Using BabelNet in Bridging the Gap Between Natural Language Queries and Linked Data Concepts

Khadija Elbedweihy, Stuart N. Wrigley, Fabio Ciravegna, and Ziqi Zhang

Department of Computer Science, University of Sheffield, UK
{k.elbedweihy, s.wrigley, f.ciravegna, z.zhang}@dcs.shef.ac.uk

Abstract. Many semantic search tool evaluations have reported a user preference for free natural language as a query input approach as opposed to controlled or view-based inputs. Although the flexibility offered by this approach is a significant advantage, it can also be a major difficulty. Allowing users complete freedom in the choice of terms increases the difficulty for the search tools to match terms with the underlying data. This causes either a mismatch which affects precision, or a missing match which affects recall. In this paper, we present an empirical investigation on the use of named entity recognition, word sense disambiguation, and ontology-based heuristics in an approach attempting to bridge this gap between user terms and ontology concepts, properties and entities. We use the dataset provided by the Question Answering over Linked Data (QALD-2) workshop in our analysis and tests.

Keywords: semantic search, natural language query approach, word sense disambiguation, evaluation

1 Introduction

In its broadest sense, *Semantic Search* describes the process of matching document content with user intent. Semantic search tools adopt different query input approaches, ranging from natural language ('free' NL [1, 2] and 'guided' NL [3]) to view-based (forms [4] and graphs [5]) interfaces. Each of these strategies offer the user different levels of flexibility, query language expressiveness and support during query formulation. Evaluations have shown that users appreciate the simplicity, flexibility, and fast query input of the free NL approach [6, 7]. However, flexibility increases the difficulty of the underlying system's task of mapping their linguistic terms onto the correct ontological concepts and properties and Linked Data entities. Core to this difficulty are *polysemy* (single word with more than one meaning) and *synonymy* (multiple words with the same meaning). While the former affects precision of results by providing false matches and the latter affects recall by causing true (semantic) matches to be missed [8], both of them are usually addressed by using a word sense disambiguation (WSD) approach. For example, in order to answer the question "How tall is ...?", the query term *tall* needs to be mapped to the ontology property *height* (a term related to tall). However, the term *tall* is also polysemous and has different senses including (from WordNet):

- "great in vertical dimension; high in stature; tall people; tall buildings, etc."
- "too improbable to admit of belief; a tall story"
- "impressively difficult; a tall order"

Therefore, the term must be disambiguated and the right sense identified (the first in this example), before attempting to gather related terms.

An additional difficulty is Named Entity (NE) recognition and disambiguation. NE recognition approaches include parsing and then matching to resources in the ontology; directly recognising and disambiguating using a state of the art (SOA) NE recogniser [9], or treating them as 'normal' query terms which are mapped to ontology concepts and resources (the approach taken by [2] and [10]). The computational cost of matching NEs to resources in a number of ontologies can increase in proportion to the ontology size. Furthermore, NEs can refer to multiple real world entities thus necessitating NE disambiguation. This is usually performed using the context of the query and the structure of the ontology in which the term occurs. For example, in the question *Which river does the Brooklyn Bridge cross?*, the terms *Brooklyn Bridge* and *Brooklyn* would be mapped to the DBpedia resources `res:Brooklyn_Bridge` describing the bridge, and `res:Brooklyn`[1] describing the city in New York, respectively. The first can be correctly selected based on an ontology structure which shows that the property *crosses* connects a river to a bridge and not to a city.

In this paper, we present a free-NL semantic search approach that bridges the gap between the sense of the user query terms and the underlying ontology's concepts and properties. We use an extended-Lesk WSD approach (Section 2) and a NE recogniser (Section 3.1) together with a set of advanced string similarity algorithms and ontology-based heuristics to match query terms to ontology concepts and properties (Section 3.3). Section 4 presents the evaluation of our approach using the dataset provided by the *Question Answering over Linked Data (QALD-2)* workshop[2].

2 Word Sense Disambiguation (WSD)

Disambiguating polysemous words in user queries is of paramount importance to understand user intent and provide correct results. Indeed, identifying the intended sense of a polysemous word is a necessity for query expansion which is often used by search tools to bridge the gap between user terms and ontological concepts and properties. Some search approaches consider all the senses of a polysemous word and use their related terms for query expansion [10]. However, we believe this increases noise and irrelevant matches and, therefore, affect precision. On the other hand, in the disambiguation approach described in [11], a specific WordNet synset is considered relevant only if one of its senses (separate words in a WordNet synset) exists in the synonyms, hypernyms, hyponyms, holonyms or meronyms of an ancestor or a descendant of the synset.

[1] The prefix `res` refers to: `<http://dbpedia.org/resource/>`

[2] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=challenge&q=2`

Knowledge-based WSD approaches use dictionaries and lexicons such as WordNet to perform WSD and are often considered a middle ground between supervised and unsupervised approaches. Although graph-based approaches have been gaining more attention recently for their high performance [12, 13], in this work, we use an extended-Lesk approach for the following reasons: 1) it was one of the highest performing in knowledge-based approaches (see UPV-WSD in [14]); [15] showed only 2 % difference when compared with the Degree Centrality algorithm, and 2) it is a simpler approach to test our hypothesis of improving the mapping between NL queries and LD ontologies using a WSD approach with high-coverage knowledgebase (BabelNet).

WordNet is the predominant resource used in such knowledge-based WSD approaches; however, it has been argued that its fine granularity is the main problem for achieving high performance in WSD [12, 16]. In light of this, we, alternatively, used BabelNet [12] for disambiguation. BabelNet is a very large multilingual ontology with wide-coverage obtained from the automatic integration of WordNet and Wikipedia. We believe that Wikipedia's richness of explicit and implicit semantic knowledge together with the lexical coverage provided by WordNet as noted by [12] could be very useful for our approach and in general for semantic search to both expand and disambiguate users' queries.

Following [17], we apply an extended-Lesk approach in which various semantic relations are added to the context and synsets' bags to perform the disambiguation. The selected relations provided the highest performance based on our experiments using the SemEval-2007 coarse-grained all-words dataset[3]. These relations – in order of their contribution to the WSD – are: 1) direct hyponyms; 2) second-level hypernyms; 3) synsets' glosses (only the examples part [18]); 4) WordNet semantic relations: *attribute*, *similar to* and *see also*; and finally, 5) glosses of synsets related to the main synset through one of the hyponyms, hypernyms, or semantic relations.

3 Sense-aware Search

Users exhibit a general preference for short NL queries, consisting of keywords or phrases, as opposed to full sentences and a random query structure with no specific order for the query terms [19]. In this section we describe a novel semantic search approach in which free-form natural language queries are processed to establish the user intent and underlying 'meaning' of the query (using word sense disambiguation; see Section 2) allowing the query terms to be more accurately associated with the underlying dataset's concepts and properties. Our approach consists of five stages:

1. Recognition and disambiguation of Named Entities.
2. Parsing the NL query.
3. Matching query terms with ontology concepts and properties.
4. Generation of candidate triples.
5. Integration of triples and generation of SPARQL queries.

[3] `http://lcl.uniroma1.it/coarse-grained-aw/index.html`

3.1 Recognition and Disambiguation of Named Entities

Named entities are recognised using AlchemyAPI[4] which had the best NE recognition performance in a recent evaluation of SOA recognisers [9]. However, AlchemyAPI exhibits poor disambiguation performance [9]; in this paper, each NE is disambiguated using BabelNet as described above.

For example, for the question "In which country does the Nile start?", the term *Nile* has different matches in BabelNet. These matches include:

- http://dbpedia.org/resource/Nile_(singer)
- http://dbpedia.org/resource/Nile_(TV_series)
- http://dbpedia.org/resource/Nile_(band)
- http://dbpedia.org/resource/Nile

Although one could select the last URI as an exact match to the query term, syntactic matching alone can not guarantee the intended meaning of the term, which is better identified using the query context. Using our WSD approach, we would select the correct match for *Nile* as a river since more overlapping terms are found between this sense and the query (such as *geography, area, culture* and *continent*) than the other senses.

3.2 Parsing and Disambiguation of the Natural Language Query

The second step is to parse the NL query, which is done using the Stanford parser [20]. However, since we do not expect users to adhere to correct grammar or structure in their queries we do not use the generated parse trees; we only use lemmatisation and part of speech (POS) tagging. Each query term is stored with its lemma and POS tag except for previously recognised NEs which are not lemmatised. Additionally, we identify the position of each term with respect to the rest of the query and use this in the later step. For example, the question *Which software has been developed by organizations founded in California?* from the QALD-2 dataset generates the following outcome:

- software: at position `1` and POS `NP`
- developed: at position `2` and POS `VBN`
- organizations: at position `3` and POS `NNS`
- founded: at position `4` and POS `VBN`
- California: at position `5` and POS `NP`

Equivalent output is also generated when using keywords or phrases. At the end of this step, any proper nouns identified by the parser and which were not recognised by AlchemyAPI as NEs are disambiguated as described in Section 2 and added to the set of recognised entities. This ensures that, for the example used above: *In which country does the Nile start* the algorithm does not miss the entity *Nile* because it was not recognised by AlchemyAPI.

[4] `http://www.alchemyapi.com`

3.3 Matching Query Terms with Ontology Concepts and Properties

The terms generated from the second step (Section 3.2) are then matched to concepts and properties in the ontologies being used. Noun phrases, nouns and adjectives are matched with both concepts and properties, while verbs are only matched with properties. After gathering all candidate ontology matches that are syntactically similar to a query term, these are then ordered using two string similarity algorithms: *Jaro-Winkler* [21] and *Double Metaphone* [22]. Jaro-Winkler depends on comparing the number and order of common characters. Similar to Monge Elkan [23] which is used by [24], it gives a high score to terms which are parts of each other. This is useful since ontology concepts and properties are usually named in this way: for instance, the term *population* and the property *totalPopulation* are given a high similarity score using this algorithm. An additional advantage of this algorithm is efficiency; [25] found it to be an order of magnitude faster than Monge-Elkan. We set the threshold for accepting a match to 0.791, which was shown by [26] to be the best threshold value. Double Metaphone captures words based on a phonetic basis and is, therefore, useful to capture similarly sounding terms.

If a query term produces no matches, its lemma is used for matching. If no matches were found, derivationally related forms of the query term are then used. For example, the property *creator* in the question *Which television shows were created by Walt Disney?* is only found after getting these forms for the term *create*. After this, if no matches are found, the query term is then disambiguated using our WSD approach and terms related to the identified synset are gathered. These terms are used to find matches in the ontology, based on both their level in the taxonomy (the nearest, the better) and in order of their contribution to the WSD as shown by the results of our experiments. Thus, synonyms are used first, then semantic relations (the appropriate ones), followed by hyponyms, and finally hypernyms. For nouns, no semantic relations are used, while for verbs, *see also* is used and finally, for adjectives, *attribute* and *similar to* are used in that order. Indeed, the attribute relation is very useful for adjectives since, for example, the property *height* is identified as an attribute for the adjective *tall*, which allows answering the question "How tall is ...?". The query term is marked as *not found* if no matches were found after all expansion terms have been used. Note that we do not match superlatives or comparatives to ontology concepts or properties; they are used in the next step to generate the appropriate triples.

3.4 Generation of Candidate Triples

After all terms have gone through the matching process, the query can be interpreted in terms of a set of ontology concepts, properties and instances that need to be linked together. The structure of the ontology (taxonomy of classes and domain and range information of properties) in addition to BabelNet are used in this step, as will be explained next.

Three-Terms Rule Firstly, each three consecutive terms are matched (using the information about their relative positions as explained in Section 3.2) against a set of templates. The intuition is to find a subject with a specified

relation to a given object. Then, the ontology matches associated with each term are used to generate one or more candidate triples. For instance, the question *Which television shows were created by Walt Disney?* which can also be given as keywords *television show, create, Walt Disney* matches the template *concept-property-instance* and generates the following triples:

```
?television_show <dbo:creator> <res:Walt_Disney>.
?television_show <dbp:creator> <res:Walt_Disney>.
?television_show <dbo:creativeDirector> <res:Walt_Disney>.
```

Triples generated from the same query term are ordered according to the similarity of the matches found in them with respect to this term. In this example, the two properties `dbo:creator` and `dbp:creator` are ordered before `dbo:creativeDirector`[5] since they have a higher similarity score with the query term *create*. Similar questions that would be matched to this template include *airports located in California*, and *actors born in Germany*. The other templates capture the different ordering that can be found in the query such as *instance-property-concept* in the question *Was Natalie Portman born in the United States?* or *property-concept-instance* in the question *birthdays of actors of television show Charmed*. Note that in the last example, since we identify the type of the instance *Charmed* as 'television show', we exclude the latter during triples generation making it: *birthdays of actors of Charmed*.

Two-Terms Rule Some user queries contain fewer than three pieces of information, thus preventing the application of the *Three-Terms Rule*. This can happen in three situations: 1) no match between the derived terms and any three-term template; 2) the template did not generate candidate triples; or 3) there are fewer than three derived terms.

For example, the second situations occurs in the second part of the question *In which films directed by Garry Marshall was Julia Roberts starring?* in which the terms *Garry Marshall*, *Julia Roberts* and *starring* would be matched to an existing template but without generating candidate triples. The requirement that the domain of the property (in this case: `Film`) must be the type of one of the instances was not met. In these scenarios, we follow the same process of template matching and triples generation for each pair of consecutive terms. For instance, the question *area code of Berlin* generates the triples:

```
<res:Berlin> <dbp:areaCode> ?area_code.
<res:Berlin> <dbo:areaCode> ?area_code.
```

Comparatives As explained earlier, superlatives and comparatives are not matched to ontology terms but used here to generate the appropriate triples. For comparatives, there are four different scenarios that we found from our analysis of the queries in datasets used by different semantic search evalutions (e.g., Mooney dataset [27] and datasets used in QALD challenges[6]). The first is when

[5] The prefix `dbo` refers to: `<http://dbpedia.org/ontology/>`; the prefix `dbp` refers to: `<http://dbpedia.org/property/>`

[6] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/`

a comparative is used with a numeric datatype property such as the property `numberOfEmployees` in the question phrase *more than 500000 employees*. This information is known from the range of the property. In this case the following triples are generated:

```
?company <dbp:numEmployees> ?employee.
?company <dbp:numberOfEmployees> ?employee.
```

These triples are ordered according to their similarity with the original query term (*employee*) and a choice is made between using the best match or all matches depending on the priority of the algorithm (i.e., whether to favour precision or recall). The chosen triples are then added to the following ones:

```
?company a <dboCompany>.
  FILTER ( ?employee > 500000)
```

The second scenario is when a comparative is used with a concept as in the example *places with more than 2 caves*. Here, we want to generate the same triples that would be generated for *places with caves* and add the aggregate restriction: `GROUP BY ?place HAVING (COUNT(?cave) > 2)`.

In the third scenario, the comparative is used with an object property which, similarly, requires an aggregate restriction. In the example *countries with more than 2 official languages*, the following restriction is added to the normal triples generated between country and official language.

```
GROUP BY ?country HAVING (COUNT(?official_language) > 2)
```

The fourth and most challenging scenario can be illustrated by the question *Which mountains are higher than the Nanga Parbat?*. The difficulty here is to identify the property referred to by the comparative term (which is 'elevation' in this example) to get its value for the given instance and then do a comparison with this value. While [24] tackles this challenge by generating suggestions using the datatype properties associated with the concept and asking the user for assistance, this can be an overhead on the user. Our algorithm tries to select the best relation according to the query context. Firstly, all numeric datatype properties associated with the query concept (in this case *mountain*) are identified. These are: `latS`, `longD`, `prominence`, `firstAscent`, `elevationM`, `latD`, `elevation`, `longM`, `latM`, `prominenceM`, and `longS`. Using our WSD approach, each of these properties is first disambiguated to identify the synset which is most relevant to the query context. Then, the selected synsets of all the properties are put together and treated as different meanings of a polysemous word in order to have the WSD approach identify the most related synset to the query. Using this technique, the algorithm correctly selects the property `elevation` to use and then proceeds to find mountains with elevation higher than that of the instance *Nanga Parbat*. In order to verify whether our WSD algorithm was affected by the abbreviations (such as `latM`), we asked the same question after replacing the abbreviations by their equivalent word (*latitude* for `latM`). We found that it still selected `elevation` as the most relevant property since it had more overlapping terms with the query than the others.

Indeed, it is more challenging to identify this link between the term and the appropriate property for more generic comparatives like *larger* in the query *cities larger than Cairo*. Several interpretations arise, including area of the city, its population or density. The ability to resolve this scenario is future work.

Superlatives For superlatives, we identified two different scenarios. Either it is used with a numeric datatype property such as in the example *city with largest population*, or with a concept as in *what is the highest mountain*. In the first scenario, the normal triples between the concept `city` and property `population` are generated, in addition to an `ORDER BY` clause together with a `LIMIT` to return the first result.

The second scenario is more challenging and similar to the last comparative scenario explained above and is indeed tackled using the same technique. All numeric datatype properties of the concept are identified and the most relevant one (identified by our WSD approach) is used in the query. Again, in this example, the term *highest* is successfully mapped to the property `elevation`.

3.5 Integration of Triples and Generation of SPARQL Queries

The final stage involves generating the SPARQL query by integrating the triples generated from the previous stages. Information about the query term position is used to order the sets of triples originating from different query terms. Furthermore, for triples originating from the same query term, care is taken to ensure they are executed in the appropriate order until an answer is found (when higher precision is preferred and thus not all matches are used).

For example, in the question *Which software has been developed by organizations founded in California?*, the terms in the first part — *software, developed, organizations* — generate the following triples:

```
?software <dbp:developer> ?organisation.
?software a <dbo:Software>.
?organisation a <dbo:Organisation>.
```

And the terms in the second part — *organizations, founded, California* — generate the following triples:

```
?organisation <dbp:foundation> <res:California>.
?organisation a <dbo:Organisation>.
```

To produce the final query, duplicates are removed while merging the triples and the `SELECT` and `WHERE` clauses are added in addition to any aggregate restrictions or solution modifiers required.

4 Evaluation

This section presents a comparative evaluation of our approach using the DBpedia test data provided by the 2nd Open Challenge on Question Answering over Linked Data (QALD-2). Results were produced by QALD-2 evaluation tool[7].

[7] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=evaltool&q=2`

Table 1. Results for our approach (SenseAware, shown in bold) with SOA approaches.

Approach	**Answered**	**Correct**	**P**	**R**	**F_1**
BELA	31	17	0.62	0.73	0.67
QAKiS	35	11	0.39	0.37	0.38
Alexandria	25	5	0.43	0.46	0.45
SenseAware	**54**	**41**	**0.51**	**0.53**	**0.52**
SemSeK	80	32	0.44	0.48	0.46
MHE	97	30	0.36	0.4	0.38

4.1 Results and Discussion

Table 1 shows the performance in terms of precision, recall and f-measure, in addition to coverage (number of answered questions, out of 100) and the number of correct answers (defined as P=R=(F_1)=1.0). Our approach (SenseAware) is compared with QALD-2 participants [28]: SemSeK, Alexandria, MHE and QAKiS, in addition to BELA [10], which was evaluated after QALD-2 but using the same dataset and questions. The results show SenseAware is very promising especially in terms of correctness: 76% of answers were 'correct'. It also achieves higher performance than the other approaches except for BELA. The latter has higher values for P, R (and F_1) since it favours these over coverage and correctness (only 31 answered of which 55% were 'correct').

After excluding out-of-scope questions (as defined by the organizers) and any containing references to concepts and properties in ontologies other than DBpedia since they are not yet indexed by our approach, we had 75 questions left. The 21 questions – out of 75 – that our approach couldn't provide an answer for fall into the following categories:

1. No matches were found for one or more query terms after query expansion.
2. Matches were found for all query terms but question type is out-of-scope.
3. Question requires higher level of reasoning than is currently provided.

Examples of the first category are: *What did Bruce Carver die from?* and *Who owns Aldi?*, in which the terms *die* and *owns* should be mapped to the properties `deathcause` and `keyPerson`, respectively. Questions that are not yet addressed are mainly the ones which require identifying the right property to use depending on the answer type. An example is *When was the Battle of Gettysburg?* which requires using the property `date`. Another example is *In which films did Julia Roberts as well as Richard Gere play?*. Here, our approach could not relate the concept `films` with *Richard Gere*. Although, it is designed to maximise the chance of linking concepts, properties and instances in the query, without being affected with the sentence structure, this version cannot yet link a term (*films*) that is being successfully related to other terms (*Julia Roberts*) to an additional term (Richard Gere). However, it can still solve complex questions that require relating terms that are not consecutively positioned (e.g., *films* and *Julia Roberts*) in the question *In which films directed by Garry Marshall was Julia Roberts starring?*. Finally, examples of questions in the third category are *Is Frank Herbert still alive?* which requires understanding that the expression *still alive* means not finding a value for the death date of the person.

5 Discussion

In designing an approach to answer user questions, it is usually difficult to decide whether to favour precision or recall, since it is well known that an increase in one commonly causes a decrease in the other. In fact, which to favour depends not only on the users but on their specific information need at some point. We experienced this while designing our approach since we had to decide on the following choices to be in favour of precision or recall:

Query Relaxation Consider the question *Give me all actors starring in Last Action Hero*. This question explicitly defines the type of entities requested as *actors* which justifies querying the dataset for only this type. Hence, we would add the triple: `?actor a <dbo:Actor>` to restrict the results generated from: `res:Last_Action_Hero dbp:starring ?actor` to only these who are actors. However, the current quality of Linked Data would be a major problem with this choice, since not all entities are typed [29], let alone typed correctly. This causes the previous query to fail, and only succeeds to return the required answer after query relaxation, i.e., removing the restricting triple `?actor a <dbo:Actor>`. This choice is in favor of *recall*. It affects precision since, for the question *How many films did Leonardo DiCaprio star in?*, following this technique would also return answers that are *TV series* rather than *films* such as `res:Parenthood_(1990_TV_series)`. Our decision was to favor precision and keep the restriction whenever it is explicitly specified in the user's query.

Best or All Matches The decision to use only the best match found in the ontology for a query term or all matches whose similarity exceeds a threshold can affect precision and recall. For instance, the term *founded* in the question *software developed by organizations founded in California* has several matches including `foundation` and `foundationPlace`. Using only the best match (`foundation`) would not generate all the results and, in turn, affects the recall. On the other hand, if these properties were not relevant to the query, this would harm the precision. To balance precision and recall, our algorithm uses all matches while employing a high value for the similarity threshold and performing checks against the ontology structure to assure relevant matches are only used in the final query.

Query Expansion when a query term is not found in the ontology, query expansion is performed to identify related terms and repeat the matching process using these terms. However, in some scenarios, this expansion might be useful to increase the recall, when the query term is not sufficient to return all the answers. Therefore, it would be useful to perform the expansion for all query terms even if they had matches in the ontology. An example of this is when one of the two terms *website* or *homepage* are used in a query and both of them have matches in the ontology. Using only one of them could affect recall for some queries. On the other hand, the quality/relevance of expansion terms (for polysemous words) depends fully on the WSD approach. If a wrong sense was identified for a query term, this list will be noisy and lead to false matches. Additionally, the disambiguation process is computationally expensive and therefore, for these reasons, we perform query expansion only when no matches are found in the ontology for a term or when no results are generated using the identified matches.

6 Conclusions

In this paper, we have presented a NL semantic search approach to answer user queries expressed in their own terms and format. The main steps of the approach are recognising and disambiguating named entities in the query; parsing and matching the rest of the query to ontology concepts and properties; and, finally, generating triples from these matches and integrating them to form the final SPARQL query. Using BabelNet in our WSD approach allowed high performance disambiguation of polysemous query terms and therefore, when required, produced highly relevant terms for query expansion. Together with a set of SOA string similarity algorithms and filtering techniques to produce accurate mappings for query terms, we have shown that our approach's performance is competitive (especially in the number of questions answered correctly: 76%) when evaluated using the QALD-2 dataset. We have also discussed challenges facing our approach (which are in common with many NL approaches). Query terms that are very difficult to be matched to the ontology or queries requiring advanced reasoning would, indeed, require a 'user in the loop' to assist the system in addressing these challenges. This agrees with our recommendation [7] to combine the benefits of visualising the search space offered by view-based query approaches (which would support the user in the above scenarios) with a NL-input feature that would balance difficulty with speed of query formulation.

References

1. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies. In: Proceedings of ESWC 2007
2. López, V., Fernández, M., Motta, E., Stieler, N.: PowerAqua: supporting users in querying and exploring the semantic web. Semantic Web **3** (2012)
3. Bernstein, A., Kaufmann, E., Kaiser, C.: Querying the Semantic Web with Ginseng: A Guided Input Natural Language Search Engine. In: Proc. of WITS 2005
4. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid Search: Effectively Combining Keywords and Ontology-based Searches. In: Proceedings of ESWC 2008
5. Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C.: Querying Ontologies: A Controlled English Interface for End-users. In: Proceedings of ISWC 2005
6. Kaufmann, E., Bernstein, A.: Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. J. Web Sem. **8** (2010)
7. Elbedweihy, K., Wrigley, S.N., Ciravegna, F.: Evaluating Semantic Search Query Approaches with Expert and Casual Users. In: Evaluations and Experiments Track, 11th International Semantic Web Conference (ISWC). (2012)
8. Voorhees, E.M.: Using wordnet to disambiguate word senses for text retrieval. In: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. (1993)
9. Rizzo, G., Troncy, R.: NERD : a Framework for Evaluating Named Entity Recognition Tools in the Web of Data. In: Proceedings of ISWC 2011
10. Walter, S., Unger, C., Cimiano, P., Bär, D.: Evaluation of a layered approach to question answering over linked data. In: Proceedings of ISWC 2012. (2012)

11. López, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: Proceedings of ESWC 2006
12. Navigli, R., Ponzetto, S.P.: Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif. Intell. **193** (2012) 217–250
13. Agirre, E., De Lacalle, O.L., Soroa, A.: Knowledge-based wsd on specific domains: performing better than generic supervised wsd. In: Proceedings of the 21st international jont conference on Artifical intelligence. IJCAI'09 (2009)
14. Navigli, R., Litkowski, K.C., Hargraves, O.: SemEval-2007 Task 07: Coarse-Grained English All-Words Task. In: Proceedings of SemEval-2007. (2007)
15. Ponzetto, S.P., Navigli, R.: Knowledge-rich word sense disambiguation rivaling supervised systems. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. (2010)
16. Ide, N., Vronis, J.: Word sense disambiguation: The state of the art. Computational Linguistics **24** (1998) 1–40
17. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In: In Proceedings of IJCAI 2003. (2003)
18. Buscaldi, D., Rosso, P., Masulli, F.: Integrating conceptual density with wordnet domains and cald glosses for noun sense disambiguation. In: Advances in Natural Language Processing. Volume 3230. Springer Berlin Heidelberg (2004) 183–194
19. Reichert, M., Linckels, S., Meinel, C., Engel, T.: Students perception of a semantic search engine. In: Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA2005). (2005)
20. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: In Advances in Neural Information Processing Systems 15 (NIPS. (2003)
21. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research. (1990)
22. Philips, L.: The Double Metaphone search algorithm. C/C++ Users Journal **18** (2000) 38–43
23. Monge, A., Elkan, C.: The field matching problem: Algorithms and applications. In: In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. (1996)
24. Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural Language Interface to Ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In: Proceedings of ESWC 2010
25. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of IJCAI-03 Workshop on Information Integration. (2003)
26. da Silva, R., Stasiu, R.K., Orengo, V.M., Heuser, C.A.: Measuring quality of similarity functions in approximate data matching. J. Informetrics **1** (2007) 35–46
27. Tang, L.R., Mooney, R.J.: Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In: ECML 2001. (2001) 466–477
28. Unger, C., Cimiano, P., Lopez, V., Motta, E., Buitelaar, P., Cyganiak, R., eds.: Proceedings of Interacting with Linked Data (ILD 2012), at ESWC 2012. (2012)
29. Nuzzolese, A.G., Gangemi, A., Presutti, V., Ciancarini, P.: Type inference through the analysis of wikipedia links. In: LDOW. (2012)