A Novel Approach Towards Building a Portable NLIDB System Using the Computational Paninian Grammar Framework

Conference Paper · November 2012

DOI: 10.1109/IALP.2012.16

CITATIONS
10

READS
173

6 authors, including:

Abhijeet Gupta
Universität Stuttgart
8 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Deepak Malladi
International Institute of Information Technology, Hyderabad
2 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Puneeth Kukkadapu
International Institute of Information Technology, Hyderabad
3 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Rajeev Sangal
International Institute of Information Technology, Hyderabad
91 PUBLICATIONS **1,034** CITATIONS
SEE PROFILE

A Novel Approach Towards Building a Portable NLIDB System Using the Computational Paninian Grammar Framework

Abhijeet Gupta, Arjun Akula, Deepak Malladi, Puneeth Kukkadapu, Vinay Ainavolu, Rajeev Sangal
Language Technologies Research Center, IIIT-Hyderabad, India
{*abhijeet.gupta, arjunreddy.akula, deepak.malladi, puneeth.kukkadapu, vinay.a*}*@research.iiit.ac.in,*
sangal@iiit.ac.in

***Abstract*—This paper presents a novel approach to building natural language interface to databases (NLIDB) based on Computational Paninian Grammar (CPG). It uses two distinct stages of processing, namely, syntactic processing followed by semantic processing. Syntactic processing makes the processing more general and robust. CPG is a dependency framework in which the analysis is in terms of syntactico-semantic relations. The closeness of these relations makes semantic processing easier and more accurate. It also makes the systems more portable.**

***Keywords*-NLIDB, Computational Paninian Grammar, Karaka Relations, Language Independence, Domain Portability.**

I. Introduction

A NLIDB system takes as input a natural language (NL) query from a user, converts it to a SQL query based on the domain semantics and database schema, retrieves appropriate data from the database and returns the output to the user. Building NLIDB systems has always been a challenge to the research community because of the inherent ambiguity that natural languages possess. Although a deeply researched area for over five decades, the NLIDB systems have not gained the much deserved and required acceptance in real-time and commercial applications. The reason for limited application of this technology is that the present systems rely heavily on linguistic and domain experts for its creation or customization. This makes the portability of the NLIDB systems, across different languages and domains, either very expensive or sometimes impossible. We believe that the key to an effective NLIDB system is to formulate an approach which accurately captures the linguistic semantics. This contributes greatly towards identifying the correct domain semantics and generating the SQL query. A major aim is to design a system which is robust in its functionality and portable across languages and domains.

In this paper, we present an approach based on the CPG framework, in which one can accurately identify the relevant domain elements from the sentence structure and their semantics with respect to each other from a given NL query. CPG produces a syntactic parse which at the same time includes sufficient semantics to map the identified terms to the relational model, thereby considerably reducing the effort put on semantic processing for SQL query generation. Hence we can build a system which can be easily customized and made portable to a new language or domain without the need of expert help.

II. Related Work

Different NLIDB systems use different combinations of processing which may be characterized as: a) surface level (keyword or pattern level), b) syntax level and c) semantic level. Some systems combine surface level processing with semantic processing, while others include syntactic processing with semantic processing but as distinctly separated stages. Hence, most systems can be classified into two major categories:

A. *Keyword and Pattern Matching Models*

This model is based on identifying keywords or patterns of relevant domain terms in the NL query by matching them against semantic patterns [1]. Domain terms thus obtained can be used to form an SQL query. Though apt for simple queries, this model has difficulty in handling complex NL queries. This model requires a large number of predefined patterns and tends to become highly language dependent. In the recent systems, like PRECISE [2], the relevant domain terms are identified by matching them to domain data dictionaries. However, the system gives high accuracy only on semantically tractable queries. Verb semantics are not captured by the system which leads to a loss of information.

B. *Syntactic Models*

Syntactic Models depend on linguistic information based on tokenizers, morph analyzers, POS taggers and word segmentors (in the case of agglutinative languages) and sentential parsers. This information is represented using a phrase structure tree (PST) or dependency tree after which the domain terms are identified with the help of semantic frames. SQL query is generated thereafter.

*1) **Phrase Structure Tree based systems**:* These systems first produce the phrase structure tree, which allows for robust parsers to be used. The PST is next analyzed to produce domain terms and SQL query using templates [3], [4]. The templates used are dependent on linguistic grammar which makes the systems less portable for a new language. The syntactic and semantic modules are tightly coupled making the system's portability a challenge and every transition or update to the system requires both linguistic and domain experts.

*2) **Dependency Tree based systems**:* These systems first produce the dependency tree over the lexical terms in the NL query. They incorporate a layer of conceptual mapping, called the domain conceptual model, which

978-0-7695-4886-9/12 $26.00 © 2012 IEEE
DOI 10.1109/IALP.2012.16

connects the query with the database schema. The NChiql [5] system uses semantic templates of type 'governing-dependent', operating on a conceptual model obtained from generalizing the database model.
We use dependency trees based on CPG in which the relations are syntactico-semantic. This makes the trees to be a lot more semantic than other kinds of dependency trees making the mapping easier.

III. Our Approach

In our approach, we use dependency syntactic analysis followed by semantic analysis using semantic frames to map the phrases or chunks to a domain conceptual model and thereby connecting with elements in the database schema. There are four major stages: normalization of NL query, syntactic parsing, semantic mapping and query generation.

The syntactic parser uses the Computational Paninian Grammar (CPG) framework [6]. It is a dependency framework in which the dependency relations are syntactico-semantic in nature. It has advantages over usual dependency trees because the dependency relations are closer to semantics. At the same time because these are syntactic relations, it is possible to build robust wide coverage syntactic parsers easily. The parse trees so produced are mapped to semantic relations using domain specific semantic frames. The latter process is domain specific and saves us from the need to building any general purpose semantic parser, which is error prone.

A. *The CPG framework*

In the general dependency model, there are two prominent lexical categories: verbs and nouns, which form verbal and nominal nodes. These nodes are connected by directed edges representing head-modifier relations. Similarly, there are relations between words of other lexical categories.

In the CPG framework, the relations between a verbal node and its argument (noun) node(s) are called *karaka* relations. There are only six different karaka relations between verbs and their nominals that participate in the action specified by the verb. They provide an elegant yet compact mapping from the verb to its corresponding syntactico-semantic relations. If we were to use Thematic roles, detailed semantics and rules would be required to identify theta relations from the dependency relations in a sentence.

The karaka relations are different from *subject-object* type of relations as well as from *agent-patient* type of (theta) relations. For example, if we have a database in the domain of manufacturing where the database keeps track of batches of parts produced by a machine under the supervision of an operator, we might come across the following queries:

(1) *Who made the axle rods yesterday?*

(2) *Which machine made the axle rods yesterday?*

In an approach based on *subject-object* type of analysis, the dependency roles filled by the question element are as shown in *Table I*. These can be identified with relatively high accuracy by a general purpose (domain independent) robust parser. Mapping these to the appropriate entity in the database requires greater effort with this variation.

Table I
DEPENDENCY ANALYSIS

Sentence	Question Element	Dependency Relation
(1)	who	subject
(2)	which machine	subject

If a *theta-role* based approach is used, the correct output is as shown in *Table II*. Identifying such theta roles with high accuracy in a general purpose semantic parser is not possible today. As a result, such semantic parsers for a natural language become highly domain dependent and have to be built separately for each domain. This increases the effort needed to build NLIDB systems for each new domain and affects portability of the system.

Table II
THETA ROLE BASED DEPENDENCY ANALYSIS

Sentence	Question Element	Theta Role
(1)	who	agent
(2)	which machine	instrument

The CPG approach treads a middle ground which is linguistically sound [7] and practically convenient. In the CPG approach, the relations are as shown in *Table III*. Note that in sentence (2), 'machine' is the *karta* or *k1* because it is the most independent of the arguments of verbs (or participants in action). Thus, the CPG theory treats 'person' as k1 in (1) and 'machine' as k1 in (2). This ambiguity is retained in k1 and is to be disambiguated later using domain specific information. As a result, disambiguation of certain hard things are systematically postponed to a later stage. For details of the approach see [7] and for high accuracy of parsers see [8].
Mapping to the semantic relations in the domain is an easier task starting from karaka analysis when compared to *subject-object* type of analysis. Most importantly, a broad coverage CPG parser can be used for parsing and the task of mapping to database or semantic elements is done using semantic frames, explained in detail later (in semantic mapping module).

Table III
CPG BASED DEPENDENCY ANALYSIS

Sentence	Question Element	Dependency Relation
(1)	who	k1(karta)
(2)	which machine	k1(karta)

Handling Active-Passive: As another example of closeness of CPG to semantics one can see the handling of *active-passive* sentences. Consider the following *active-passive* examples:

(3) Which CS students were taught the NLP course?

(4) The NLP course was taught to which CS students?

The first sentence is in active voice and the question element (namely, *which students)* is the *subject*. The second sentence is in passive voice and the question element is the *by_object* (students). To map both these to the correct domain element (namely, *student* entity in DB), we would require additional processing. In contrast, in the karaka approach, the question element would be marked by karta and the mapping is straight forward.

B. *Syntactic Parsing Module*

For the syntactic parsing of english we first use the Stanford Typed Dependency parser [9]. We then convert from Stanford typed dependency to CPG karaka relations. One could also use a CPG dependency parser directly which is available for several languages (other than English). For example, consider the query:

(5) Which students took NLP?

The syntactic parse and the syntactico-semantic parse for this NL query are shown in *Fig. 1* and *Fig. 2* respectively.

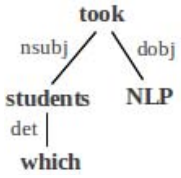

Figure 1. Stanford dependency parse

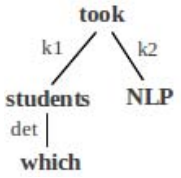

Figure 2. CPG dependency parse

C. *Semantic Mapping Module*

In the domain specific semantic mapping, for each phrase, we identify the domain elements and the underlying relations connecting these elements with the help of semantic frames.

Semantic Frames are used to interpret the NL query using the dependency tree. The frames help in identifying and restricting the scope of NL query to the domain. They capture the semantics of language on one hand and the related domain concepts on the other. There are two types of frames:

(a) *Verb frames* capture the semantics associated with verbs. The ways in which various domain entities participate in relation to the verbs are represented and a mapping to dependency structure (karaka relations) is shown.

(b) *Noun frames* help to identify attribute-value pairs for various entities and relate them to dependency relations between noun and elements.

In the example query (5), the verb 'took' (take) can imply either *taught* or *registered for*. This ambiguity is resolved with the help of semantic frames using the CPG dependency parse. For 'take' there are two possible frames as shown in *Table IV*:

1) teach - which has an expectation of *a faculty teaching a course*.

2) register - which has an expectation of *a student registering for a course*.

Table IV
SEMANTIC FRAMES FOR 'TEACH'(TAKE-1) AND 'REGISTER'(TAKE-2)

Role	Concept	Query Term	Value
root	teach	-	-
k1	faculty	-	-
k2	course	-	-

Role	Concept	Query Term	Value
root	register	-	-
k1	student	-	-
k2	course	-	-

We populate the 'teach' and 'register' frames with the karaka values of 'take' in the CPG dependency parse. Thus, providing the frame instances shown in *Table V*. The semantic frame instances thus contain: a) domain related modifier-modified relationships between the entities in terms of karaka relations, b) the domain concepts corresponding to the populated data and c) their mapping to the relational model.

Table V
SEMANTIC FRAME INSTANCES FOR 'TEACH'(TAKE-1) AND 'REGISTER'(TAKE-2) IN QUERY (5)

Role	Concept	Query Term	Value
root	teach	took	-
k1	faculty	students	which
k2	course	-	NLP

Role	Concept	Query Term	Value
root	register	took	-
k1	student	students	which
k2	course	-	NLP

We observe that in the case of *teach* frame there is a mismatch between concept(expectation) and the query term but not so in the *register* frame. Thus, the verb meaning ambiguity is successfully dealt with using frames in the domain. The same accuracy cannot be achieved using theta roles as theta role based general semantic parsers are very difficult to build. Frame instantiation and disambiguation using subject-object type of syntactic parse are error prone because of the distance between the parse and the semantic frames. Semantic frame instances are appended to their corresponding elements on the CPG dependency parse resulting in a structure we call the Domain Semantic Tree (DST) as shown in *Fig. 3*.

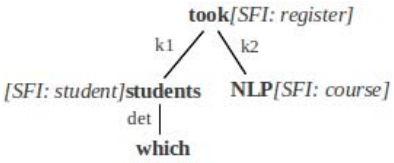

Figure 3. Domain semantic tree (DST) with attached semantic frame instances (SFI) of identified domain elements

Ambiguities in frame identification result in the formation of multiple DSTs which we resolve later by using a weight-based deterministic algorithm (out of scope of this paper). Next we describe the two remaining modules.

D. *Normalization Module*

Normalization module processes the query before it is passed to the syntactic parsing module, adding to the robustness of the processing. The normalization procedure includes: a) identifying the named entities using a NER tool, b) correcting possible errors in named entities using *soundex* and *edit-distance* algorithms, c) finding synonyms to map lexical terms to domain concepts, d) generalizing the lexical terms to synchronize with domain terms and e) identifying and correcting the incomplete words or abbreviations based on the domain.

E. *SQL Query Generator Module*

The SQL generator module follows the semantic mapping module. It takes the DST as its input and maps the query elements to the domain conceptual model based on an entity relationship graph. A path in the entity relationship graph is found using the Minimum Spanning Tree (MST) algorithm, which is used to generate the SQL query.

Example: From the DST for query (5): *Register*, *Student* and *Course* are mapped to the domain conceptual model. A shortest path connecting these entities is found and any intermediate elements lying in between the mapped query elements are also included in the path, giving a connected non-cyclic graph, as shown in *Fig. 4*. Thus, generating the SQL query:

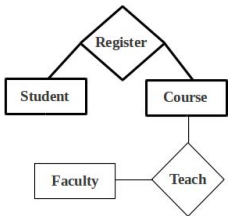

Figure 4. Mapped elements and the path identified using DST on the relationship graph.

```
SELECT student.name
FROM   student, register, course
WHERE  course.name='NLP' AND
       student.ID=register.studentID AND
       register.courseID=course.ID
```

IV. Future Work & Conclusion

Future work will consist of improving the disambiguation methods by using probabilistic weight based algorithms. We also intend to add certain functionalities to the semantic module to deal with NL queries having quantifiers (such as good, etc) and aggregation operators (such as average, etc). Separation between syntax and semantics helps in dealing with these phenomena more easily. The SQL generator module would also be improved further to deal with self and complex join queries.

This paper presents an approach to a NLIDB system which strives to be language independent, is robust in its functionality and is portable. The main contribution of our paper is the application of CPG to map the linguistic semantics to domain semantics. CPG having a syntactico-semantic structure gives us the advantage of interpreting the query structure and extracting information. The earlier approaches, even though syntactically and semantically elaborate, require linguistic and domain experts to set up a system which makes the portability of the system a challenge. Our system can be adapted to any domain by people having basic knowledge of the domain. Hence with this we come a step closer in making NLIDB systems easily portable to a large number of applications.

References

[1] T. Johnson, "Natural language computing: the commercial applications," *The Knowledge Engineering Review*, vol. 1, no. 03, pp. 11–23, 1984.

[2] A. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 149–157.

[3] Y. Li, H. Yang, and H. Jagadish, "Nalix: an interactive natural language interface for querying xml," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 900–902.

[4] W. Woods, R. Kaplan, and B. Nash-Webber, "The lunar sciences natural language information system," 1972.

[5] X. Meng and S. Wang, "Nchiql: The chinese natural language interface to databases," in *Database and Expert Systems Applications*. Springer, 2001, pp. 145–154.

[6] A. Bharati, M. Bhatia, V. Chaitanya, and R. Sangal, "Paninian grammar framework applied to english," Technical Report TRCS-96-238, CSE, IIT Kanpur, Tech. Rep., 1996.

[7] A. Bharati, V. Chaitanya, R. Sangal, and K. Ramakrishnamacharyulu, *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India, 1995.

[8] A. Bharati, S. Husain, D. Sharma, and R. Sangal, "A two-stage constraint based dependency parser for free word order languages," in *Proceedings of the COLIPS International Conference on Asian Language Processing 2008 (IALP)*, 2008.

[9] M. De Marneffe, B. MacCartney, and C. Manning, "Generating typed dependency parses from phrase structure parses," in *Proceedings of LREC*, vol. 6, 2006, pp. 449–454.