Querying the Semantic Web with Ginseng: A Guided Input Natural Language Search Engine

Abraham Bernstein, Esther Kaufmann, Christian Kaiser
Department of Informatics, University of Zurich, Switzerland
{bernstein, kaufmann}@ifi.unizh.ch

Abstract. The Semantic Web presents the vision of a distributed, dynamically growing knowledge base founded on formal logic. Common users, however, seem to have problems even with the simplest Boolean expression. As queries from web search engines show, the great majority of users simply do not use Boolean expressions. So how can we help users to query a web of logic that they do not seem to understand? We address this problem by presenting Ginseng, a quasi natural language guided query interface to the Semantic Web. Ginseng relies on a simple question grammar which gets dynamically extended by the structure of an ontology to guide users in formulating queries in a language seemingly akin to English. Based on the grammar Ginseng then translates the queries into a Semantic Web query language (RDQL), which allows their execution. Our evaluation with 20 users shows that Ginseng is extremely simple to use without any training (as opposed to any logic-based querying approach) resulting in very good query performance (precision = 92.8%, recall = 98.4%). We, furthermore, found that even with its simple grammar/approach Ginseng could process over 40% of questions from a query corpus without modification.

1 Introduction

The Semantic Web presents the vision of a dynamically growing knowledge base that should allow users to draw on and combine distributed information sources specified in languages based on formal logic. Common users, however, were shown to have problems even with the simplest Boolean expressions; the use of the description logic formalism underlying the Semantic Web is beyond their understanding. Experience in information retrieval, for example, demonstrates that users are better at understanding graphical query interfaces than simple Boolean queries [1]. As queries from web search engines reveal, the great majority of users simply do not use Boolean expressions. Bowen et al. even show that people (CS students) who are trained in composing queries in a logic-based formalism (SQL) are usually inept in composing correct queries in realistically-sized databases rather than the small toy examples used in database classes [2]. *So how can we bridge the gap between the (description) logic-based Semantic Web and real-world users, who are at least ill at ease and, oftentimes, unable to use formal logic concepts?*

We address this problem by presenting *Ginseng, a **g**uided **i**nput **n**atural language **s**earch **eng**ine for the Semantic Web*. Ginseng relies on a simple querying grammar which is dynamically extended based on the ontology structure. The extended grammar is used to parse queries which strongly resemble plain English. When the user enters queries an incremental parser relies on the grammar to constantly check the entries to propose possible continuations of the query similar to completion suggestions in Unix shells and prevent ungrammatical entries. Once a query is fully entered Ginseng uses additional query construction information to translate the quasi English query into the *RDF Data Query Language* (RDQL)[1] and executes it. As such Ginseng provides a guided input natural language search engine for the Semantic Web.

In this paper we introduce the technical setup of Ginseng by providing an overview to its functionality. Next, we describe the empirical evaluation of Ginseng with 20 users. The paper closes with a discussion of the limitations of our approach as well as related work.

2 Ginseng – A Guided Input Natural Language Search Engine

Ginseng provides a quasi natural language (NL) querying access to any OWL[2] knowledge base. The main difference between Ginseng and full natural language interfaces (NLI) [3] is that Ginseng doesn't use any predefined vocabulary and doesn't try to interpret the queries (logically or syntactically). Instead, Ginseng

[1] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
[2] http://www.w3.org/TR/owl-features/

In proceedings of 15th Workshop on Information Technology and Systems (WITS 2005),
Las Vegas, December 2005.

"only knows" the vocabulary defined by the currently loaded ontologies (complemented with added synonym information into the ontologies). The vocabulary is closed and the user has to follow it. This can limit the user's possibilities in general but ensures that every query leads to a correctly interpretable result. Alleviating this limitation, however, the vocabulary grows with every additionally loaded ontology.

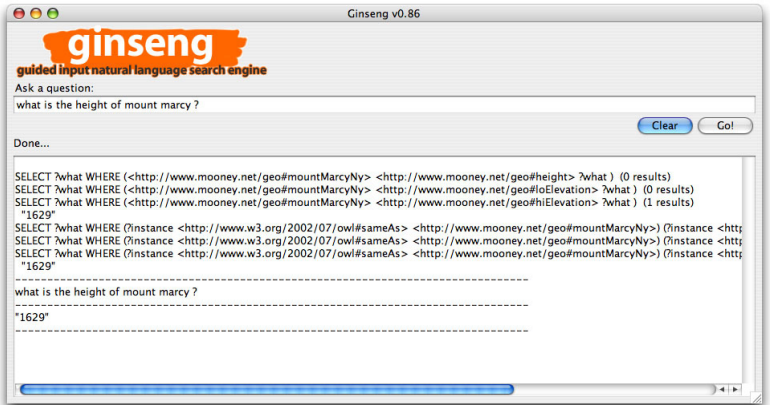

Fig. 1. The Ginseng user interface after executing a query

Ginseng allows users to query any Semantic Web knowledge base using a guided input NL. As shown in Fig. 1 the user enters the query in English into a free form entry field. When the user starts typing the system predicts the possible completions of what the user enters and presents the user with a choice popup box. Whilst the user is in the middle of a word, the popup offers suggestions on how to complete the current word (first three examples in Fig. 2). Obviously, the possible choices get reduced as the user continues to type. Ginseng, thus, guides the user through the set of possible queries while avoiding ungrammatical statements. When a query is completed Ginseng translates it into a Semantic Web query language and displays the result (shown as RDQL statements and result at the bottom of Fig. 1).

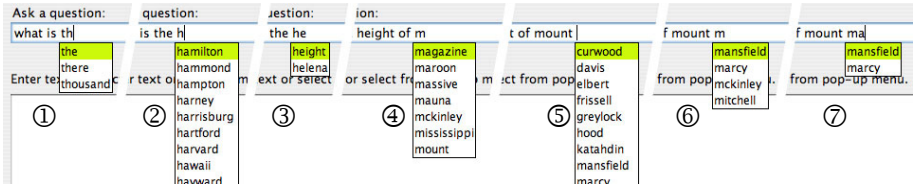

Fig. 2. Ginseng query-completion popup/choice window (numbers added)

The *Ginseng software architecture* has three parts: a partially dynamically generated multi-level grammar, an incremental parser, and an ontology-access layer. The first two parts were developed as part of the Ginseng project, as the ontology-access layer we used Jena.[3] When starting Ginseng, all ontologies in a predefined search path are loaded. For each ontology the necessary rules are generated to extend the static part of the grammar, which contains the ontology-independent rules specifying general sentence structures. The grammar is then used by the incremental parser in two ways: First, its specifies the complete set of parsable sentences, which can be used to provide the user with alternatives during entry as described above and prevent incorrect entries. Second, the grammar contains information on how to construct the RDQL queries. Thus, a complete parse tree can also be used to generate the resulting query.

The *Ginseng grammar* describes both the parse rules of the English queries (entered by the user) and the query composition elements of the RDQL queries. It mostly follows the BNF notation. A detailed description of the grammar would go beyond the scope of this paper. Consider the following example grammar[4]:

[3] http://jena.sourceforge.net/.

[4] The pipe symbol "|" is used to separate the parse elements of each rule from the associated query elements. Non-terminals are shown in double arrows ("<<" and ">>") to distinguish them from URIs. To state a logical "or" in the grammar despite the already used "|" symbol one repeats the rule's head, which causes the parser to interpret the two rules as alternatives. The second

```
(1)   <START>     ::= <OQ> ?
                      | SELECT <<OQ>>
                      | WHERE (<<OQ>>)

(2a)  <OQ>        ::= which <subject> <verb>
                      | <<subject>>
                      | <<subject:1>> <<verb>>

(2b)  <OQ>        ::= what <subject> <verb>
                      | <<subject>>
                      | <<subject:1>> <<subject>>) (<<subject:1>> <<verb>>

(3)   <subject>   ::= state
                      | ?state
                      | <RDF#type> <geo#state> (type=[<geo#state>])

(4)   <verb>      ::= borders <object>
                      | -
                      | <geo#borders> <<object>> (domain=[<geo#state>], range=[<geo#state>])

(5)   <object>    ::= New York City
                      | ?newyorkcity
                      | <geo#newYorkCity> (type=[<geo#city>, <geo#capital>])

(6)   <object>    ::= Mississippi
                      | ?mississippi
                      | <geo#mississippiRiver> (type=[<geo#river>])

(7)   <object>    ::= Mississippi
                      | ?mississippi
                      | <geo#mississippiState> (type=[<geo#state>])
```

The grammar recursively searches for possible matches for the non-terminal symbols on the left side of the rules and replaces them by the right side of a conformable rule. When finding terminal symbols (e.g., "which" and "what" in rules 2a and 2b) Ginseng offers them to the user in a popup as possible entries. The OWL-based type information (providing type, domain, and range) ensures that the resolution of <object> in a sentence such as "What state borders <object> ?" leads to the correct symbol with a domain that is subsumed by the range specification of the verb/predicate (e.g., <geo#state> in rule 4). The verb "borders" excludes rule 5 for the following object, as "New York City" is a <geo#city> and <geo#capital>. Consequently rule 6 is used.

Behind the first "|" symbol of every rule the grammar also contains the information to assemble the RDQL query by following the parse tree in a bottom up fashion. Taking the query assembly instructions of "What state borders Mississippi?" (parsed with rules 2b, 3, 4, 6), thus, results in the RDQL statement "SELECT ?state WHERE (?state <RDF#type> subject) (?state <geo#borders> <geo#mississippiState>)."

Having briefly explained the functionality of the grammar we have to explain how it is created. As mentioned above, Ginseng's grammar has both a static part specifying the generally possible query sentence structures and a dynamic part which get generated from the ontology.

The *static grammar rules* (e.g., rules 1, 2) provide the basic sentence structures and phrases for questions. It handles general question structures as the one used in the example above as well as other types of queries such as closed questions (typically resulting in an answer of "yes" or "no") or questions resulting in numbers (e.g., starting with "how many…"). Furthermore, it provides sentence construction rules for the conjunction or disjunction of two phrases (or sentence parts). The static grammar consists of about 120 mostly empirically constructed domain-independent rules.

The *dynamic grammar rules* (e.g., rules 3–6) get generated from the loaded OWL ontologies. Ginseng essentially parses an ontology and generates a grammar rule for each class, instance, objects property, and data type property. Ginseng also allows annotating the ontology with Ginseng tags (from the ginseng namespace) to allow for synonyms of the expressions used in the ontology model. While such annotations aren't necessary for Ginseng to run correctly, they do extend the vocabulary and increase its usability.

---

part of the query construction instruction rules (after the second "|") can have back-references to the first part (between the two "|"): here "<<xx:1>>" refers to the symbol "<<xx>>" in the first query part (i.e., both are bound to the same symbol).

3 Evaluations – Confronting Ginseng with the Real World

To evaluate Ginseng we confronted users with our prototype written in Java. Our goal was to (1) evaluate the *usability of Ginseng* in a realistic task, (2) the users' *query performance* in terms of speed as well as precision/recall, and (3) *Ginseng's ability to parse* a large number of real-world queries. The evaluation's basis was the NLP knowledge base interface evaluation task proposed by [4]. The knowledge base we used in our evaluation contains geographical information about the US. To make the knowledge base accessible to Ginseng we translated it to OWL.

To achieve our first two goals we recruited 20 subjects from the local CS and computational linguistics department. Each subject was confronted with 30 randomly chosen queries from the set of the geographical queries. For half the questions the subjects were asked to enter the query into Ginseng and for the other half we provided them with an SQL interface (as well as an ERM of the database model). Note that we chose SQL because people were more familiar with it than with RDQL. We randomly assigned every subject a different half to be processed by Ginseng (respectively by SQL). We also randomly alternated the order of the queries and which of the two interfaces was to be used first. The subjects had to translate the queries from plain English to either Ginseng or SQL. We logged each key entry and timed the overall querying process indicating the speed of query entry. At the end of each half we performed the SUS [5] standardized usability test, which establishes a person's impressions regarding the user interface.

The results were quite interesting and not necessarily expected on all accounts. We found that the *subjects were significantly faster with Ginseng than entering the SQL queries* (confirmed by a t-test with $p = 1.8\%$; average times: Ginseng = 2 min 16 s, SQL = 3 min 17 s). This is perhaps unsurprising, as Ginseng relieves the subjects from thinking about the meta-model and having to remember a query language syntax. Using the SUS questions Ginseng was rated a *better integrated* (t-test: $p = 3.5\%$) querying package and being *easier to learn* (t-test: $p = 0.97\%$). Overall, Ginseng did get a higher SUS score but that result is not significant (t-test: $p = 17.4\%$). Note, however, that all subjects were researchers from the CS and computational linguistics department. Thus, we can assume that they all had formal SQL training in their studies and, partially, even use it in their research. Consequently, one could argue that Ginseng entered the comparison with a handicap. We can, thus, assume that a test with subjects drawn from the general population would result in an even clearer (and most probably significant) lead of Ginseng.

To evaluate the power of Ginseng's grammar we ran all the 880 queries from the geographical Mooney knowledge base. We found that Ginseng could execute 40% of the queries right out of the box, which is a good result, given that Ginseng has no general vocabulary. The queries that could be parsed resulted in a precision of 92.8% and a recall of 98.4%. The excellent recall shows nicely that Ginseng can capture the Gestalt of a NL query interface without any complicated logic processing. The slightly worse precision is probably a result of Ginseng's limited ability to actually "understand" the queries: it "only" extracts part of the information in the NL. Yet, the overall retrieval performance (F-measure = 0.955) reconfirms the power of Ginseng's simple design.

4 Limitations and Future Work

We can think of three limitations to the work presented here. First, and most obviously, Ginseng cannot process all NL queries due to its construction. Nevertheless, as the usability evaluation shows, this limitation didn't seem to bother the users as much as could be expected.

Second, the usability evaluation is limited by the choice and size of the subject pool among CS and computational linguistics researchers. We, therefore, intend to carry out a usability test with a more representative selection of the population in general and with more than 20 subjects. This should allow us to thoroughly evaluate Ginseng's usability with regard to a wide range of real-world users as well as to investigate how people's retrieval performance and affinity to different tools is related to their background. Furthermore, the performance evaluation could be improved by using additional data sets. We evaluated the power of Ginseng's grammar running 880 queries from only one of the three Mooney knowledge bases [4]. We plan to include the other two knowledge bases in Ginseng. The complete corpus of 1770 Mooney queries might allow us to learn the set of static grammar rules (instead of creating them manually). Note,

however, that exactly the fact that Ginseng does not need a large data set to learn (or infer using machine learning methods) the grammar from might be one of its key advantages.
Last, even though Ginseng could execute over 40% of the geographical queries without any grammar modification, we believe that we can clearly improve this result by extending Ginseng's property tags. The tags' generation can probably be automated with WordNet [6] and machine learning techniques. This could lead to a semi-automated ontology markup environment to prepare the ontologies for Ginseng querying – an endeavor we intend to undertake in the near future.

5 Related Work

We discuss related work in three categories: *guided query interfaces*, *NLI to databases*, and *NLI to Semantic Web knowledge bases*.

Guided query interfaces. One recent approach to guided query interfaces is the PENG system presented by Schwitter and Tilbrook [7]. PENG is a machine-oriented controlled NL that has a restricted grammar and lexicon. When applied it guides the user through the formulation of PENG sentences. After each word form entered the editor indicates what kind of syntactic structures can follow, therefore guaranteeing compliance to the rules of PENG. The user does not have to learn and remember the restrictions of the controlled language. The underlying framework of the PENG system consists of a complete NL processing engine, whereas Ginseng uses a simple querying grammar which can be dynamically extended by any OWL ontology structure. Note that PENG's goal is also different than Ginseng's. PENG aims at providing a full NL processing environment. Thus, knowledge has to be entered into the system using PENG. Ginseng, in contrast, aims at querying existing semantically annotated content. The Kaleidoscope system is an earlier approach that belongs to the family of so-called *menu-guided NL inferfaces* [8].
In Kaleidoscope SQL query creation is guided by menus. The user adds constituents to a query by selecting lexical items from a set of popup menus. Depending on the current state of the query the system successively updates the menus. Unfortunately, no evaluation is performed showing the usability or the retrieval performance of the system.

NLI to databases. We found that work on NLI to databases (not ontologized knowledge bases) has largely tapered off since the 80's [3], even though the need for them has become increasingly acute. A few approaches in the area of database interfaces have emerged recently [9-11]. Among them the most closely related approach is the PRECISE project [12] that proposes a NLI to relational databases. PRECISE uses a data-base augmented tokenization of a query's parse tree to generate the most likely corresponding SQL statement. While PRECISE allows users to phrase queries in full English, our approach limits the language constructs of the queries ensuring that only those queries are formulated to which answers can be found in the queried ontology. Our evaluation shows that Ginseng achieves similar precision/recall rates as PRECISE. We plan to conduct an exhaustive empirical comparison between the two approaches in our future work.

NLI to Semantic Web knowledge bases. It is hard to find approaches querying Semantic Web content. GAPP [13] is a question-answering system developed for querying the *Foundational Model of Anatomy* (FMA) knowledge base. GAPP takes NL questions as input and translates them into the structured query language *StruQL*. The results of the evaluation show that GAPP provides an intuitive and convenient way for anatomists to browse the FMA knowledge base. The approach differs from ours in that its query construction and, therefore, its overall application are highly restricted to one semantically constrained domain. Furthermore, their model seems to be limited to a set of domain-specific user-defined pattern matching rules. In contrast, our simple approach manages to integrate new ontologies with good query processing performance without any modification of the grammar.
The START question answering system [14] allows NL annotations describing the content of a knowledge base. To answer a user question START also annotates and compares the query against the annotations derived from the knowledge base. The goal of the augmentation by metadata in NL is to render the logic-based Semantic Web foundation friendlier to humans. While START allows full NL questions, Ginseng limits the vocabulary to the universe of discourse as defined by the loaded ontologies. Consequently, every query entered into Ginseng leads to a result.

6 Conclusions

People's familiarity with NL might be the key to simplify their interaction with the Semantic Web. Ginseng provides a guided input NL search engine for Semantic Web knowledge bases. Based on the structure of the ontologies applied it relies on a simple, dynamically extendable grammar that can be used to parse quasi-natural English queries. The evaluation showed that Ginseng is extremely easy to use without any training while it achieves very good retrieval performance and adapts well to new ontologies. Following Dittenbach et al.'s [15] findings that using a subset of English is sufficient to query knowledge bases, we could forgo the need for a full NL processing machinery avoiding all the computational and linguistic complexities involved with such an effort. The result is a simple and computationally cheap but highly adaptive approach to guided English Semantic Web querying – a potentially important component for bridging the gap between real-world users and the logic-based underpinnings of the Semantic Web.

Acknowledgements

The authors would like to thank Ray Mooney and his group for having generously supplied the knowledge bases, English questions, and queries to us, and the anonymous reviewers for their helpful comments. This work was partially supported by the Swiss National Science Foundation (200021-100149/1).

References

1. Spoerri, A.: InfoCrystal: A Visual Tool for Information Retrieval Management. ICKM1993. Washington, D.C. (1993) 11-20
2. Bowen, P.L., Chang, C.-J.A., Rohde, F.H.: Non-Length Based Query Challenges: An Initial Taxonomy. WITS 2004. Washington, D.C. (2004) 74-79
3. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural Language Interfaces to Databases - An Introduction. Natural Language Engineering 1/1 (1995) 29-81
4. Tang, L.R., Mooney, R.J.: Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. ECML-2001. Freiburg, Germany (2001) 466-477
5. Brooke, J.: SUS - A "quick and dirty" Usability Scale. In: Jordan, P.W., et al. (eds.): Usability Evaluation in Industry. Taylor & Francis, London (1996)
6. Miller, G.A., et al.: Introduction to WordNet: An On-line Lexical Database. Technical Report. Cognitive Science Laboratory, Princeton University, Princeton, NJ (1993)
7. Schwitter, R., Tilbrook, M.: Dynamic Semantics at Work. International Workshop on Logic and Engineering of Natural Language Semantics. Kanazawa, Japan (2004) 49-60
8. Cha, S.K.: Kaleidoscope: A Cooperative Menu-guided Query Interface (SQL Version). IEEE Transactions on Knowledge and Data Engineering 3/1 (1991) 42-47
9. Guarino, N., Masolo, C., Vetere, G.: OntoSeek: Content-Based Access to the Web. IEEE Intelligent Systems 14/3 (1999) 70-80
10. Andreason, T.: An Approach to Knowledge-based Query Evaluation. Fuzzy Sets and Systems 140/1 (2003) 75-91
11. Minock, M.: A Phrasal Approach to Natural Language Interfaces over Databases. Umeå Techreport UMINF-05.09. University of Umeå, Sweden(2005)
12. Popescu, A.-M., Etzioni, O., Kautz, H.: Towards a Theory of Natural Language Interfaces to Databases. 8th International Conference on Intelligent User Interfaces. Miami, FL (2003) 149-157
13. Distelhorst, G., et al.: A Prototype Natural Language Interface to a Large Complex Knowledge Base, the Foundational Model of Anatomy. American Medical Informatics Association Annual Fall Symposium. Philadelphia, PA (2003) 200-204
14. Katz, B., et al.: Natural Language Annotations for the Semantic Web. International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE 2002). Irvine, CA (2002)
15. Dittenbach, M., Merkl, D., Berger, H.: A Natural Language Query Interface for Tourism Information. ENTER 2003. Helsinki, Finland (2003) 152-162