TANLION – TAmil Natural Language Interface for Querying ONtologies

Vivek Anandan Ramachandran(✉) and Ilango Krishnamurthi

Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore 641 008 Tamilnadu, India
{rvivekanandan, ilango.krishnamurthi}@gmail.com

Abstract. An ontology contains numerous information in a formal way which cannot be easily understood by casual users. Rendering a natural language interface to ontologies will be more useful for such users, as it allows them to retrieve the necessary information without knowing about the formal specifications existing in the ontologies. Until now, most such interfaces have only been built for accepting user queries in English. Besides English, providing interface to ontologies in other native and regional languages should also be explored, as it enables the casual users to gather information from an ontology without any language barrier. One of the most popular languages in South Asia is Tamil, a classical language. In this paper, we present our research experience in developing TANLION, a Tamil interface for querying ontologies. It accepts a Tamil query from an end-user and tries to recognize the information that the user needs. If that information is available in the ontology, our system retrieves it and presents to the user in Tamil.

Keywords: Natural Language Interface · Ontology · Tamil

1 Introduction

Knowledge management systems that utilize semantic models for representing a consensual knowledge about a domain are widely in use. In such systems, knowledge mostly exists in the form of ontologies[1]. Ontology is a graph consisting of a set of concepts, a set of relationships connecting those concepts and a set of instances. Usually ontologies are developed in formal languages such as Resource Description Framework (RDF) [1], Web Ontology Language (OWL) [2] etc. In order to acquire information from the knowledge available in an ontology, a casual user should know about:

- The syntax of the formal languages used in modeling the ontology.
- The formal expressions and vocabularies used in representing the knowledge.

Providing a Natural Language Interface (NLI) to ontologies will help them to retrieve the necessary information without knowing about the formal specifications

[1] In this paper we refer ontology as a knowledge base that includes concepts, relations and instances existing in a domain.

W. Kim et al. (Eds.): JIST 2013, LNCS 8388, pp. 89–100, 2014.
DOI: 10.1007/978-3-319-06826-8_8,

existing in the ontologies. In addition to English, investigations in providing NLIs should also be carried out in other native and regional languages, as it enables the casual users to acquire information from an ontology without any language hindrance. Further, such NLIs decrease the gap of ontology utilization between the professional and the casual users [3]. As the gap of usage diminishes both ontologies and Semantic web spread widely.

The need for regional language NLIs to ontologies can be explained with a scenario. A farmer who knows only Tamil[2] [4] wants to acquire the answer for the query, *nelvaaRpuuchchiyai aJikka e_n_na uram iTa veeNTum?* (*What fertilizer can be used to destroy threadworms in Rice-plant?*), from a Rice-plant ontology developed in English. In this scenario the farmer faces the following problems:

- He might not know English which is used in developing the ontology.
- He might not know the syntax of the formal language used in modelling the ontology.
- He might not be able to follow the formal expressions used in the ontology.

Considering this as a potential research problem, we have developed a Tamil NLI for querying ontologies (TANLION).

In the above scenario, we have considered using Tamil NLI for querying the Rice-plant ontology. But addressing the factor of customizing NLIs to other ontologies is also a crucial issue. Portable or Transportable NLIs are those that can be customized to new ontologies covering the same or different domain. Portability is an important feature of an NLI because it provides an option for an end user to move NLIs to different domains. TANLION is a portable NLI that can accept a Tamil Natural Language Query (NLQ) and a given ontology as input, and returns the result retrieved from the ontology in Tamil.

This paper is organized as follows: In Sect. 2 we discuss the studies related to TANLION. In Sect. 3 we brief about the design issues considered for developing TANLION. In Sect. 4 we describe about the System Architecture. In Sect. 5 we present the System Evaluation. In Sects. 6 and 7 we give the limitations and the concluding remarks respectively.

2 Related Work

Researches in NLIs have been reported since 1970s [5, 6]. Extensive studies have been conducted to provide NLIs to a database [7–9]. As a result of such research, good NLIs to database have emerged [10]. But the major constraint of database is that they are not easily shareable and reusable. Hence the usage of ontologies became increasingly common, as it can be easily reused and shared.

Thus the increased utilization of ontologies inspired researches for providing NLIs to ontologies [11, 12]. The main goal of such NLIs is to recognize the semantics of the

[2] Tamil is a Dravidian language spoken predominantly by the Tamil people of the Southern India. It is also a classical language. It has official status in India, Sri Lanka and Singapore. Tamil is also spoken by substantial minorities in Malaysia, Mauritius and Vietnam.

input NLQ and use it to generate the target SPARQL[3] [13]. Further, those NLIs should either assure a correct output or indicate that it cannot process the NLQ. Along with the efforts on rendering NLIs to ontologies many systems such as Semantic Crystal [14], Ginseng [15], FREyA [16], NLP-Reduce [17], ORAKEL [18], e-Librarian [19], Querix [20], AquaLog [21], PANTO [22], QuestIO [23], NLION [24] and SWAT [25] have evolved.

Semantic Crystal displays the ontology to an end-user in a graphical user interface. The end-user clicks on the needed portion in the ontology, and the system will display the answer accordingly. Comparatively, Ginseng allows the user to query ontology with the help of structures such as pop-up menus, sentence completion choices and suggestion boxes. FREyA system is termed after Feedback, Refinement and Extended Vocabulary Aggregation. It displays ontology contents to the end user in a tree structure. The end-user clicks on the suitable portions in the tree, and the system will display the answer accordingly. Remaining systems are functionally similar, but the difference is to recognize the semantics of the input NLQ:

- Querix uses Stanford parser output and a set of heuristic rules.
- PANTO utilizes the information in the Noun phrases of the NLQ.
- e-Librarian uses normal string matching techniques.
- NLP-Reduce employ stemming, WordNet and string metric techniques.
- AquaLog uses a shallow parser and hand-crafted grammar.
- NLION utilizes the semantic relation between the words in the NLQ and the ontology.
- ORAKEL uses tree structure.
- QuestIO employs shallow language processing and pattern-matching.

SWAT totally differs from the other systems as it transforms the input NLQ in Attempto Controlled English (ACE) to N3. The above systems render NLIs to ontologies by accepting the input NLQ in English. But providing Tamil NLIs to ontologies is not explored as the supporting technologies are required to a great extent and also the availability is scant. Among all the approaches NLION provides an inherent support to develop an NLI for ontology in an easy and a fast way for languages whereas the supporting technologies are availability is scant. So we decided to extend NLION approach to Tamil. In the next Section, we discuss about the technical issues needed to be addressed for developing a Tamil NLI.

3 TANLION Computing Issues

In this Section, we elaborate about the issues to be dealt for developing a standard Tamil NLI and how we handle those issues in TANLION.

[3] SPARQL is a Query Language used to retrieve and manipulate the information from the ontologies that is stored in the RDF/OWL format.

3.1 Splitting

Splitting is the process of separating two or more lexemes present in a single word. For example, the word ain-taaNTuttiTTam (five-year-plan) contains three lexeme ain-tu (five), aaNTu (year) and tiTTam (plan). When ain-taaNTuttiTTam (five-year-plan) is matched directly with 'ain-tu aaNTu tiTTam' (five year plan) it will be erroneous. So splitting should be done on such NLQ words for effective Information Retrieval (IR).

3.2 Stemming

Stemming is the process of reducing derived words to their root form. For example, reducing the word Mothers to the word Mother is stemming. The reason for using stemming in IR systems is that, most of the words exist in the target database in their root form. For example, to represent a concept 'Mother' in any ontology it will be mostly labeled as Mother and not as Mothers. So each NLQ word should be stemmed for effective IR. In TANLION we have used a stripping stemmer for improving the system retrieval ability [26].

3.3 Synonym Expansion

Synonym Expansion (SE) is a technique where variants of each word in a NLQ are used to improve the retrieval in an IR system. Say, an ontology contains a concept 'Mother' with the label Mother and an IR system is looking in the same ontology for the concept 'Mother' with the keyword mom. In such case, the IR system should use either Thesaurus or WordNet and make use of the fact contained in them that Mother is the synonym of mom [27].

3.4 Translator

Providing a Tamil NLI to English ontologies require a bi-directional translation service between English and Tamil. Say, a user query contains a word ammaa, it should be translated to Mother for mapping it with the concept Mother in the ontology. Currently, complete automized translation software does not exists for English to Tamil translation, as it is in the research stage. In TANLION, we handle the translation issue in a simple way. We annotate all the ontology elements with its Tamil equivalence. For the entity Mother we annotate its Tamil equivalent as ammaa. If a system needs the Tamil equivalence of Mother, it can refer to its Tamil annotation value and acquire the result ammaa. By dealing only with the translation of the ontology elements, it is conclusive that our system answering ability will be restricted to Factoid[4] and List[5] NLQs. This can be easily inferred by tracing TANLION working principle. In the next Section, we explain about TANLION working with its architecture.

[4] Factoid query focuses on questions whose answers are entities.

[5] List query focuses on questions whose answers are list of entities.

4 TANLION Computing Issues

In this Section we provide an overview about our System architecture, depicted in the Fig. 1, with its working principle. The system consists of four main parts viz. user interface, query expansion processor, triple extractor and SPARQL convertor. Each component function is explained in the following subsections as follows.

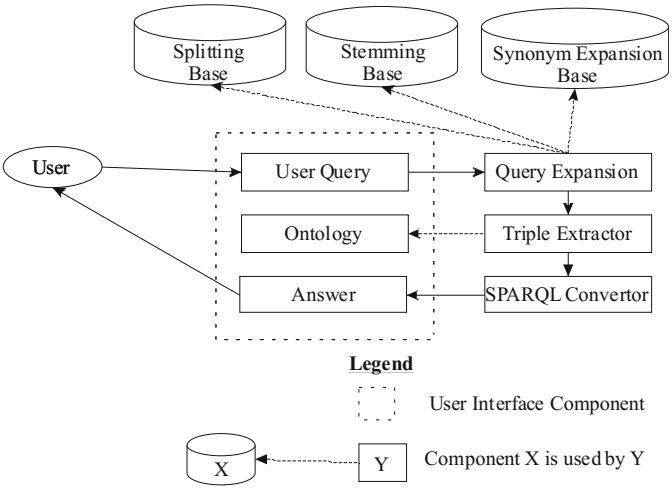

Fig. 1. System Architecture

4.1 User Interface

TANLION UI contains a query field, an answer field and an ontology selection field. TANLION UI is portrayed in the Fig. 2.

4.2 Query Expansion

Query Expansion (QE) is the process of reformulating the input query to improve the performance of the IR system. There are many QE techniques. We use three of them viz. splitting, stemming and synonym expansion. They are briefed as follows:

4.2.1 Splitting

In the example[6], the query token *varuTattiTTatti_n* (*yearly-plan*) is split to *varuTam* (*year*) and *tiTTam* (*plan*) for improving the system retrieval. So it becomes necessary to separate multiple lexemes existing in each input NLQ token for effective IR.

[6] Throughout this Section the term example refers to the query, *mutalaavatu aintu varuTatti_n poJutu yaar piratamaraaka iruntaar*, that is represented in the Fig. 3.

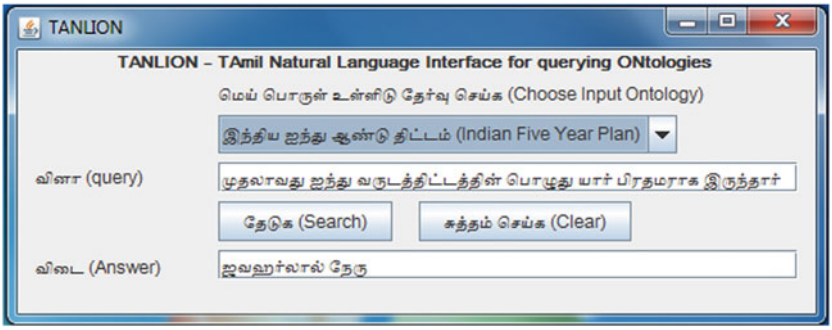

Fig. 2. TANLION User Interface

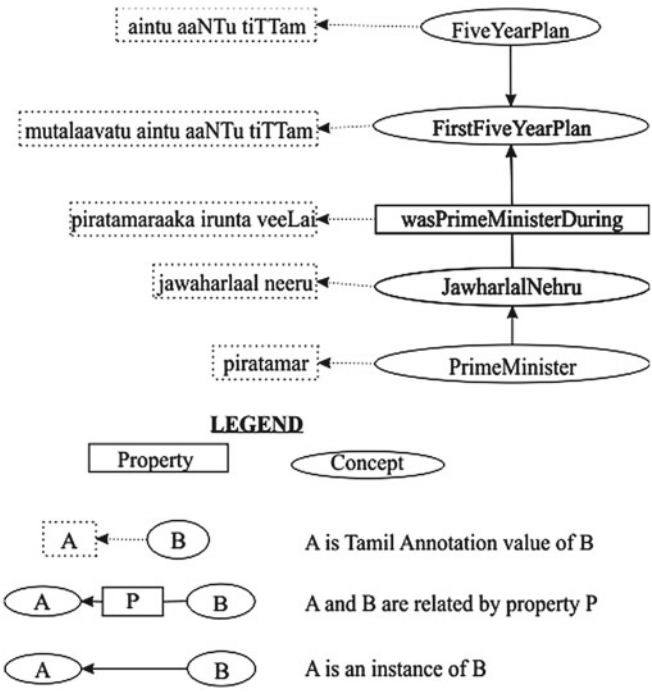

Fig. 3. Example used for explanation

4.2.2 Stemming

In the example, the token in the user query *mutalaavatu* (*The first*) is stemmed to *mutal* (*first*), to make it possible for the system to compare with '*mutal aintu aaNTu tiTTam*' (*first five year plan*). Hence, without stemming it will be difficult for TANLION to interpret the user requirement.

4.2.3 Synonyms Expansion

In the example, the synonyms of the token *varuTam* (*year*) viz. *aaNTu* (*year*) and *varusham* (*year*) should be used by the system to compare it with '*mutal ain-tu aaNTu tiTTam*' (*first five year plan*). Therefore without synonyms expansion it will be hard for TANLION to recognize the user requisite.

4.3 Triple Extractor

Until now basic NLP operations are performed on the user query. But the most important phase is to interpret from the input query what the user needs. In this subsection, we explain the methodology adopted in TANLION to construct the user requirement.

4.3.1 Probable Properties and Resources Extractor

If all the lexicons of a property in an ontology exist in the user query, then there is always a probability that the corresponding property might be the one that the user requires. In the ontology, applying stemming on the lexicons of the annotation value '*piratamaraaka irunta veeLai*' (*was Prime minister during*) yields the result '*pirat-amar iru veeLai*'(*was Prime minister during*). All the lexicons in the resultant are present in the example stemmed user query. With this inference we assume that the property corresponding to '*piratamaraaka irunta veeLai*' is '*wasPrimeMinsiterDuring*' and it is the probable property that the user might need. So TANLION treats '*wasPrimeMinsiterDuring*' as the Probable Property Element (PPE). In this scenario there is only one PPE but in other cases there can also be more than one PPE. Similarly, we deduce that the Probable Resource Elements (PRE) which the user needs are *PrimeMinister*, *FiveYearPlan* and *FirstFiveYearPlan*.

4.3.2 Ambiguity Fixing

In NLP, there is always a possibility of encountering ambiguities while processing a NLQ. In our example user query too there is an ambiguity whether the user requires *FiveYearPlan* or *FirstFiveYearPlan*. After obtaining PPEs and PREs to resolve the ambiguities, our system follows the following hand-crafted rules:

Rule-1: If a PRE, say PREx, subsumes another PRE, say PREy, then remove the PREyfrom the PREs.

Rule-2: If a PRE, say PREx, is an instance of another PRE, say PREy, then remove the PREy from the PREs.

Rule-3: If a PPE subsumes a PRE, say PREx, then remove the PREx from the PREs.

Rule-4: If a PPE, say PPEx, is a sub-property of a PPE, say PPEy, then remove the PPEy from the PPEs.

In the example, as *FiveYearPlan* is subsumed in *FirstFiveYearPlan* our system removes *FiveYearPlan* from the PREs by applying *Rule-1*. Now, the system will be able to solve the ambiguity that the user intended *FirstFiveYearPlan* and not

FiveYearPlan. Similarly, as *PrimeMinister* is subsumed in *wasPrimeMinisterDuring* our system removes *PrimeMinister* from PREs by applying *Rule-3*. Finally, PREs list will contain only *FirstFiveYearPlan*.

4.3.3 Probable Triple extractor

Consider the fact, '*Jawaharlal Nehru was the Prime Minister during First five year plan*'. It is represented in a Triple as, '*:JawaharlalNehru :wasPrimeMinisterDuring :FirstFiveYearPlan.*'. After resolving the ambiguity, our approach tries to find whether any Triple exists in the ontology of the following forms:

?ar :PPEy :PREx . (or) *:PREx :PPEy ?ar.* (or) *:PREx ?ap :PREx.*

where '?ar' and '?ap' means any resource and any property in the ontology respectively. Notice that '?' is used in the Triple form to satisfy the standard notation requirement. If any such Triple exists, it is treated as a Probable Triple (PT).In our example, after resolving the ambiguities the PRE is *FirstFiveYearPlan* and the PPE is *wasPrimeMinisterDuring*. A Triple '*?ar :wasPrimeMinisterDuring :FirstFiveYearPlan.*', exists in the ontology. So it is treated as PT. In this example there is only one PT, in other cases there can be more than one PT too. In the next sub-section we explain how the PTs are converted to SPARQL for retrieving the user requested information from an ontology.

4.4 SPARQL Convertor

In general, any query language is used for retrieving and manipulating the information from a database. Similarly, a RDF Query Language (RQL) is used to retrieve and manipulate the information from the ontologies that is stored in the RDF/OWL format. SPARQL is a RQL and it is standardized by the RDF Data Access Working Group of the World Wide Web Consortium. After extracting all the PTs we convert it to SPARQL using the template:

*SELECT distinct * WHERE {PT1 . PT2 . ……. PTn .}.*

It is basically extracting all PTs and placing them in a correct position of a formal SPARQL query to satisfy the syntax constraint. For the example user query, SPARQL generated by TANLION is:

*SELECT distinct * WHERE {?ar :wasPrimeMinisterDuring :FirstFiveYearPlan.}*

The above SPARQL is executed using the Jena SPARQL engine and the result is acquired [28]. Finally, the acquired result's Tamil annotation value is presented to the user. In the example, the result of SPARQL is *JawaharlalNehru*. Its Tamil annotation value *javaharlaal neeru* is displayed to the user.

5 Evaluation

In general, effectiveness means the ability to bring out the result that the user intended. In this Section, we present the parameters used in calculating the effectiveness of our system. To evaluate the performance of TANLION, we implemented a prototype in

Java with the help of the Jena framework. Also, we developed an Indian Five year plan ontology based and used it for evaluation. We analyzed the system effectiveness using two parameters; System ability and Portability.

5.1 System Effectiveness

After developing any NLI system, it is important to analyze the system effectiveness, i.e., to assess the range of the standard questions that the system is able to answer. Unfortunately, there are no standard Tamil query sets. So we requested 7 students, none of whom are not directly or indirectly involved in the TANLION project, to generate questions. They generated totally 108 queries[7]. A deeper analysis on executing these questions over the Indian five year plan ontology (which contains 102 concepts and 46 properties) revealed the following:

- 74.1 % (80 of 108) of them were correctly answered by TANLION.
- 7.4 % (8 of 108) of them were incorrectly answered.
- 10.2 % (11 of 108) of them were not answered due to the failure in Probable Triple generation.
- 8.3 % (9 of 108) of them were not answered due to the failure in query expansion.

After adding the user requested information that is not found in the ontology, we calculated the overall System Effectiveness (SE) using the formula (1).

$$System\ Effectiveness(SE) = \frac{Number\ of\ queries\ correctly\ answered}{Total\ Number\ of\ queries} \quad (1)$$

SE was found to be 74.10 %, which is considerably a good value. Yet, the reason behind achieving a moderate effectiveness value is that our system is in an earlier stage. Still a lot of functionalities, such as increasing the rules for resolving ambiguities in the user query are to be incorporated in our system.

5.2 Portability

We have addressed earlier that TANLION is portable. It could be inferred from our system working principle that construction of answer for the user query depends on the fact that whether the words in the user query exists in the ontologies as resources and properties. So extraction of answer is not dependent on the ontology rather it is dependent on the content of the ontology. The answer extraction phase is dependent on whether the words in the user query exist as resource and property in the ontology. In order to evaluate the portability factor, we interfaced TANLION with a Rice-plant

[7] We have given the information to the students that our system will work for only factoid and list based generic queries.

ontology[8] (which contains 72 concepts and 29 properties) and found the SE to be 70.8 %. This calculation was based on testing the system with 79 queries generated by the same students as mentioned in the previous sub-section. So far we have described about the analysis carried out by us to evaluate our approach. We now proceed to brief about our possible future enhancements in the next Section.

6 Future Work

The limitation with the current version of TANLION includes the following:

6.1 Restriction in Query Handling

Consider the user query, *aintaavatu aintaaNTu tiTTatti_n poJutu etta_nai pirata-markaL naaTTai aaTchi cheytaarkaL* (*How many Prime Minister governed the country during the Fifth five year plan*). The TANLION output is: '*intiraa kaanti, moraaji teechaay*'(*Indira Gandhi, Moraji Desai*), while the user required answer is '*2*'. This issue will be handled in the near future by classifying the questions and providing the result accordingly.

6.2 Scalability

The ontologies used for evaluation are relatively much smaller. Investigation on system performance with the larger ontologies is a part of our future work.

7 Conclusion

In this paper, we hypothesize an approach for providing a Tamil NLI to ontologies that allows an end-user to access the ontologies using Tamil NLQ. We process the user queries as a group of words and do not use complicated semantic or NLP techniques as a normal NLI systems does. This drawback is also TANLION's major strength as it is robust to ungrammatical user queries. Our approach is highly dependent on the quality of vocabulary used in the ontology. Yet this fact is also TANLION's big strength, as it does not need any changes for adapting the system to new ontologies. Evaluation results have shown that our approach is simple, portable and efficient. Further evaluation of correctness of TANLION in terms of precision and recall is a part of our future work. To conclude, in this paper we have extended NLION approach to Tamil with splitting as an additional supporting technology. Explorations on extending NLION approach to other languages will be done in the near future.

[8] Rice-plant ontology is developed by us based on the information given at the Tamilnadu Agricultural University Website.

Appendix: Tamil Transliteration Scheme Used in this Paper

a	aa	i	ii	u	uu	e	ee	ai	o	oo	au	q					
அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ	ஃ					
k	-N	ch	-n	_n	T	N	t	n	p	m	y	r	l	v	J	L	R
க்	ங்	ச்	ஞ்	ன்	ட்	ண்	த்	ந்	ப்	ம்	ய்	ர்	ல்	வ்	ழ்	ள்	ற்

References

1. http://www.w3.org/RDF/
2. http://www.w3.org/TR/2003/CR-owl-guide-0030818/
3. Spink, A., Dietmar, W., Major, J., Tefko, S.: Searching the web: the public and their queries. J. Am. Soc. Inform. Sci. Technol. **52**(3), 226–234 (2001)
4. http://en.wikipedia.org/wiki/Tamil_language
5. Burger, J., Cardie, C., Chaudhri, V., et al.: Tasks and program structures to roadmap research in question & answering (Q&A). NIST Technical Report (2001)
6. Hirschman, L., Gaizauskas, R.: Natural language question answering: the view from here. Nat. Lang. Eng. Special Issue on Question Answering **7**(4), 275–300 (2001)
7. Copestake, A., Jones, K.S.: Natural language interfaces to databases. Knowl. Eng. Rev. **5**(4), 225–249 (1990)
8. Popescu, A.-M., Etzioni, O., Kautz, H.: Towards a theory of natural language interfaces to databases. In: Proceedings of the 8th International conference on Intelligent user interfaces, 12–15 January 2003, Miami, Florida, USA (2003)
9. Jerrold Kaplan, S.: Designing a portable natural language database query system. ACM Trans. Database Syst. (TODS) **9**(1), 1–19 (1984)
10. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: MASQUE/SQL: an efficient and portable natural language query interface for relational databases. In: Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1–4 June 1993, Edinburgh, Scotland, pp. 327–330 (1993)
11. Thompson, C.W., Pazandak, P., Tennant, H.R.: Talk to your semantic web. IEEE Internet Comput. **9**(6), 75–78 (2005)
12. McGuinness, D.L.: Question answering on the semantic web. IEEE Intell. Syst. **19**(1), 82–85 (2004)
13. http://www.w3.org/TR/rdf-sparql-query/
14. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 281–294. Springer, Heidelberg (2007)
15. Bernstein, A., Kaufmann, E., Kaiser, C., Kiefer, C.: Ginseng: a guided input natural language search engine for querying ontologies. In: 2006 Jena User Conference, Bristol, UK, May 2006
16. Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural language interfaces to ontologies: combining syntactic analysis and ontology-based lookup through the user

interaction. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 106–120. Springer, Heidelberg (2010)
17. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: a naïve but domain independent natural language interface for querying ontologies. In: 4th ESWC, Innsbruck, Austria (2007)
18. Cimiano, P., Haase, P., Heizmann, J. Porting natural language interfaces between domains – an experimental user study with the orakel system. In Proceedings of the International Conference on Intelligent User Interfaces (2007)
19. Stojanovic, N.: On Analysing Query Ambiguity for Query Refinement: The Librarian Agent Approach. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 490–505. Springer, Heidelberg (2003)
20. Kaufmann, E., Bernstein, A., Zumstein, R., Querix: a natural language interface to query ontologies based on clarification dialogs. In: 5th International Semantic Web Conference (ISWC 2006), Springer (2006)
21. Lopez, V., Uren, V., Pasin, M., Motta, E.: AquaLog: an ontology-driven question answering system for organizational semantic intranets. J. Web Semant. **5**(2), 72–105 (2007)
22. Wang, C., Xiong, M., Zhou, Q., Yu, Y.: PANTO: a portable natural language interface to ontologies. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 473–487. Springer, Heidelberg (2007)
23. Tablan, V., Damljanovic, D., Bontcheva, K.: A natural language query interface to structured information. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 361–375. Springer, Heidelberg (2008)
24. Ramachandran, V.K., Krishnamurthi, I.: NLION: Natural Language Interface for querying ONtologies. In: Proceedings of 2nd ACM International Conference on Applied research in contemporary computing, Compute 2009, Bangalore, India, 9–10 January, (2009)
25. Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C.: Querying ontologies: a controlled english interface for end-users. In: Gil, Y., Motta, E., Benjamins, V., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 112–126. Springer, Heidelberg (2005)
26. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
27. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
28. http://www.jena.sourceforge.net/ontology/index.html