Question Answering, Semantic Search and Data Service Querying

Silvia Quarteroni
DEI - Politecnico di Milano
via Ponzio 34/5, 20133 Milan, Italy
quarteroni@elet.polimi.it

Abstract

Question Answering (QA) systems have profoundly evolved since their inception as natural language interfaces to databases. QA technology has indeed become state-of-the-art on open-domain, unstructured information retrieval and more recently touched the Semantic Web and the problem of querying structured data (e.g. RDF triples) on the Web. A natural new challenge is QA over data services, supporting simple natural language query interfaces to the composition of such services for extracting complex results. This paper discusses the above challenges and illustrates natural language QA over data services with a concrete example.

1 Introduction

Question Answering systems, originally designed as natural language interfaces to databases (Waltz, 1978), have nowadays evolved towards the open domain. This involves supporting natural language questions of arbitrary complexity (requiring e.g. definitions, explanations or complex factoids) and extracting answers from potentially unlimited data sources (Kwok et al., 2001).

Information Retrieval (IR) on unstructured data, e.g. Web documents and large textual collections, has been widely researched and is now a mature technology, partly thanks to evaluation campaigns such as TREC[1]. The same goes for QA systems, which are currently able to solve complex questions by exploiting search engines as their information retrieval modules (see e.g. (Moschitti and Quarteroni, 2010; Delpech and Saint-Dizier, 2008)).

On the other hand, recent work in the Semantic Web community has brought to the (semi-)automatic creation of semantic information from partially structured Web resources such as Wikipedia[2], resulting in wide-coverage knowledge bases like YAGO (Suchanek et al., 2007). This in turn has promoted the notion of *semantic search*, which may be defined as IR augmented with semantic information in the purpose of reasoning about the concepts involved in queries and answers (Fazzinga and Lukasiewicz, 2010). Also in the Semantic Web area, natural language interfaces to ontologies have been proposed in a number of studies as an alternative to keyword-based interfaces or interfaces based on query languages (Damljanovic et al., 2010b; Kaufmann and Bernstein, 2007). Generally speaking, NL interfaces to ontologies attempt to perform an exact mapping of the NL query into a logical formula in order to access knowledge structured in e.g. RDF triples.

Even more recently, the exponential growth of data providers on the Web has made proprietary data increasingly available through Web APIs, e.g. Google Places[3], and/or search-specific languages, e.g. the Yahoo Query Language[4]. Data sources are usually wrapped as data services, specified by input and output parameters; normally, the body of data services includes queries and the output is a data collection, with a structured or semi-structured schema.

Enabling natural language interfaces to data services is still an ambitious goal towards which there is little research; this paper analyze aspects of the above-mentioned technologies that bring this objective closer. It also illustrates ongoing work within the Search Computing project[5] that aims at accessing powerful result composition and ranking infrastructures via natural language multi-

[1]trec.nist.gov

[2]wikipedia.org

[3]code.google.com/apis/maps/documentation/places

[4]developer.yahoo.com/yql

[5]search-computing.it

domain questions.

The paper is organized as follows. Section 2 introduces a number of dimensions of question answering challenges; Section 3 discusses QA towards data services; Section 4 describes an approach towards natural language interfaces to data services. Finally, Section 5 illustrates a use case supporting natural language query processing and Section 6 concludes.

2 Questions: from documents to services

Table 1 reports a small snapshot of state-of-the-art information retrieval along two dimensions: query language (logical vs natural) and data sources (unstructured documents, ontologies and services). While querying unstructured documents in natural language offers challenges in terms of the difficulty of specific types of questions (e.g. definitions (Moschitti and Quarteroni, 2010) and procedures (Delpech and Saint-Dizier, 2008)) and the robustness/confidence of information extraction from text, ontologies and data services offer different types of issues. On the one hand, information is structured and may be retrieved with a high degree of confidence, however the main problems deal with interfacing with data sources, i.e. mapping the user's query into a logical format, and possible with executing such a query by composing different data services. Sections 2.1 and 2.2 deal in particular with the issues of performing QA over ontologies and data services.

2.1 Semantic Search

Question Answering over semantic information has been proposed in recent years (McGuinness, 2004) as a natural consequence of the development of Semantic Web technologies. QA is indeed agreed to be one of the ultimate objectives of Semantic Search (Fazzinga and Lukasiewicz, 2010), encouraged by the fact that a number of user studies have found natural language queries to be the most user-friendly and time-efficient input format to semantic information (Kaufmann and Bernstein, 2007; Damljanovic and Bontcheva, 2009). However, open domain QA on this type of resources is far from being a consolidated discipline, partly due to the gap between well-defined semantic resources as can be found for specific domains and the unstructured nature of Web information at large.

Another important issue is the nature of interaction with a semantic QA service, as performing the lexical to semantic level mapping needed to interpret natural language queries into combinations of domain concepts is an only partially solved problem. Typical approaches in this direction involve a combination of statistical techniques (syntactic parsing) and semantic operations to identify ontology concepts in the user's input. For instance, QUERIX (Kaufmann et al., 2006) combines the Stanford parser (Klein and Manning, 2003) with WordNet[6] to obtain RDF triples from natural language user queries, while PANTO (Wang et al., 2007) translates the syntactic parse tree of a natural language query into SPARQL by exploiting a reference lexicon. This often involves user interaction to support the system's interpretation, as in (Damljanovic et al., 2010b).

A crucial question is to figure out whether open domain QA techniques, which leverage statistical methods and require minimal (if any) intervention at design stage, can be successfully combined with semantic search techniques, in order to leverage the benefits of both. In this perspective, the development and widespread usage of vast knowledge bases such as DBPedia[7], GeoNames[8] and YAGO (Suchanek et al., 2007) demonstrates that semantics can encompass universal vocabularies and domains, opening the way to open-domain search.

2.2 Querying Data Services

Recent initiatives such as the W3C Linked Open Data[9] community project are fostering the adoption of Linked Data (LD) as a best practice. Clearly, the widespread presence of data services makes them a valuable resource for IR; however, the problems typically addressed in this field concern service discovery (Carenini et al., 2008), automatic composition (Martin et al., 2007; Fensel et al., 2011) and mediation (Manolescu et al., 2005) rather than the issue of interfacing to data services.

In particular, natural language interfaces are still at an early stage: for instance, (Lim and Lee, 2010) propose a mapping of natural language query blocks to services using predefined workflow templates, however it may be generally said that "core NLP" methods are still far from the data service querying problem. Section 3 reports

[6]wordnet.princeton.edu

[7]`www.dbpedia.org`

[8]`http://www.geonames.org/ontology`

[9]`esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`

Table 1: Querying documents, ontologies, services

Query Language	Documents	Ontologies	Data services
Logical	-	Semantic search (Fazzinga and Lukasiewicz, 2010)	LOD, YQL, OWL-S (Martin et al., 2007)
Natural	Open-domain QA (Moschitti and Quarteroni, 2010; Delpech and Saint-Dizier, 2008)	QUERIX (Kaufmann et al., 2006), PANTO (Wang et al., 2007), (Damljanovic et al., 2010b)	(Lim and Lee, 2010), SeCo (Bozzon et al., 2011b)

progress in the direction of open-domain QA over data services.

3 Towards QA over Data Services

In order to conduct QA over data services, the semantics of the latter needs to be aligned to the semantics of natural language interpretation; such a mapping should be carried out with minimal human contribution, maximal domain coverage, and a high level of robustness. Ideally, natural language interpretation of the user's question should map the user's intent (expected answer type in QA terminology) and the main concepts mentioned in the question to the same knowledge representation used by service interfaces to execute queries over their respective data services.

To this end, we have been working on a pragmatic approach to the description of data services in the SeCo project (Bozzon et al., 2011a), that aims at efficiently and effectively composing data services in order to address complex queries. Here, the infrastructure supports the efficient execution of queries over service compositions and different ranking schemes are also deployed for result presentation flexibility. The models used for service description and annotation in the purpose of querying are described in Section 3.1, while querying itself is described in Section 3.2.

3.1 Pragmatic Service Annotation

We model data services at different levels of abstraction according to a three level Service Description Framework (SDF). The topmost conceptual level, denoted as *service mart* level, represents the domain entities described by services (e.g. **Theater**).

Then, the logical view sees possible implementations of data services focused on specific entities in terms of *access patterns* (APs), i.e. relations between the I/O parameters they involve. For example, an AP returning movies shown near the user's current position would be named `GET Movie WITH Theater BY CurrentPosition`.

Finally, at the physical level, *service interfaces* encode the actual interaction protocol; different service interfaces may be associated to the same AP, e.g. a YQL service returning US movies based on the user's position or a wrapper to the Google movies service returning Italian movies and theaters by location. The above three views compose the so-called Service Description Framework (SDF).

Access Pattern information from data services is used for building a common *domain diagram* (DD), for which we use a simple Entity-Relationship model. Each data service is "focused" upon a DD entity, has a schema which includes several related entities, and is linked to other data services through relationships. Figure 1 illustrates a sample DD deriving from services dealing with movies and theaters.

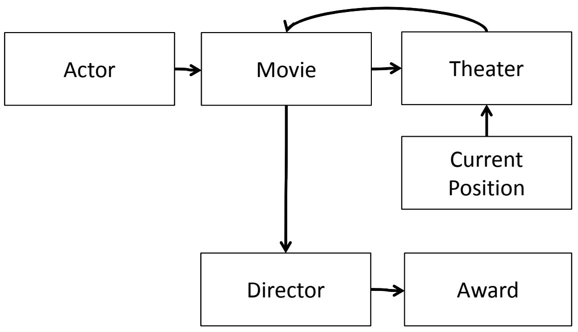

Figure 1: A sample Domain Diagram

One or more existing *knowledge bases* (KBs) are used as a reference for DD terminology and entities in order to validate it and support natural language querying; for instance, Section 4 shows an example of usage of YAGO (Suchanek et al., 2007) as the reference KB.

In addition to the DD, a *knowledge base proxy* (KBP) is constructed by using the reference KB to extract the most relevant concepts for describing the DD and reasoning upon those concepts.

The SDF is progressively populated in a bottom-up fashion by processing service interfaces available from data service providers and devising the corresponding APs by identifying the relevant DD entities they involve. In case no suitable entities are found in the DD, the reference KB is used as a source of entities that are immediately "projected" over the DD. Once access patterns are created and their focal entity is identified, the corresponding service mart is the one identified by such a focal entity.

3.2 Query Processing

Once available services have been registered following the method in Section 3.1, they can be queried in a variety of ways, including via Graphical User Interfaces or via textual input. In all cases, the general processing of a query over the registered data services is organized according to three steps, as summarized in Figure 2.

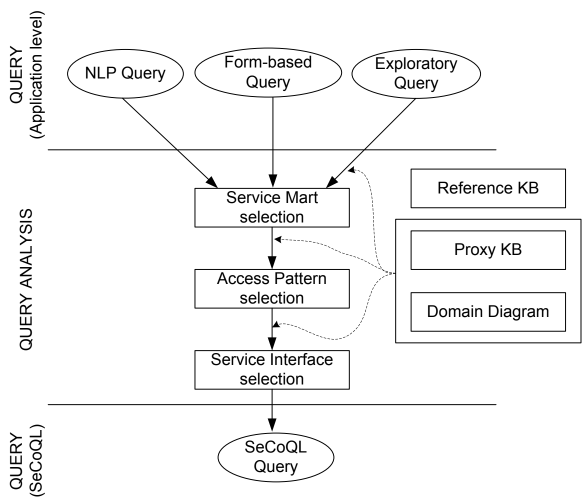

Figure 2: Top-down query processing steps over the Service Description Framework, with the contribution of domain semantics encoded in the Domain Diagram, the Knowledge Base Proxy and the reference Knowledge Base.

Starting from the queries submitted by the user in various formats at the application level, the steps progressively "formalize" the semantics of such queries at the conceptual, logical and physical level, mapping their terms into objects of the Service Description Framework. In brief, the core entities must be identified in the user's query in order to locate relevant service marts; once the latter are known, a suitable access pattern must be chosen for each service mart in order to fill in the query's input/output parameters; finally, a physical service implementation must be chosen to route the query to the appropriate data service.

As SeCo allows for the composition and concatenation of different data services, constraints must be formulated according to how APs are combined. These are expressed in a specific query format named SeCoQL (Braga et al., 2011), an SQL variation supporting queries over data services. A SeCoQL query is the final outcome of the query process in Figure 2.

Section 4 illustrates how a natural language query may be mapped to a logical query language expression invoking service interfaces and APs that refer to the Semantic Annotation Framework.

4 Natural Language Service Querying

Addressing a natural language (NL) query over the service registration architecture outlined in the previous section implies the difficulty of generalizing over text by mapping the lexical level to the conceptual representation of the domain at hand. We represent natural language query processing as a three-step process: first, a (probabilistic) matching is made between each query's syntactic focus and a focal DD entity connected to a service mart; then, a number of refinement operations allow to obtain the I/O parameters of APs; finally, AP parameters are mapped into lower-level service interface parameters to complete query specification. We summarize these steps as intent classification, access pattern selection and service interface selection, as described in the remainder of this section.

4.1 Step 1: Intent Classification

The processing of a natural language query starts with intent classification, i.e. a categorization of the user's query in terms of eligible service marts. Such a query categorization method must be able to decompose an arbitrarily complex, multi-domain query q into N subqueries q_i, $i \in \{1, .., N\}$. Then, it must map each q_i into a class c_j from the set of all query classes C, such that each c_j is mapped to a service mart in the conceptual model – or *other* to account for uninterpretable input.

Query decomposition can be performed in many ways; for instance, a method based on syntactic parsing will determine a sentence's subclauses as well as their syntactic relationships (coordination or subordination).

Once subqueries are identified, the next step toward intent classification is the identification of the question's syntactic foci, i.e. its salient words/phrases, each of which is to be later matched to one of the available service marts. This is performed according to domain-independent criteria based on morphological and/or (shallow) syntactic analysis (Li et al., 2009; Damljanovic et al., 2010a). This process can be re-conducted to the question classification phase in open-domain Question Answering, which consists in estimating the most likely expected answer type from a domain-independent taxonomy – see e.g. (Li and Roth, 2002).

A pilot experiment To investigate intent classification, we ran a pilot experiment in question focus identification by comparing a number of existing focus extraction algorithms on a collection of 50 spontaneous user queries dealing with movies, theaters, lodgings and events, collected within the SeCo project. Queries in the collection have a single focus and do not require an initial decomposition, e.g. "where can I stay for one night close to cinema Plinius?". On the latter corpus, we have reached a recall of 69% when applying the state-of-the-art focus identification algorithm proposed by (Li et al., 2009). In addition, we were able to observe that, out of the 33 correctly identified foci, 14 could directly be mapped to a specific service mart (e.g. "movie", "events"); of the remaining queries, 5 referred to a synonym of a service mart identifier (e.g. "cinemas", a synonym of *Theater*), while 3 referred to subclasses of an identifier (e.g. "exhibition" is a type of *Event*). For all of the above cases, the presence of a KB indexing instances and preferred meanings of such terms has enabled the lexical to conceptual mapping. The 9 remaining foci referred to terms amenable to a specific service mart only after context disambiguation, e.g. "room" or "place", which both refer to the *Hotel* service mart; again, such context disambiguation is feasible by looking at KB properties of concepts around the focus entity.

This suggests that the accuracy of mapping foci to service marts is greatly dependent on both the precision of focus extraction and the ability to generalize and perform shallow reasoning abilities over the DD and KB; a machine learning approach may provide a more robust result than simple rule-based mapping in case a representative dataset is available.

4.2 Step 2: Access Pattern Selection

Once focal entities are identified, the subsequent step in NL query processing consists in selecting the AP that best responds to the user information needs for each focus. One possibility is to make a hypothesis about the best AP to represent the DD entities/attributes extracted so far from the user's query and then interacting with the user for obtaining the missing information. The best match with respect to the user's query is computed by selecting the AP that:

1. uses all the inputs provided by the user or by other outputs of previously selected APs;
2. requires the minimum number of additional inputs with respect to the ones provided by the user;
3. produces the maximum number of outputs with respect to the ones explicitly requested by the user;
4. connects properly to the other entities requested by the user through DD relationships.

Information from the Knowledge Base Proxy can be valuable in the entity extraction phase; query terms can be associated with specific entities or attributes by looking at KBP indexes; in particular, terms such as "Times square" or "Central Park" help situating the query's location within New York, and terms such as "Angelina Jolie" or "Brad Pitt" are mapped to actor's names.

KBP relationships can also be very useful in AP selection; a notable case is the use of inheritance associated with IS-A relationships. Assume that one of the query foci is entity E_1 and that we cannot find a suitable AP of E_1 for translating the query, but then assume that entities E_1 and E_2 are linked by an IS-A relationship; then, we may also consider those of E_2 as suitable APs for the query. For example, considering that the IS-A relationship holds in YAGO between *Hostel* and *Hotel*, if we recognize hostel instances such as ("inn", "ymca") in the query, then we may consider using an AP focused on the *Hotel* entity.

Generally speaking, the eligibility of an AP may be formalized as a function giving different weights to the mappings from AP parameters to query terms; for instance, mappings using DD concepts are preferable to mapping using KBP concepts, while use of semantic relationships such as IS-A are weighted by introducing suitable semantic distances, e.g. to a common subsumer node (Jiang and Conrath, 1997).

4.3 Step 3: Service Interface Selection

After APs are selected, each AP should be associated with exactly one service interface. Since by construction service interface parameters and AP parameters match directly, selection at this level is based on additional context knowledge, that can be inferred from the query, for instance by looking for the best implementation that covers the instances used in the query. This can be obtained by exploiting the defined selectors and the Knowledge Base Proxy; for instance, suppose that the user is looking for movies in San Francisco. If the available SIs cover Canada, the US, and the West Coast respectively, one can exclude the first one and prefer the third one over the second one. We next exemplify natural language question processing within a QA use case.

5 A Question Answering Scenario

Let us assume that our Semantic Annotation Framework consists of the Domain Diagram in Figure 1 and YAGO as a reference knowledge base. We suppose that the natural language query q = "Where is a theatre that shows an action movie near Times Square?" is issued by via a search engine-style interface. To understand q, we first need to process it to extract relevant domain information, i.e. its entities and attributes; then, a logical query must be formulated to represent the semantics of q in terms of service marts, APs and service interfaces.

The entity/attribute extraction phase makes high usage of the KBP, which in turns contains YAGO resources supporting the identification of landmarks such as "Times Square", and the instance index in each KBP attribute which allows to directly match a natural language string with the instance of a specific DD entity attribute (e.g. "Times Square" as a possible value for the DD attribute *CurrentPosition.address*). Similar analysis may be used to recognize "action" as a value for the *genre* attribute of entity *Movie*. As a result, we are able to annotate q with DD attribute values, having:
q_{ann} = "Where is a theater that shows an *Movie.genre*[action] movie near *CurrentPosition.address*[Times Square]?"

Syntactic analysis conducted via the Stanford parser (Klein and Manning, 2003) will return the tree in Figure 5, which we name q_{tree}.

Note that q_{tree} contains a main clause ($main$) "Where is a theater" with a clearly distinguishable sub-clause (sub) "that shows an action movie near Times Square", introduced by $\boxed{\textbf{SBAR}}$.

Now, a focus extraction approach such as e.g. the one in (Li et al., 2009) allows us to extract the syntactic focus of both $main$ and sub, i.e. "theatre" resp. "movie"; in particular, $main$, whose focus is "theatre", is mapped to the **Theater** service mart thanks to a simple lookup of the KBP that shows that the two words are synonyms (*Theater* is the preferred meaning of the string "theatre" in YAGO).

Next, a suitable AP is identified for each focus entity. Let us assume that the only available AP for **Movie** is

AP1 `GET Movie BY Movie.genre`

returning movies based on their genre, and that the available APs for **Theater** are

AP2 `GET Theater WITH Movie BY CurrentPosition`

AP3 `GET Theater WITH Movie BY Movie.title`

respectively returning theaters based on the current position and based on the titles of movies they show. The annotation of q suggests AP2 as the most eligible AP. In turn, sub, whose focus is "movie", may be directly mapped to the **Movie** service mart; we then select AP1, its only AP.

Finally, joining AP1 and AP2 to fully express the semantics of q and get to a logical query (e.g. the SeCoQL query in Figure 4) requires a number of additional actions.

First, the values of *Movie.title*, appearing as AP2's output and AP1's input, must match (`ON` predicated of the `JOIN` clause in Figure 4). Secondly, the KB can be used to obtain the mapping of "Times Square" to "New York City"; this in turn allows to identify the theater's country via KB. The latter piece of information is used both to infer a query parameter (`WHERE $W='US'` in Figure 4) and to select a service interface for **Theater** associated with US theaters (`USING S2` in Figure 4).

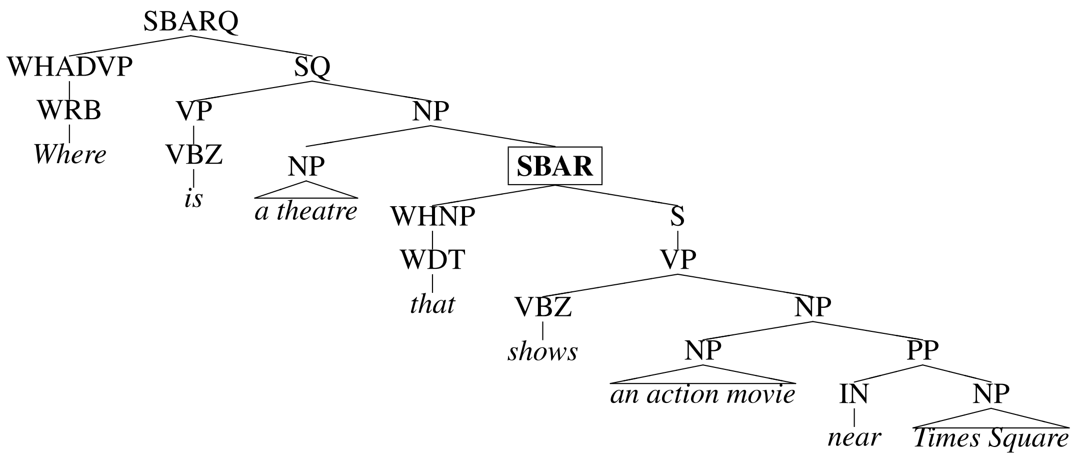

Figure 3: Top syntactic parse tree of the question "Where is a theatre that shows an action movie near Times Square?" according to the Stanford Parser (Klein and Manning, 2003). A subordinate clause is introduced by $\boxed{\textbf{SBAR}}$

```
DEFINE QUERY Cinema($X:String,
              $U:String, $V:String) AS
SELECT M.*, T.* FROM
    AP1(Movie.genre: $X) AS M USING S1
  JOIN
    AP2(Theater.addr: $U, Theater.city: $V,
    Theater.country: $W) AS T USING S2
  ON M.Movie.title=T.Movie.title
WHERE $W='US'
```

Figure 4: SQL-like query representing "Where is a theatre that shows an action movie near Times Square?". `S1` and `S2` denote service interfaces implementing `AP1` and `AP2`, respectively.

6 Conclusions

This paper discusses issues and opportunities offered by Question Answering over data services, which may be regarded as a natural next step of IR with respect to QA over unstructured documents (a consolidated discipline) and over semantic resources (a relatively recent one).

The main issues of QA over data services involve on the one hand the mapping from natural language questions to a semantic representation of the domain and functionalities covered by such services, and on the other the acquisition of the necessary parameters in order to execute physical queries over the latter.

As a possible solution toward this goal, this paper illustrates an architecture that registers service interfaces within an ER domain model where entities, relationships and attributes are sourced from an open-domain universal knowledge base (i.e. YAGO). This approach is motivated by the argument that aligning the semantics of data services with the semantics of natural language queries is the first step towards Question Answering over Web services.

However, the challenge of QA over data services is far from being reached. Future work in this direction chiefly involves 1) efficient strategies (e.g. dialog-based) to acquire service interface parameters from the user in order to feed them to a query execution engine, and 2) effective answer extraction and presentation approaches.

Acknowledgments

Part of the research reported in this paper is in collaboration with Stefano Ceri and Marco Brambilla within the Search Computing project (European Commission IDEAS grant no. 227793).

References

A. Bozzon, D. Braga, M. Brambilla, S. Ceri, F. Corcoglioniti, P. Fraternali, and S. Vadacca. 2011a. Search computing: Multi-domain search on ranked data. In *Proceedings of SIGMOD*.

A. Bozzon, M. Brambilla, E. della Valle, P. Fraternali, and C. Pasini, 2011b. *Integrated Exploration of Linked Data and Semi-structured Web Information*, volume 6585 of *LNCS*. Springer.

D. Braga, M. Grossniklaus, F. Corcoglioniti, and S. Vadacca, 2011. *Efficient Computation of Search Computing Queries*, volume 6585 of *LNCS*, pages 141–155. Springer.

A. Carenini, D. Cerizza, M. Comerio, E. Della Valle, F. De Paoli, A. Maurino, M. Palmonari, and A. Turati. 2008. Glue2: A web service discovery engine with non-functional properties. *Web Services, European Conference on*, 0:21–30.

D. Damljanovic and K. Bontcheva. 2009. Towards enhanced usability of natural language interfaces to knowledge bases. In V. Devedic and D. Gaevic, editors, *Web 2.0 & Semantic Web*, volume 6 of *Annals of Information Systems*, pages 105–133. Springer US.

D. Damljanovic, M. Agatonovic, and H. Cunningham. 2010a. Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In *LREC*.

D. Damljanovic, M. Agatonovic, and H. Cunningham. 2010b. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of ESWC*, pages 106–120.

E. Delpech and P. Saint-Dizier. 2008. Investigating the structure of procedural texts for answering how-to questions. *LREC2008, Marrakech*.

B. Fazzinga and T. Lukasiewicz. 2010. Semantic search on the web. *Semantic Web Journal*.

D. Fensel, F.M. Facca, E. Simperl, and I. Toma. 2011. *Semantic Web Services*. Springer.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics (ROCLING X)*.

E. Kaufmann and A. Bernstein. 2007. How useful are natural language interfaces to the semantic web for casual end-users? In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 281–294. Springer Berlin / Heidelberg.

E. Kaufmann, A. Bernstein, and R. Zumstein. 2006. Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th International Semantic Web Conference (ISWC 2006)*, pages 980–981. Citeseer.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

C. T. Kwok, O. Etzioni, and D. S. Weld. 2001. Scaling question answering to the web. In *Proceedings of WWW*.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL*, pages 1–7. Association for Computational Linguistics.

X. Li, Y.Y. Wang, and A. Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of SIGIR*, pages 572–579. ACM.

J. Lim and K. Lee. 2010. Constructing composite web services from natural language requests. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):1 – 13.

I. Manolescu, M. Brambilla, S. Ceri, S. Comai, and P. Fraternali. 2005. Model-driven design and deployment of service-enabled web applications. *ACM Trans. Internet Technol.*, 5:439–479, August.

D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, Katia S., D. McGuinness, E. Sirin, and N. Srinivasan. 2007. Bringing semantics to web services with owl-s. *World Wide Web J.*, 10:243–277.

D. L. McGuinness. 2004. Question answering on the semantic web. *IEEE Intelligent Systems*, 19:82–85.

A. Moschitti and S. Quarteroni. 2010. Linguistic kernels for answer re-ranking in question answering systems. *Information Processing & Management*, In Press, Corrected Proof:–.

F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

D. L. Waltz. 1978. An english language question answering system for a large relational database. *Communications of the ACM*, 21(7).

C. Wang, M. Xiong, Q. Zhou, and Y. Yu. 2007. Panto: A portable natural language interface to ontologies. In E. Franconi, M. Kifer, and W. May, editors, *The Semantic Web: Research and Applications*, volume 4519 of *Lecture Notes in Computer Science*, pages 473–487. Springer Berlin / Heidelberg.