

# A Natural Language Processor for Querying Cindi

Niculae Stratica, Leila Kosseim, and Bipin C. Desai

Department of Computer Science  
Concordia University,  
1455 de Maisonneuve Blvd. West  
Montreal, H3G 1M8, Canada

{stratica, kosseim, bcdesai}@cs.concordia.ca

**Abstract--**In this paper, we present our work in querying a Virtual Library database using natural language (NL) questions rather than queries formulated in a formal language. Recent advances in question-answering (QA) have made it possible to answer a question formulated in natural language and locate the answer in a document collection. However, the performance of these systems is not currently high enough to be useful in everyday systems. By using QA techniques to analyse questions, but leaving out the document analysis, we believe that the performance of such a system would be high enough to be used on a daily-basis. The system we are currently developing initially analyses incoming questions formulated in NL. Questions are tagged, lemmatised, parsed and are finally represented by a 3-tuple: a question-word ("who"), a focus verb ("wrote") and a discriminant ("the Old Man and the Sea"). Then, using general lexical and semantic relations such as those found in WordNet, we map the question representation to a meta-representation of the database to translate the NL question into a set of SQL queries. We believe that using state-of-the-art NL techniques in QA and rich lexical resources will allow us to build a system that is domain independent and outperforms previous approaches to NL interfaces to database systems.

**Index Terms--**Natural language interfaces, Database query processing, Database searching, Relational databases, Information retrieval.

## I. INTRODUCTION

### A. The Cindi System

Today an increasing number of electronic documents are accessible via the World Wide Web and, in a short span of ten years, it has become the de-facto standard method of disseminating current contents. Search for relevant documents has also come a long way in these ten years. Today's search engines have improved considerably over those in the early days of the web.

In spite of these improvements, the accuracy of these search engines is still far from acceptable for serious scientific search. The main problem with many of these search engines (many of them generate index entries by using software to scan the source documents) is that their selectivity of

documents is often far from satisfactory [6]. The chances of getting inappropriate documents and missing relevant information because of poor choice of search terms are great. In addition, the user is required to access the actual resource, based on just words appearing in the document without the benefit of the context. The actual document has to be retrieved and perused to judge its relevancy.

These problems are addressed in the Cindi system by using an appropriate index entry called the Semantic Header [3] and providing a mechanism to register, manage and search the bibliography. The system is an active system requiring the provider of information to register the resource by entering an index entry for the resource. Since the provider is responsible for preparing the index entry, there is the potential for its accuracy to be high. Without such an index, generated by the provider of the resource, the cost of centralised cataloguing for the increasing number of Internet resources would become prohibitively high.

The overwhelming quantity of web based textual information necessitates the development of semi-automatic tools to manipulate it. To answer this need, the natural language processing (NLP) community has harnessed together formerly independent technologies in order to build more sophisticated second-generation systems such as information retrieval engines, text summarisers, named entity extractors and question answering systems.

Recent developments in Open-Domain QA [20] have made it possible for users to ask a fact-based question in natural language (ex. *Who was the Prime Minister of Canada in 1873?*) and to receive a specific answer (*Alexander Mackenzie*) rather than an entire document where the users must further search for the specific answer themselves. In this respect, QA can be seen as the next generation of daily tools to search huge text collections such as the Internet.

Question Answering is not a new field. Work in this area dates back to the early days of artificial intelligence [12]. At that time, however, QA systems were applied to toy worlds and were not robust to deal with real-world applications. Since then, some work has been done independently [15].

However, a few years ago, two important factors strongly re-activated interest in QA. Core NLP techniques such as passage information retrieval, part-of-speech tagging and named entity extraction became ripe enough to allow the building of robust second generation systems, and the overwhelming need for real-world intelligent information access systems became apparent with the emergence of the information age heralded by the development of the Internet in general and the Web in particular.

The field of QA has been chiefly driven by the DARPA initiative through the TREC conferences [19], [20]. Most QA systems perform four tasks: *passage retrieval*, which identifies sections from pertinent documents that are more likely to contain the answer; *question analysis*, which tries to decompose the question according to a set of intentional and semantic features; *passage analysis*, which performs syntactic and semantic analysis of the passage; and *answer ranking*, which assigns a score to snippets of the passages according to their closeness to the question representation.

However, the inherent difficulties in analysing real texts has made the performance of these current systems is not high enough to be useful in every-day systems. Question analysis can be performed with greater confidence, as it requires the analysis of individual sentences rather than a whole discourse. Therefore problems of discourse analysis such as co-reference resolution, world-knowledge and complex syntactic analysis are not required.

The structure of Semantic Header is similar to the ones used for most libraries' indices and includes both intrinsic and extrinsic information deemed useful for on-line systems. The syntax of the semantic header is based on the SGML mark-up language. One advantage of the SH is that it could be used for the initial search for our QA system; it doesn't require the analysis of the source documents, since the SH contains the most often used search terms and essential summary of the source resource.

By using QA techniques to analyse questions, but substituting the document analysis by the use of the contents of the Semantic Headers, we believe that the performance of such a system would be high enough to be used on a daily-basis. This is precisely what we are trying to do: use techniques developed in the domain of question-answering to analyse incoming questions, but increase the precision of the answer by querying a structured database, rather than free-form texts. The structured database being based on analyzed bibliographic information of the underlying resources in the form of semantic header.

### B. Natural Language Interfaces to Databases

One of the earliest and most widely studied areas of NLP is the development of a natural language interface to database systems [9], [14]. Such front ends relieve the users of the

need to know the structure of the database and the interaction is more convenient and flexible for them. Due to the advantage its more convenient user-interface, this application of NLP is still widely popular today. The strength of such front-ends is typically measured using two criteria:

1. **The size of the accepted data base schema:** that is, the number and the complexity of the relations in the schema. For example, the SQ-HAL [17] system accepts one or two tables and a relationship.
2. **The accuracy of the answer:** that is, how close is the SQL query to the user's natural language question. High figures are often cited on some commercial systems; for example, the Elf system [8] claims a success rate of up to 90%, as compared to 10% for English Query (Microsoft). Elf can resolve simple queries as in "Show the domestic suppliers" or in "What are the names of our Canadian customers?" However, to our knowledge, no strict evaluation method has been proposed to standardize and compare the measure of accuracy of NLP front ends. Different systems use different evaluation metrics based on often unknown benchmarks. There being no standard evaluation in place, it is difficult to compare systems from different vendors.

Natural Language interfaces to database systems and question answering systems differ fundamentally in their scientific goals and their technical constraints. QA systems try to answer a NL question by analysing a collection of unrestricted and unstructured texts; while NL interfaces to DB have the advantage of dealing with structured texts, that is, texts that has already been structured semantically (person names, dates, ... are already organised into specific tables). However, both NL interfaces to DB and QA systems share an interesting similarity: they both take as input a question formulated in natural language and must interpret it in order to answer it. The goal of our project is to make use of the current techniques used in QA to analyse the question in the context of a NL front-end to a DB.

The advantage of the QA domain, is that standard metrics do exists to evaluate and compare different approaches. Currently, the best scoring systems can provide up to 50-60% correct hits when the answer is a passage 50 bytes long [22] (use patterns to find potential answers).

Both QA systems and NL interfaces systems are mainly based on a part-of-speech tagging of the words (or tokens), this is often followed by a partial syntactic analysis and then a more or less precise semantic interpretation. This third step, finding the meaning of the user input, is the most difficult part. Most NL interfaces and QA systems do semantic analysis one way or another. This is the area that mostly differentiates different NLP systems. In an open-domain situation, where the user can ask questions on any domain, this task is often very tentative and relies mainly on lexical semantics only. However, when the discourse domain is limited the

interpretation of a question becomes easier as the space of possible interpretation is smaller and templates can be used for the interpretations of questions [21].

Using templates is appropriate for simple questions, such as those found in [20]. A question, such as “*Who wrote The Old Man and the Sea?*” (where the verb has only one modifier) is expected to be interpreted with a higher success rate than a more complex question such as “*List all books on Algorithms written in the USA between 1960 and 1990*” (where the question has 3 modifiers, 1 for the noun and 2 for the verb).

As the complexity of the user input increases, the difficulty of developing a correct template increases. To increase the success rate, the system needs another source of information in addition to the templates.

We believe that the high level of organisation of the database, namely from the schema of the database can be used as an additional resource to interpret the question in a limited domain. This system addresses the problem of large answer sets, by restricting the search area. This can be done through better organising the knowledge base. A relational database is a good example of organising the data. The semantic analysis can be greatly improved by using the information from the data base schema. This approach combines the top-down and bottom-up approaches, in a sense that the token parsing and tagging is done as usual, but when it comes to the semantic interpretation, this is done partially based on the underlying database. In other words, regardless of the user input, the system knows in advance the limits of answers it can provide.

The two strategies described above are presented in Figure 1 and Figure 2. Figure 1 illustrates the general architecture of a NL interface system, whose knowledge base is more or less

## II. OUR PROPOSED ARCHITECTURE

The system initially analyses incoming questions formulated in NL. The architecture of the system is illustrated in Figure 3. The design of the NLP processor is elaborated in Figure 4. As illustrated in Figure 2, the questions are syntactically parsed using the Link Parser [18] and are finally represented by a 3-tuple: a question-verb, a focus and a set of modifiers. The 3-tuple representation of the question is similar to the work of [16] in question answering. Once the question is parsed, we map its 3-tuple representation to a meta-representation of the database constructed by the pre-processor and subsequently translate the NL question into a set of SQL queries.

The goal of the pre-processor is to establish a set of rules: the rules can be generic or specific to the database. The rules need to be generated only once and are used in the semantic analysis of the user input. The user can decide whether a specific rule can become a generic one. The pre-processor has

organized. The processing is linear, going through the stages detailed above. Figure 2 shows how the semantic interpretation can be improved by making use of the meta-information, i.e. the schema of the database.

The main difference between the two approaches shown in Figures 1 and 2 is the use of a pre-processor. The pre-processor reads the schema of the data base and a set of predefined and specific rules and produces the full set of rules specific to the target data base. The general goal of a portable system is to be database independent and precise. Unfortunately, these two goals are usually contradictory: the more specific rules lead to more precise results; however, the system, unless it can be trained easily, tend to becomes less portable.

Our NLP system uses templates to analyse questions and represent them into a 3-tuple formed of :

- a question verb
- a focus, which is mostly a direct object to the question verb
- a set of modifiers (possibly empty):

We believe that the identification of the role of the modifiers distinguishes our system from the NL interface systems mentioned previously [17], [21]. In SQL terms, the system will cross-reference more than 3 tables in finding out the result (or answer) set

Once the question has been analyzed, our system tries to match the results of the pre-processor to the 3-tuple-question representation to elements of the database, namely table names, attributes and values.

three steps: first it reads the schema of the database, then it identifies the table and attribute names and it creates a list of hyponyms, hypernyms and synonyms for each table and attribute name. The pre-processor has been integrated with WordNet [13]. WordNet accepts a token as input and returns sets of words that are semantically and lexically related to the input word (ex. synonyms, hypernyms, hyponyms) For each name the pre-processor uses WordNet to find the list of synonyms, hyponyms and hypernyms. Then it proposes this list to the user. The user can accept, rejects or modify the list.

The Cindi library system contains many relations. For simplicity we only consider three of these: *resource*, *author* and *resource\_author*. The table *resource* has the attributes *resource\_id*, *title* and *publish\_date*. The table *author* has the attributes *author\_id* and *name*. There are other attributes as well. The table *resource\_author* has two foreign keys: *resource\_id* and *author\_id*. The pre-processor reads the

schema of the database. Consider the synonyms for resource: for the library application, we guide the pre-processor to accept *book* as a synonym for *resource*. The pre-processor stores *book* and all its hyponyms, hypernyms and synonyms. Second, the user can produce rules that are specific to the given schema. For example, the schema for the Cindi library system has these tables: *resource*, *author*, *resource\_author*. The user produces this rule: if the tables *resource* and *author* appear in the SQL query, then the table *resource\_author* must appear as well. This rule determines the list of tables, and the list of conditions in the final SQL query. The NLP processor knows how to identify the foreign keys in the 3-table relation. This kind of rule is derived from the existing relations in the schema. Third, the user enters rules that are not covered by the schema. Example: an *author* presumably wrote a *book* (*resource*); the action verb *write* is related to both *author* and *resource* tables. The pre-processor uses WordNet to store all hyponyms, hypernyms and synonyms for the verb *write*. When the custom rules are first introduced to the pre-processor, the user can specify whether or not they are to be stored permanently. Permanently stored rules are generic rules.

The relational schema is object-oriented in the sense that each table represents an entity of the real life in most of the cases. Each entity can act, or be acted upon. In the example above, the *author* is the actor and *resource* is acted upon by the *author*. The *author* executes an action (*write*) and the *resource* receives the action. This is an example of a rule discussed above. Each object in the data base is assigned a list of actions. Example: (*author*, *write*, *A*) (*book*, *write*, *S*) where *A* stands for *Actor*, and *S* for *Spectator*.

Another class of rules are the identifying rules. They are based on the fact that each object can be addressed, or identified in a natural language by queries of the form: *who*, *which*, *how much*, etc. In our case, *author* is associated with *who*, *whom*. The rules are identified for table and attribute names.

For a given table, each attribute falls in one of these classes: time, date, distance, place, name, number, weight, other. As this list grows, the system becomes more and more capable. In our example, the attribute *name* is in class *name*.

The last class of rules is about the default attributes. When the user input is parsed, the NLP system identifies the direct object and its determiners. If a determiner cannot be associated with one of the existing attributes, it defaults to the base attribute, which is defined by the user. For example, the base attribute for *resource* is *title*, and for *author* it is *name*.

Let's consider the following NL query:

“Who wrote books on Algorithms between 1992 and 2002?”

As a result of the NLP parsing, the action verb is identified as being *wrote* (*write*) and the direct object is *books* (*resource*) The first determiner is *on Algorithms*, and the second

determiner is *between 1992 and 2002*. The NLP identifies the second determiner as being of type *date*. This is done through a set of built-in rules, in the NLP. The table *resource* has an attribute in the class *date* which is called *publish\_date*.

The question word *who* is related to a person (*author*). The action verb is *write*. Because *author* can *write books*, and *book(s)* is related to *resource*, the table *resource* is involved as well. Because of that, and based on the rules presented above, the table *resource\_author* and the corresponding foreign keys are added. Because there is no determiner for *author*, the attribute *name* is used. Because *Algorithms* as a determiner to *resource* (*book*) cannot be associated with any of the attributes for *resource*, the *title* is considered. The NLP processor processes the dates. It identifies the numbers as year dates, and then it identifies a date interval, with lower date *1992* and higher date *2002*. These dates are determiners for *books* (*resource*); we already saw that the table *resource* has an attribute in the class *date*, which is *publish\_date*.

Based on these rules, the NLP processor builds the following SQL query:

```
SELECT author.name FROM author, resource,
resource_author
WHERE resource.title="Algorithms"
AND resource.publish_date > 1992
AND resource.publish_date < 2002
AND resource_author.resource_id=resource.resource_id
AND resource_author.author_id=author.author_id
```

### III. IMPLEMENTATION

#### A. The OO Approach

The NL system is implemented using a platform-independent, object and data oriented approach. The main objects are the user interface, the HTTP server, the NL processor, the NLP pre-processor and the RDBMS engine. The User Interface is based on the PHP language, and communicates with the Apache server. PHP allows us to execute the NLP ANSI C code on the server side. Once the SQL query has been generated by the system, it is passed on to the RDBMS engine. The results set is then dynamically formatted through PHP and sent back to the user.

#### B. Integration of existing tools and techniques

The NL pre-processor reads the schema of the database and builds a set of rules. In the example given previously with the Cindi library system:

```
Table resource (resource_id, title, publish_date)
Table resource_author (resource_id, author_id)
Table author (author_id, name)
```

The pre-processor is integrated with the run-time interface of WordNet. In this example, the pre-processor uses WordNet to find the list of hyponyms, hypernyms and synonyms for each of the following tokens:

#### Table resource

resource	table name	(modified by user: search for book instead)
resource_id	attribute	(rejected by user, no list)
title	attribute	
publish_date	attribute	(modified by user: search date instead)

#### Table author

author	table_name	
author_id	attribute	(rejected by user, no list)
name	attribute	

#### Table resource\_author

resource_author	table_name	(rejected by user, no list)
resource_id	attribute	(rejected by user, no list)
author_id	attribute	(rejected by user, no list)

For each token, the pre-processor proposes a list of hyponyms, hypernyms and synonyms to the user. The user can accept, reject or modify the list. In the *resource* case, the user introduces a new keyword: *book*. WordNet returns the list of synonyms for *book*.

The next step is to produce the rules, as described in Section 2. These rules are database specific:

#### Rule for the relations:

(author, resource) then (resource\_author) and the foreign keys in resource\_author match the corresponding primary keys in author, resource.

The rule for relations indicate that any use of two of the three names will imply the third one as well. In SQL terms, if the list of tables contains two of the three tables, the third one is to be expected. This kind of rule is based on the relation between the three tables, as described by the schema of the database.

#### Rules for the action verb:

(author, write, A)  
(resource, write, S)

#### Rules for accepted questions:

(author, who)

#### Rules for the object class:

(author, name, name)  
(resource, publish\_date, date)

The rules generated by the pre-processor are used at the run time by the NLP processor. The NLP processor has been integrated with the Link Parser [18]. The user input is tokenized and tagged. The Link Parser finds the possible links between tokens. The NLP processor uses these links to identify the question word, the action verb, the direct object and its determiners. The NLP can also process sentences that start with an imperative verb, as in the query “List all books on Mathematics”

Each determiner can be composed of one main token, and one or more connection words to the determined object. For example, in “List all books written by Leiserson after 1985” the main object is *books*. In this case the determiners are for the action verb. They are *by Leiserson* and *after 1985*. *By* and *after* are the connection words. The verb *written* is related to *resource* and to *author* but because of the connection word *by* only *author* is considered at this time. This is because of the rule (author, write, A). On the contrary, *after 1985* is related to both *resource* and *author* and thus both tables are involved. If both tables contain an attribute named *date*, like in *author\_birthdate* and *publish\_date* the system might fail to pick the right attribute. This is one of the uncertainties we faced. This kind of error was addressed by introducing a new rule in the pre-processor: the attribute *publish\_date* was related to the verb *write*.

The result of the sentence parsing is a list of tokens, the associated part-of-speech tags and the syntactic links between tokens. Figure 5 shows another example of a SQL generation with the example database consisting of 3 tables: *resource*, *resource\_author* and *author*.

The NL processor used the Link Parser to identify the tokens from the question and their syntactic links, then it picked the best choice of link, based on its degree of confidence on the parse tree as computed by the Link Parser

The next step was to match the parsed input with the 3-tuple template: a question verb (element), a focus object, and some modifiers.

Some sentences cannot be parsed due to a limitation in the Link Parser, as it is the case in the following example:

All books on Mathematics by David Helm.

To handle such cases, our NLP system adds an imperative action verb as indicated by the modified query given below:

Show all books on Mathematics by David Helm.

## IV. PRELIMINARY RESULTS

The accuracy of the system is currently being tested on the database for the Cindi system. The schema contains 15 tables

(relations) The natural language input goes from 3 to about 20 words. By creating additional rules, and by improving the mechanism of generating the generic rules, we intend to raise the capabilities of the NLP system beyond these limits.

The accuracy of the system depends on three factors: the accuracy of WordNet, the accuracy and number of links generated by the Link Parser, and on the precision of the rules generated by the pre-processor. Figure 5 gives an example of how links are generated.

At the present time, we estimate a success rate of up to 70% for questions having one action verb, one direct object and two modifiers. For three determiners, we estimate a 55% success rate. This is due to the quality of the links found by the Link Parser, and to the quality of the rules generated by the pre-processor. We plan to improve both. When the number of modifiers increases, i.e., the input phrase gets more complex, the Link Parser tends to generate an increased number of possible links. This is a normal trend. The NLP needs to be improved in this area, so as to make a better selection of the possible links offered by the Link Parser.

However, we expect that the quality of the results will depend on:

1. The size of the database: the larger the schema of the data base, the lower the accuracy is expected to be.
2. The specificity of the rules: The less specific the rules are, the lower the accuracy is expected to be (but the higher the portability of the system).
3. The role of the pre-processor: The more specific rules and code is in the pre-processor, the higher the accuracy is expected to be (but the less portable the system will be).

Other independent factors:

1. The quality of the results returned by WordNet has impact on the NLP QA system. More results WordNet returns, lesser accuracy the NLP has.
2. The Link Parser may return many possible links for a complex sentence. The greater the number of possible link sets, the lesser the accuracy is.

### Limitations

Our system is not a generalized question-answering system as those developed by the TREC community. It is designed to be tuned to a given database. Once tuned, there is a dependency between the system and the targeted database. However, this dependency is localized in the pre-processor.

### V. CONCLUSION AND FUTURE WORK

The NLP system uses a combined top-down and bottom-up approach. The top-down approach is used in the early stages of input parsing, in order to decompose the user input into a tuple (question-focus-determiners). The bottom-up technique

is involved in the semantic parsing; in a sense the system improves the quality of the parsing by using pre-computed information. The pre-computed information is based on the actual structure of the database.

If the data is highly organised, as is the case of a RDBMS, the search domain can be reduced, and thus the precision of the QA system can be improved. The reduction of the search domain is made possible by the simple fact that the system knows, through the rules, in advance what it can answer.

We plan to associate probability measures to the constructed SQL queries using our confidence level in the lexical and semantic relations used by the pre-processor and to the analysis of the question. Our focus is on improving: the link selection mechanism; the generation of the rules in the pre-processor; and the NLP built-in rules.

### Acronyms

<b>DARPA</b>	<b>Defence Advanced Research Projects Agency</b>
<b>DB</b>	<b>Data Base</b>
<b>DBMS</b>	<b>DB Management System</b>
<b>HTTP</b>	<b>Hyper Text Transfer Protocol (world wide web protocol)</b>
<b>NL</b>	<b>Natural Language</b>
<b>NLP</b>	<b>NL Processor</b>
<b>OO</b>	<b>Object Oriented</b>
<b>QA</b>	<b>Question Answering</b>
<b>RDBMS</b>	<b>Relational DBMS</b>
<b>SH</b>	<b>Semantic Header</b>
<b>SQL</b>	<b>Structured Query Language</b>
<b>TREC</b>	<b>Text REtrieval Conference</b>
<b>WWW</b>	<b>World Wide Web</b>

### REFERENCES

- [1] Brill, E. (1995) Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics* 21(4): 543-565.
- [2] Clarke, C.L.A., Cormack, G.V., Kisman, D.I.E. and Lynam, T.R. (2001) Question Answering by Passage Selection (MultiText Experiments for TREC-9). *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*.
- [3] Desai, B.C. (1997) Supporting Discovery in Virtual Libraries, *Journal of the American Society of Information Science*, 48-3, pp. 190-204.
- [4] Desai, B.C., Shinghal, R., Shyan, N. and Zhou, Y. (1999) CINDI: A System for Cataloguing, Searching, and Annotating Electronic Documents in Digital Libraries, *Proceedings of ISMIS'99*, Springer-Verlag, Warsaw, Poland, June, pp. 154-162.
- [5] Desai, B.C., Shinghal, R., Shyan, N. and Zhou, Y. (1999) CINDI: A System for Cataloguing, Searching, and Annotating Electronic Documents in Digital Libraries, *Library Trends*, Summer 1999, 48(1), pp209-233.

- [6] Desai, B.C., Haddad, S.S. and Ali, A. (2000) Automatic Semantic Header Generator, *Proceedings of ISMIS'2000*, Springer-Verlag, Charlotte, NC, pp.444-452.
- [7] Desai, B.C. (2002) Search and Discovery on the Web, *Proceedings of SSGRR'02*, July, L'Aquila, Italy.
- [8] Elf, <http://www.elf-software.com/FaceOff.htm>, site visited in May 2002.
- [9] Gal A. and Minker, J. (1985). A Natural Language Database Interface that Provides Cooperative Answers. In *Artificial Intelligence Applications*. C.R. Weisbin (ed.), IEEE Press, Washington.
- [10] Harabagiu S. et al., (2001): FALCON: Boosting Knowledge for Answer Engines. *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*.
- [11] Hovy E. et al. (2001): Question Answering in Webclopedia. *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*.
- [12] Lehnert, W. (1978) *The Process of Question-Answering*, New Jersey.
- [13] Miller, G. (1995) WordNet: a Lexical Database for English, *Communications of the ACM*, 38 (1), November, pp. 39-41.
- [14] McTear, M. (1987). *The Articulate Computer*. Oxford, England: Basil Blackwell.
- [15] Molla, D., Berri, J. and Hess, M. (1998) A Real World Implementation of Answer Extraction, In: *Proceedings of DEXA Workshop*, pp. 143-148.
- [16] Plamondon, L. and Kosseim, L. (2002) QUANTUM: A Function-Based Question Answering System. . To appear in: *Proceedings of The Fifteenth Canadian Conference on Artificial Intelligence (AI 2002)*.
- [17] Ruwanpura, S. (2000) SQ-HAL: Natural Language to SQL Translator <http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura>, Monash University, site visited in May 2002.
- [18] Sleator D., Davy Temperley, D. (1993) Parsing English with A Link Grammar. *Proceedings of the Third Annual Workshop on Parsing Technologies*.
- [19] *TREC-9 (2000)*. *Proceedings of the Ninth Text Retrieval Conference*, November, Gaithersburg, Maryland, USA (available at [http://trec.nist.gov/pubs/trec9/t9\\_proceedings.html](http://trec.nist.gov/pubs/trec9/t9_proceedings.html))
- [20] *TREC-X (2001)*. *Proceedings of the Tenth Text Retrieval Conference*, November, Gaithersburg, Maryland, USA (available at [http://trec.nist.gov/pubs/trec10/t10\\_proceedings.html](http://trec.nist.gov/pubs/trec10/t10_proceedings.html))
- [21] Watson, M. NLBean(tm) version 4: a natural language interface to databases! [www.markwatson.com](http://www.markwatson.com). Site visited in May 2002.
- [22] M.M. Soubbotin, S.M. Soubotin (TREC-X 2001) Patterns of Potential Answer Expressions as Clues to the Right Answers, see reference *TREC-X (2001)*

The list of figures used in the paper

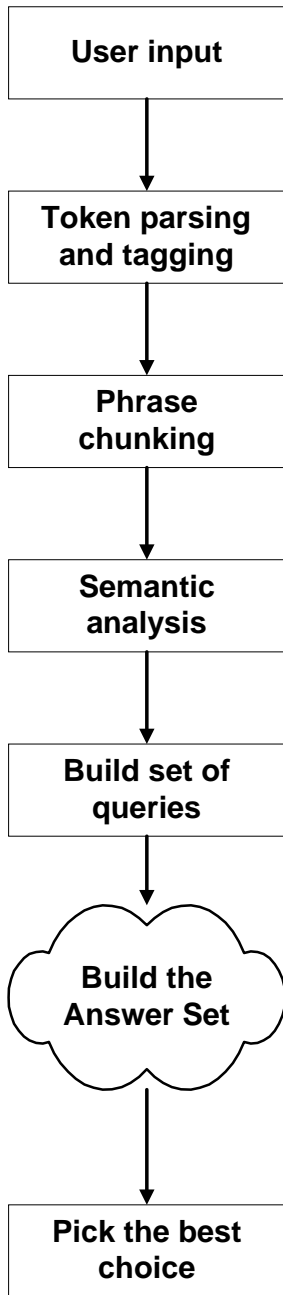


Figure 1: The process flow for a NL interface to the data base

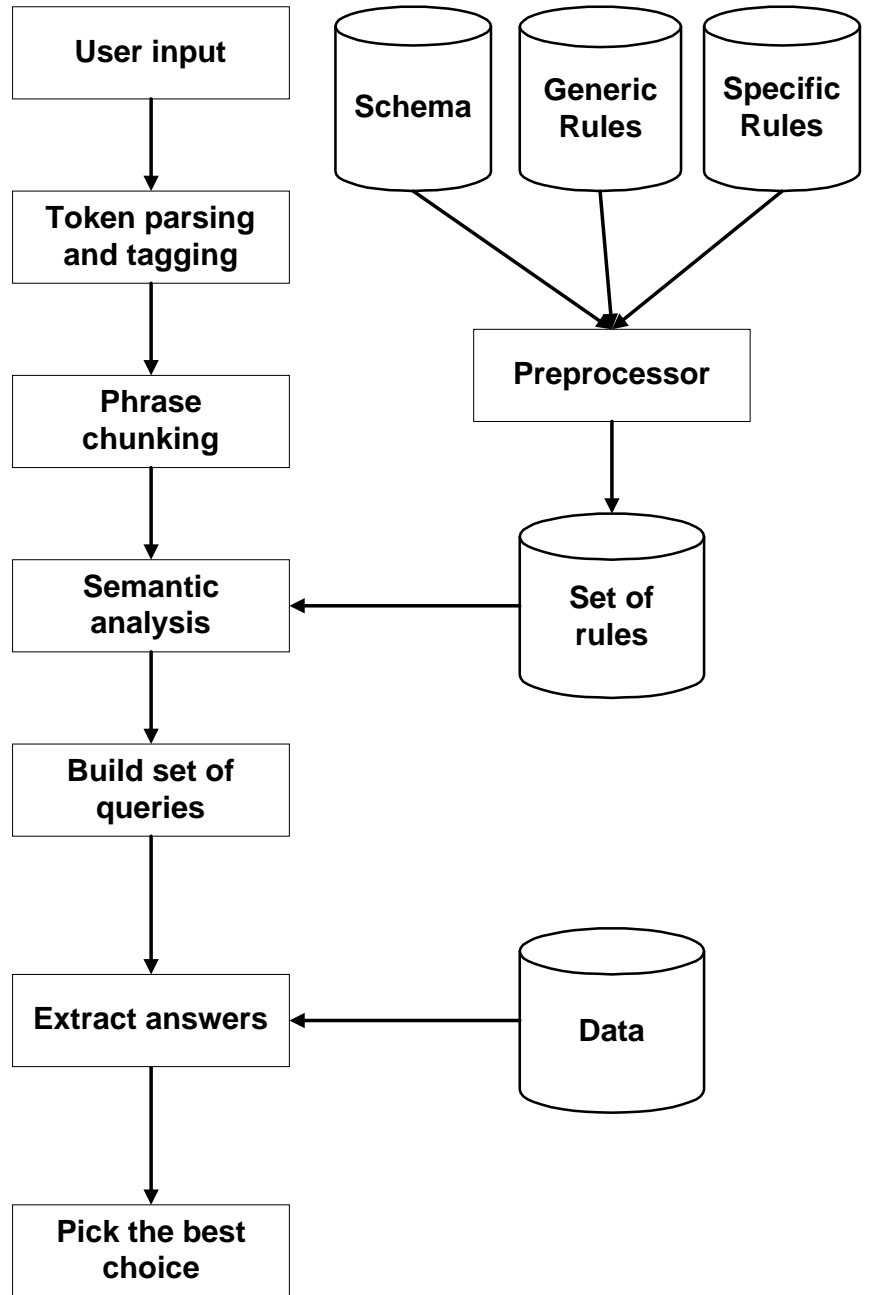
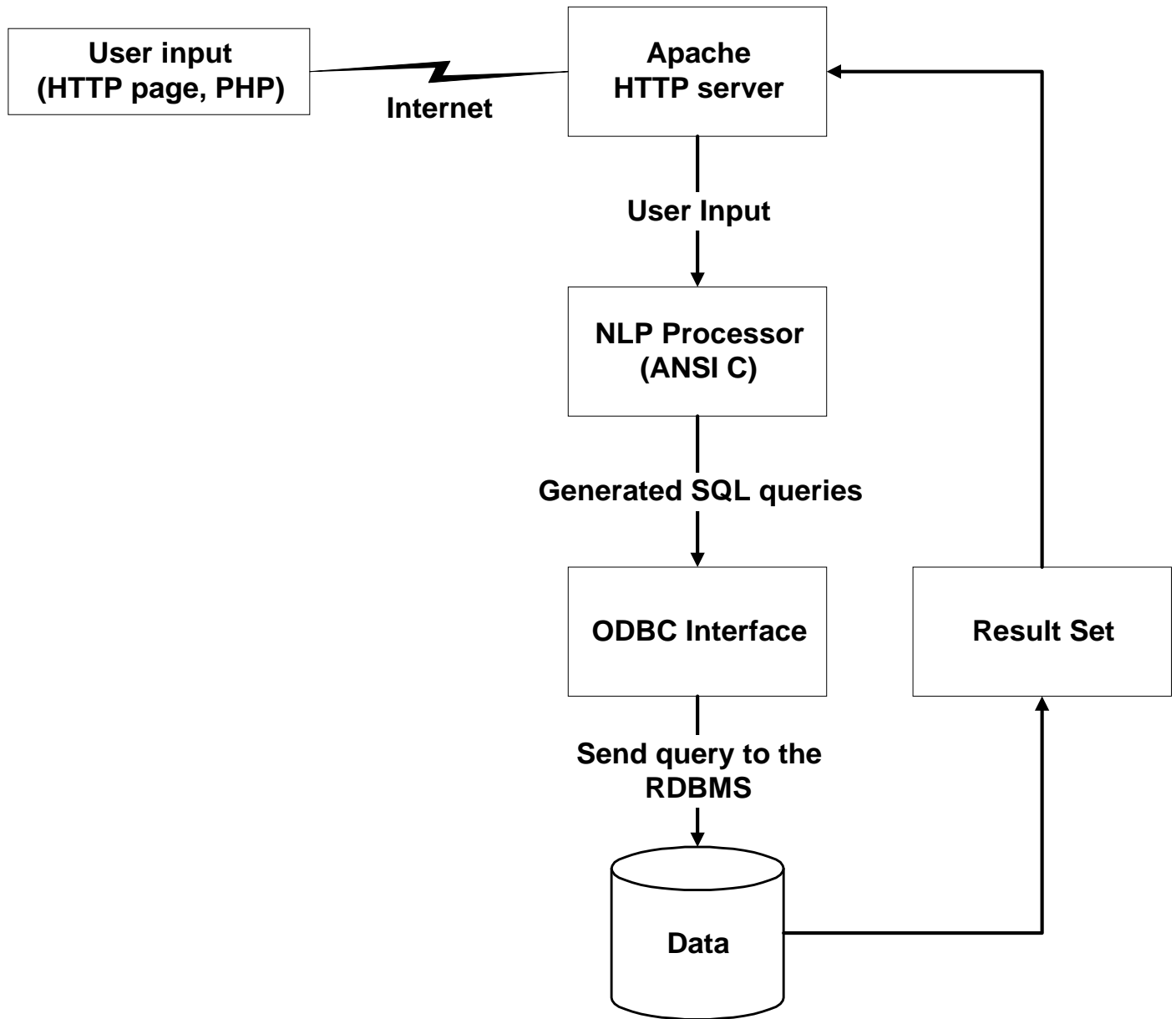


Figure 2: The process flow for our NL interface to the data base. The Semantic parsing is making use of the schema.



**Figure 3 The architecture of the NLP QA system (platform independent)**

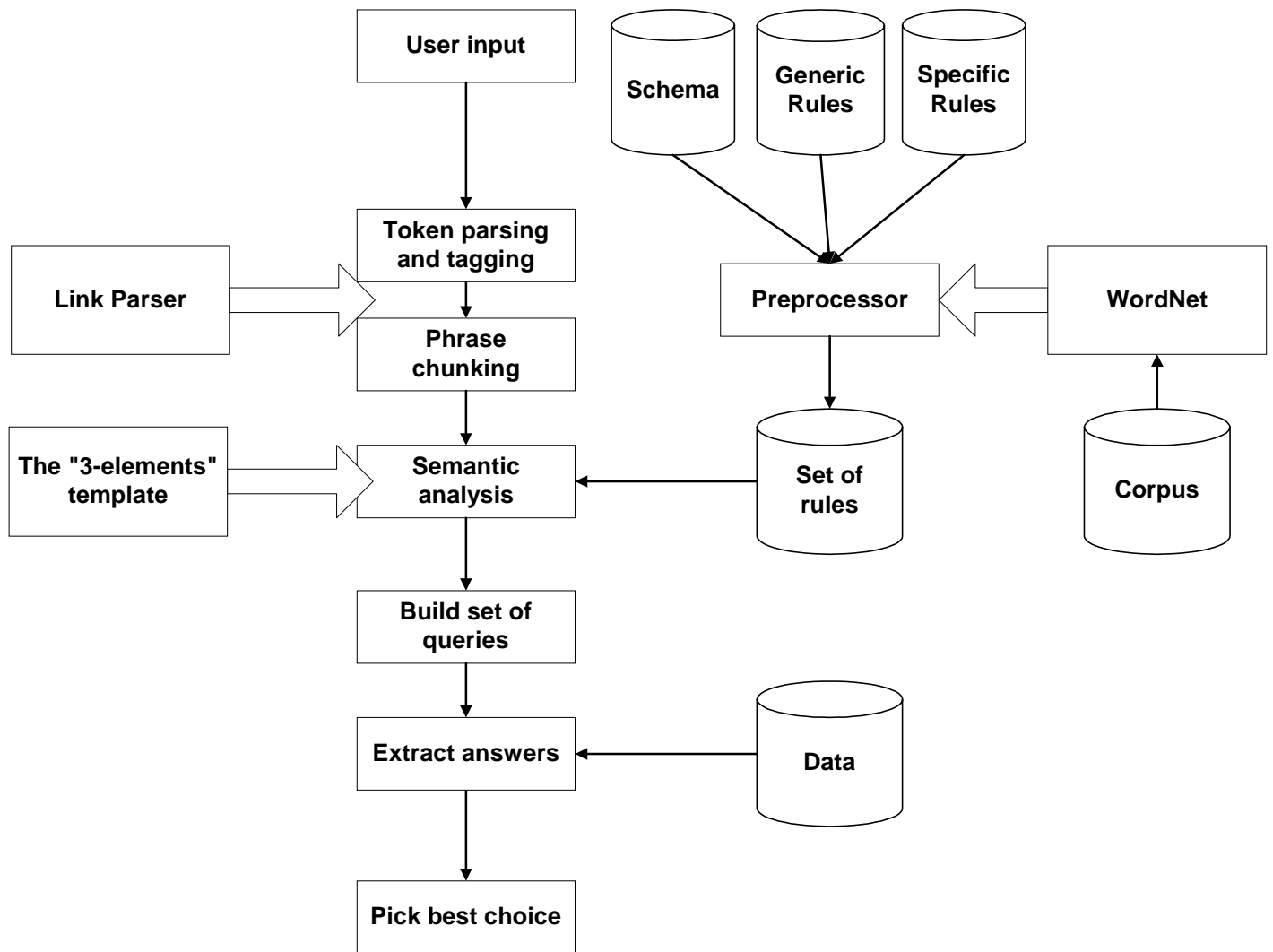


Figure 4 The design of the NL Processor

```

linkparser> list all books written by Leiserson.

NLP: print the link information:
LEFT-WALL - Xp - . : Connects to the period at end of the sentence
LEFT-WALL - Wi - list : connect imperative to the wall
list - Op - books : Connects transitive verb to direct or indirect objects
(p)
all - Dmc - books : Connects determiners to countable nouns
books - Mp - by : Prepositional phrase modifying noun
books - Mv - written : connects noun to participle modifier
by - Js - Leiserson : Connects preposition to its object
. - RW - RIGHT-WALL : Connects the right-hand wall to the left-hand wall
Imperative verb = list
Direct object = books

```

---

```

The actual SQL query is:
SELECT author.name FROM resource,resource_author,author
WHERE resource_author.resource_id=resource.resource_id AND
resource_author.author_id=author.author_id AND author.name='L
eiserson'
=====
Linkage 3, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=8)

+-----Xp-----+
|           +---Op---+-----Mp-----+
+---Wi---+   +-Dmc+---Mv---+   +---Js-+
|         |         |         |         |
LEFT-WALL list.v all books.n written.v by Leiserson .

```

Enter query:

Figure 5: Example of an SQL generation by the system