-based structuring approach has the best performance across all metrics. While the NB approach is comparable, the improved annotation accuracy of the CRF approach yields better results. The graph search approach achieves reasonable recall and MRR, meaning it can find many correct answers and return good answers at decent ranks, however the answer set also becomes polluted with many non-relevant results hurting precision. The approach has no capacity to determine which results are proper answers and which are coincidental keyword matches. Also, the graph search is given the advantage of not having to find the precise answer to queries (any answer graph containing the query answer is considered correct). Not surprisingly, the text graph approach has very low precision as it is designed to find resources related to the query terms and does not attempt to interpret queries in order to find exact answers. Precision is never higher than 0.1 for any value of k. While the unbounded text graph approach achieves 97% recall on the positive query set, it has no means of detecting the negative queries and answers each incorrectly, bringing down the average considerably.

CRF.Drct	f (Incorrect)	t (Correct)	Total +/−
+ (Interp)	3 (3.1%)	45 (46.9%)	48 (50%)
− (No Interp)	4 (4.2%)	44 (45.8%)	48 (50%)
Total f/t	7 (7.3%)	89 (92.7%)	

NB.Drct	f (Incorrect)	t (Correct)	Total +/−
+ (Interp)	3 (3.1%)	36 (37.5%)	39 (40.6%)
− (No Interp)	12 (12.5%)	45 (46.9%)	57 (59.4%)
Total f/t	15 (15.6%)	81 (84.4%)	

Figure 5: Confusion matrices for the query interpretation task. Raw counts and percentages are shown for the true/false positive/negatives, as well as the marginals. The 96 query workload contains 48 positive queries and 48 negative queries.

System	Prec.	Recall	MRR
CRF.Drct	0.789	0.819	0.825
NB.Drct	0.722	0.784	0.768
graph search	0.516	0.682	0.644
text graph, k = 10	0.034	0.178	0.118
text graph, k unbounded	0.000	0.486	0.125

Figure 6: Average Precision, MRR, and Recall over the combined positive/negative query workload.

5.4 Efficiency Results

Precision and recall can be traded off for run-time performance by varying the number of candidate structures. Figure 7 shows the effect of varying the number of structures considered from 1 to 10 for the CRF.Drct approach. As the data points move from left to right, the total number of structures is increasing. This gives an increase in precision and recall as we explore more of the search space to find the correct query interpretation, but at the expense of greater

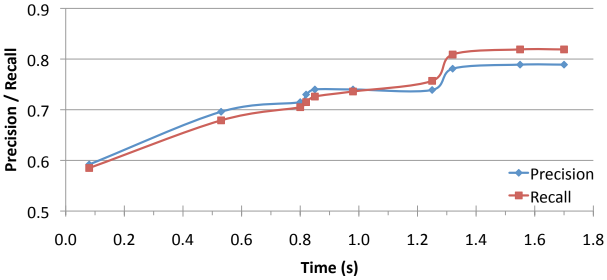

Figure 7: Average precision and recall vs. average run time for varying number of KB interpretations.

run times. Occasionally an incorrect interpretation is found causing a dip in precision. It is important to note that the lower precision configurations of our system are most often due to queries *not being answered*, as opposed to being answered incorrectly. The highest precision (0.789) is obtained for a configuration that takes 1.5 seconds per query on average. An average precision of approximately 0.74 can be achieved in sub-second query processing time.

Figure 8 shows a breakdown of the average run-time for each component of the system when considering the top-10 structures. The actual annotation and structuring time accounts for only a small percent of the total run time, with the majority of time spent mapping candidate structures into the knowledge base. The graph search approach (k = 10) took around 67 seconds on average, often hitting an imposed two minute timeout. This is often due to the graph search attempting to walk the entire search space if it is unable to find k results. The text graph approach is very efficient at $k = 10$, but at the cost of effectiveness.

6. RELATED WORK

Tran et al. [24] have explored mapping keyword queries into conjunctive queries by matching terms against knowledge base items and searching for connections among the matches. This approach is similar to our graph search approach used in the evaluation, though their top-k algorithm may explore less of the total search space than our BFS approach. Zhou et al. have also proposed a similar data driven approach [26]. These techniques are in contrast to our query-log driven approach which elicits structures from the information needs of real users. Lei et al. use manually defined templates to map query terms into formal structured queries over a knowledge base, and employ heuristics to reduce the search space when the query contains more than two terms [13]. Fagin et al. [5] use user specified grammars and vocabularies to annotate query terms with type and attribute labels. Our proposed approach could possibly be used to learn these types of grammars from a query log.

Many recent works have also looked at annotating individual terms in a keyword query with various levels of semantic meaning, such as part-of-speech tags [2], and attribute labels from a relational schema [20]. These works do not address how terms relate to form single coherent expressions of the underlying query intent. Manshadi and Li have also proposed query interpretation using pre-defined grammars. They also annotate terms that specify operations such as "SortOrder" [17]. Li proposed a method of determining the *intent head* of a query, the part of the query that represents what the user is looking for (e.g., "songs" in our example query) [14]. Our approach implicitly identifies the intent head of the query by the root of the concept search query.

System	Struct	Map	Exec	Total
CRF.Drct	0.002s	1.23s	0.32s	1.55s
NB.Drct	0.001s	0.94s	0.29s	1.24s
graph search	-	-	-	67.27s
text graph, $k = 10$	-	-	-	0.33s
text graph, k unbounded	-	-	-	17.33s

Figure 8: Avg. time for annotation and structuring (Struct), KB mapping (Map) and execution (Exec).

Agarwal et al., mined query templates from a query log [1]. Templates would generalize terms that match KB items, for example *"jobs in #location"*. These templates were used to find relevant websites based on clicks. Cheng et al. allowed user queries to specify a similar template, which explicitly specifies the intended return type [4]. Our template mining is similar in nature that of Agarwal, though our templates fully specify semantic structure and not just types occurring in keyword queries. Also, we do not rely on input query terms exactly matching a set of known types, but instead match against syntactically similar types and resolve ambiguity from the query context. Our Web knowledge base setting also yields over 250,000 possible types.

Search over relational data can be viewed as a form of semantic query understanding. Query terms are mapped into structured data producing structured queries that can be used to retrieve data that contains the queried terms. There is a long line of research for search over structured data, starting with the DISCOVER [8] and BANKS systems [9] that aim to efficiently find tuples that span all query terms, as well as extensions that aim to integrate IR style ranking [7] or augment the graph traversal algorithm [10]. The heuristics used in our graph search approach are based on this line of research. Tata et. al, extend keyword search over structured data to include aggregates and top-1 queries [23]. Queries are translated to SQL with the allowance of various aggregation keywords such as *num* or *avg*.

Blanco et al. have used relevance based retrieval to search for nodes in large knowledge bases [3]. This approach however does not attempt to interpret the semantics of queries, and thus does not necessarily provide exact answers.

Semantic parsing of natural language questions shares a similar high level goal with semantic query understanding. Both tasks are to construct a logical representation of a query. Semantic parsing aims to parse natural language queries to lambda calculus expressions of the intention (see for example [15]). Semantic parsing approaches depend on fully specified domain-relevant questions, as opposed to short underspecified keyword queries. Also, these approaches are applied to small controlled domains with relatively small homogeneous knowledge bases, as opposed to the massive heterogeneous knowledge bases found on the Web.

Exploiting Web knowledge bases for question answering (QA) has also been explored. Fernandez et al. propose the PowerAqua system [6, 16] that integrates a front-end QA system on a semantic search back-end to answer questions over Linked Data knowledge bases. Katz et al. have built the OMNIBASE+START system that answers questions using a knowledge base extracted from the Web [11]. While QA has a similar high level goal to our task, QA approaches generally depend on either having properly formatted questions that can be parsed to take advantage of the grammatical structure, or they depend on having the relatively large amount of text (as compared to keyword queries) from the question in order to match against answer candidates.

7. CONCLUSIONS AND FUTURE WORK

We have shown how keyword queries can be interpreted over large scale heterogeneous Web knowledge bases by learning semantic structures from an annotated query log.

Future directions include exploring refinements on the granularity of semantic annotations. For example, distinguishing locations from other types of entities could provide additional structuring hints, since locations often appear as context information in keyword queries. We are also interested in applying these techniques to other data sets and search verticals such as product and medical knowledge bases.

8. REFERENCES

[1] G. Agarwal, G. Kabra, and K. C. Chang. Towards rich query interpretation: walking back and forth for mining query templates. In *Proc. 19th intl. conference on World wide web*, pages 1–10. ACM, 2010.

[2] C. Barr, R. Jones, and M. Regelson. The linguistic structure of english web-search queries. In *Proc. Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1021–1030. ACL, 2008.

[3] R. Blanco, P. Mika, and S. Vigna. Effective and efficient entity search in rdf data. In *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 83–97, 2011.

[4] T. Cheng, X. Yan, and K. C. Chang. EntityRank: searching entities directly and holistically. In *VLDB*, pages 387–398, 2007.

[5] R. Fagin, B. Kimelfeld, Y. Li, S. Raghavan, and S. Vaithyanathan. Understanding queries in a search database system. In *Proc. twenty-ninth ACM symposium on Principles of database systems*, PODS '10, pages 273–284. ACM, 2010.

[6] M. Fernandez, V. Lopez, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells. Semantic search meets the web. In *Semantic Computing, 2008 IEEE intl. conference on*, pages 253 –260, Aug. 2008.

[7] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *Proc. 29th intl. conference on Very large data bases*, pages 850–861. VLDB Endowment, 2003.

[8] V. Hristidis and Y. Papakonstantinou. Discover: keyword search in relational databases. In *Proc. 28th intl. conference on Very Large Data Bases*, VLDB '02, pages 670–681. VLDB Endowment, 2002.

[9] A. Hulgeri and C. Nakhe. Keyword searching and browsing in databases using BANKS. In *Proc. 18th intl. conference on Data Engineering*, ICDE '02. IEEE Computer Society, 2002.

[10] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Proc. 31st intl. conference on Very large data bases*, VLDB '05, pages 505–516, 2005.

[11] B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. J. Lin, G. Marton, A. J. McFarland, and B. Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *NLDB*, pages 230–234, 2002.

[12] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Eighteenth intl. conference on Machine Learning*, ICML '01, pages 282–289, 2001.

[13] Y. Lei, V. S. Uren, and E. Motta. SemSearch: a search engine for the semantic web. In *EKAW*, pages 238–245, 2006.

[14] X. Li. Understanding the semantic structure of noun phrase queries. In *Proc. 48th Association for Computational Linguistics*, ACL '10, pages 1337–1345. ACL, 2010.

[15] P. Liang, M. I. Jordan, and D. Klein. Learning Dependency-Based compositional semantics. In *ACL*, pages 590–599, 2011.

[16] V. Lopez, M. Fernndez, E. Motta, and N. Stieler. PowerAqua: supporting users in querying and exploring the semantic web content. *Semantic Web Journal*, 2011.

[17] M. Manshadi and X. Li. Semantic tagging of web search queries. In *Proc. Joint Conference of the 47th ACL and the 4th Intl. Joint Conference on Natural Language Processing*, pages 861–869. ACL, 2009.

[18] J. Pound, I. F. Ilyas, and G. Weddell. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *Proc. 2010 intl. conference on Management of data*, SIGMOD '10, pages 423–434. ACM, 2010.

[19] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *Proc. 19th intl. conference on World wide web*, WWW '10, pages 771–780. ACM, 2010.

[20] N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of web queries. In *Proc. 2010 intl. conference on Management of data*, SIGMOD '10, pages 771–782. ACM, 2010.

[21] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. 2003 Conference of the North American Association for Computational Linguistics*, NAACL '03, pages 134–141, 2003.

[22] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge - unifying WordNet and wikipedia. In *16th Intl. World Wide Web Conference (WWW 2007)*, pages 697–706, 2007.

[23] S. Tata and G. M. Lohman. SQAK: doing more with keywords. In *Proc. 2008 ACM SIGMOD intl. conference on Management of data*, SIGMOD '08, pages 889–902. ACM, 2008.

[24] T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th intl. conference on*, pages 405–416, 2009.

[25] Yahoo! Academic Relations. http://webscope.sandbox.yahoo.com/catalog.php - L13 - Yahoo! Search Query Tiny Sample.

[26] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu. SPARK: adapting keyword query to semantic search. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 694–707. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-76298-0_50.