Natural Language Database Interface for the Community Based Monitoring System*

Krissanne Kaye Garcia, Ma. Angelica Lumain,
Jose Antonio Wong, Jhovee Gerard Yap, Charibeth Cheng

De La Salle University – Manila. 1410 Taft Avenue Malate, Manila
{10533834, 10517197, 10514848, 10506349, koc}@dlsu.edu.ph

Abstract. In most information systems, databases are accessed and manipulated typically through systems developed to tailor-fit the company's needs. The usual problem in these cases is the limitation on data accessibility because the users are constrained to the forms created for the system. Another way of accessing the database is through Structured Query Language (SQL), a language that is not familiar to end users, thus still limiting the access to the data. Natural Language Database Interfaces were developed to address limited database accessibility for end users. This paper presents AlLaDIn, a web-based natural language interface to the Community-Based Monitoring System of a city in the Philippines.

Keywords: natural language database interface, SQL, CBMS, natural language query

1. Introduction

CBMS is a poverty monitoring system that is now used in Pasay City. It tracks poverty by surveying the people living in a certain area. Once this is done, they input them using the programs Census Professional (CSPro) and CMS-Natural Resource Database (NRDB), which are customized free software used to encode the data and to digitize spot maps. The output of CSPro is a text file, which is then used as an input to the program STATA, which they use to actually statistically monitor poverty. Since STATA needs technical skills to be used effectively, the data is very hard to access. Thus, people who actually must have direct access to these pieces of information must first ask the people/person who are/is capable of using the software to get the data for them, which might take some time. To address this problem, a Natural Language Database Interface (NLDBI) was developed for CBMS that will allow users to access the CBMS data without having to learn STATA.

A NLDBI allows users to access a database using natural language query. It accepts a user query, extracts pertinent data from the query, converts the extracted data to SQL, then retrieves the data from the database. AlLaDIn is domain-dependent, database-dependent natural language interface for CBMS-Pasay.

2. System Architecture

AlLaDIn is concerned with translating the English query by the user into its corresponding SQL query to retrieve the data from the database. The result will then be displayed to the user. This system has four major modules, as shown in the architectural design of the system in Figure 1. This design is partially based on the Masque/SQL system (Androutsopoulos *et al.*, 1995) and the FST mapping technique (Gala, 2005).

2.1 Parser and Semantic Analyzer

This phase involves the (1) the tokenization of input, (2) parsing and analyzing the semantics of

words, (3) retrieval of syntactic and semantic properties of words in the lexicon, and (4) checking the grammatical and semantic rules of the input. A context-free grammar for the English language is used to check if the sentence is grammatically correct.

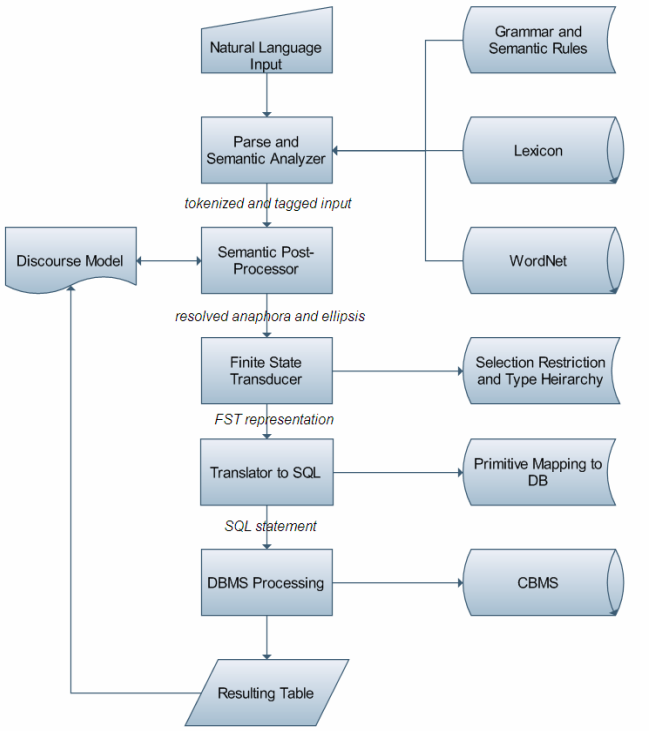

Figure 1: Architectural Design of NLDBI-CBMS

Once the sentence has been inputted, the Parser and Semantic Analyzer would tokenize the sentence. Each token would then be passed to WordNet for it to be properly tagged with its part-of-speech (POS), other information tag (eg. singular, person), and the token's synonyms. After being tagged by WordNet, it would then be passed to the lexicon. The lexicon contains domain-specific words or terminologies that are not found in WordNet. It would then tag each token that was not tagged by WordNet. After tagging all tokens, the sentence would then be parsed and checked if it is grammatically correct.

2.2 Semantic Post-Processor

Once the natural language query is accepted and confirmed by the system as grammatically and semantically correct, the query is then passed to the semantic postprocessor. The semantic post-processor is necessary for queries present the referencing problems of anaphora and ellipses (Androutsopoulos *et al.*, 1995). Consider the compound user query `"How many people live in Barangay 1? What about Barangay 2?"` The second query is ambiguous, however, the handling of such ambiguities presents a simpler way of querying for the users by allowing follow-up questions to be used and short.

If the user's input query does not have any anaphoric expression or ellipsis, the query is scanned. The query would be looked up in the discourse model to fill up the succeeding queries that have to be further resolved.

Template of the discourse model is shown in Table 1 and a sample discourse template is shown in Figure 2.

Table 1: Discourse Model Template.

Tag	Possible values	Description
`<Discourse>`		Marks the start of a discourse entry
`<Sentence>`		Marks the start of an input sentence
`<Token>`		Marks the start of a token in a sentence
`<Term>`	`person`	Indicates the term in the token

<POSTag>	noun	Indicates the term's part-of-speech tag
<OtherInfo>	singular, common, number, plural, proper}	Indicates other information related to the token
<Synonym>	Members	Lists all possible synonyms of the token. These synonyms are taken from WordNet if the token is in WordNet, otherwise, these synonyms came from the lexicon.

Figure 2. Sample Discourse template.

2.3 Finite State Transducer

A finite state transducer (FST) consists of states that maps one string to another string. It transduces or replaces a certain string to another.

AlLaDIn uses FST to replace keywords extracted with their corresponding SQL terms and database table attributes and conditions. The FST is represented by the rules defined in the Selection Restriction and Type Hierarchy, and the Translation Rules.

The Selection Restriction and Type Hierarchy is in charge of looking for words that are searchable in the CBMS Database. These words were manually created based on the CBMS database schema. The words that we placed here are those that have a particular mapping to the database, whether it be a table, attribute, or a condition. A sample of this is found in Table 2.

The Translation Rules is a table containing SQL keywords with a list of its corresponding English terms. These English terms are just synonyms of the SQL keyword's English literal. For example, the SQL keyword `max` has an English literal `maximum`, the list of English terms were derived from the synonyms of `maximum`. A sample of this is found in Table 3.

Table 2: Sample entries of the Selection Restriction with the mapping.

word	map1
boarder	BOARDERS
tenant	BOARDERS
calamity	CALAMITIES
cleanliness program	CLEANPROGRAMS
computer income	COMPUTERINCOME
construction income	CONSTRUCTIONINCOME
credit program	CREDITPROGRAMS

Table 3: Sample entry of the Translation Rules.

rule	output	synonyms
avg_stmt	avg	,average,
between_stmt	between	,between,between,betwixt,tween,
count_stmt	select count(*)	,count,count,number,enumerate,numerate,matter,weigh,consi
equal_stmt	=	,equal,equal,be,touch,rival,match,equalize,equalise,equate,
except_stmt	except	,except,demur,except,exclude,leave out,leave off,omit,take o
exists_stmt	exists	,exists,
greater_stmt	>	,greater,greater,bigger,older,more,more(a),more than,
having_stmt	having	,having,have,have,have got,hold,feature,experience,receive,
less_stmt	<	,less,less(a),smaller,younger,
like_stmt	like	,contains,begins,
max_stmt	max	,maximum,maximal,maximum,biggest,most,most(a),largest,
min_stmt	min	,minimum,minimal,minimum,smallest,least,least(a),
select_stmt	select	,select,choose,take,select,pick out,show,show,demo,exhibit,p

2.4 Translator to SQL

The tokens here are first replaced by their table attributes. This is done by passing each token term as a parameter to the Primitive Mapping to DB, which will return their mapping(s). The Primitive Mapping to DB is basically a table, which contains the word and their corresponding mappings. This is shown in Table 2. The first column represents the token while the other column represents the mappings. There are 40 more columns here that represent the mappings of the token, but due to space constraint it cannot be displayed all here. 41 was used as the maximum number of mappings because there are 41 tables in the CBMS database and we assumed that a token can only be mapped to one attribute per table.

After replacing all keywords with their table attributes, the Translator to SQL module will group the keywords into two groups, namely, the *Object* group and the *Components* group. The Object group contains the `select` and `from` clauses, while the Components group contains the `where` clause.

3. Simulation

To further understand how each module works, the inputs and outputs of each module will be shown based on the query "`How many people live in Barangay 1? What about Barangay 2?`".

3.1 Paser and Semantic Analyzer

As the sentences enter the Parser and Semantic Analyzer module, the sentences are separated so processing may be performed one sentence at a time. Figure 3 shows the sentences after separation.

```
[1]     How many people live in Barangay 1?
[2]     What about Barangay 2?
```

Figure 3: Sentences after being separated.

After separating the sentences, the sentences would then tokenized and passed to WordNet for part-of-speech tagging and retrieve of semantic information and synonyms. The underlined entries in Table 4 are the data taken from WordNet.

Table 4: The extracted tokens with WordNet information.

Term	Part-of-Speech tag	Other Information tag	Synonyms
How many	WH	Null	Null
People	Noun	Plural	Members
Live	Verb	Null	Null
In	Preposition	Null	Null
Barangay 1	Unknown	Null	Null
What about	WH	Null	Null
Barangay 2	Unknown	Null	Null

As seen in Table 4, "Barangay 1" is tagged as unknown. This is because WordNet does not contain the domain-specific terminologies used by the CBMS, which can be found in the lexicon. Table 5 shows the tokens with the lexicon information (underlined entries).

After tagging all tokens, the sentences will be parsed to check if the sentences are grammatically correct. If one of the sentences is grammatically incorrect, the sentences proceed to the Post-Semantic Analyzer for anaphora or ellipsis resolution.

Table 5: The extracted tokens with WordNet and lexicon information

Term	Part-of-Speech tag	Other Information tag	Synonyms
How many	WH	Null	Null
People	Noun	Plural	Members
Live	Verb	Null	Null
In	Preposition	Null	Null
Barangay 1	Noun	Singular	Null
What about	WH	Null	Null
Barangay 2	Noun	Singular	Null

3.2 Semantic Post-Processor

Upon entering the second module, the system checks if the sentences contain an anaphora or an ellipsis for them to be resolved. Consider the second sentence "*What about Barangay 2?*". This sentence contains an ellipsis and need to be processed. The Semantic Post-Processor determines the missing components of the sentence with ellipsis from the previous query. In our example, the previous query is "*How many people live in Barangay 1?*". The processed sentence is shown in Table 6.

Table 6: Query 2 after ellipsis resolution.

Term	Part-of-Speech tag	Other Information tag	Synonyms
How many	WH	Null	Null
People	Noun	Plural	Members
Live	Verb	Null	Null
In	Preposition	Null	Null
Barangay 2	Noun	Singular	Null

The Semantic Post-Processor compares each token of the previous query with the current query, and then merging distinct tokens from both queries into one to formulate the new and resolved query. However, the result of this process might not always be correct or complete.

3.3 Finite State Transducer

In the finite state transducer module, tokens are replaced, when possible, with their corresponding SQL equivalents based on the Translation Rules table. In our example, **`how many`** was replaced **`select count(*)`**. Its part-of-speech tag is then changed to SQL to indicate that the token is already an acceptable SQL token. Other tokens are replaced with

database-specific tokens, based on the Selection and Type Hierarchy entries and the synonyms of the tokens. For our sample queries, the token **`people`** was replaced to the CBMS table **`member`**, since it has a synonym of **`members`**. Its part-of-speech tag is then changed to **`selection restriction`**. Tokens that are tagged as `adverb`, `adjective`, or `prepositions` are filtered out of the token list. Table 7 shows the resulting set of tokens for both sentences.

Table 7: Tokens after passing through FST.

Term	Part-of-Speech tag	Other Information tag	Synonyms
select count(*)	SQL	Null	Null
member	Selection Restriction	Plural	Members
live	Verb	Null	Null
Barangay 1	Noun	Singular	Null
select count(*)	WH	Null	Null
member	Selection Restriction	Plural	Members
live	Verb	Null	Null
Barangay 2	Noun	Singular	Null

3.4. SQL Translation

The fourth module, *Translator to SQL*, converts those tokens tagged **`Selection Restriction`** to its corresponding SQL syntax. Each token is checked in the *Primitive Mapping to DB* if it has a corresponding SQL syntax. After which, all tags that are not tagged as **`SQL`** are removed from the token list. Table 8 shows the partial output of the module.

Table 8: Partial output of SQLTranslator module.

Term	Part-of-Speech tag	Other Information tag	Synonyms
select count(*)	SQL	Null	Null
MEMBERS	SQL	Null	Null
MEMBERS.brgy = 1	SQL	Null	Null
select count(*)	SQL	Null	Null
MEMBERS	SQL	Null	Null
MEMBERS.brgy = 2	SQL	Null	Null

The tokens will then be separated to the Objects and Component groups. Table 9 shows the separation of tokens into groups.

Table 9: Object and Component groups.

Object	Component
select count(*)	MEMBERS.bgry = 1
MEMBERS	

After grouping the tokens, it will then converted to its SQL statement which is *`select count(*) from MEMBERS where MEMBERS.brgy = 1`*.

4. Results and Conclusion

AlLaDIn is domain-specific natural language database interface. The information that it knows is heavily dependent on the data stored in and the structure of the CBMS database. If the database schema is revised and/or when the data is updated, it is not guaranteed that the system would still work perfectly. Entries in the lexicon came from the CBMS data, which means that it needs to be update manually through the Administrator module when the CBMS database is updated. The *Primitive Mapping to DB* table relies heavily on the database schema and if the schema is changed, it also has to be revised.

The Parser and Semantic Analyzer module accepts simple interrogative and imperative sentences. This module is dependent on WordNet and the lexicon in recognizing and tagging words in the given sentence. If a word in the sentence is not matched or is matched but tagged incorrectly, the system may generate an incorrect SQL statement thus generating incorrect results or no results at all.

The Semantic Post-Processor module, in regards with sentence ambiguity detection and resolution process, the module does not have any sentence ellipsis detection mechanism and the resolution process is still imperfect. In the sentence ellipsis detection, the module just assumes that every input natural language is an ellipsis sentence ambiguity problem. Therefore, all input undergoes the resolution process even though the input natural language originally is not an ellipsis. And with the resolution process, both resolution processes uses simple algorithms that might not be enough to handle all possible cases. So there is a possibility that the input natural language might not be resolved well, especially those inputs that are too complex. To further improve the module, future works can be done to find a proper way of detecting the ellipsis sentence ambiguity problem and to develop resolution algorithms that can handle and resolved complex inputs.

The Finite State Transducer is able to handle the words that are found in the CBMS database and the translation rules. If a given word is synonymous (according to WordNet) to a word found in the database, the system would be able to successfully process it.

The Translator to SQL works as expected if at least one of the tokens is the query refer to a table found in the CBMS database. If the tokens do not contain any database-specific attribute, a default set of attributes is used by the system per table. The translator is also capable of distinguishing if an attribute is used as an attribute or as a condition. For example, the query "`List the age of the people.`" and "`List the people with age greater than 30.`" In the first query **`age`** is used as an attribute, while in the second query **`age`** is used as a condition. Although the system can handle this, it cannot distinguish if there are 2 or more attributes used as conditions. If a query is ambiguous (i.e. tokens may be mapped to 2 or more tables), the translator will be able to create 2 or more queries and determine which attributes and/or condition belongs to a specific query.

To further improve AlLaDIn, it is recommended that the parser handle compound sentences. Reliance of the system to CBMS data may also be lessened by automating the lexicon update whenever the CBMS data has been updated. Ellipsis resolution and detection may also be improved. Finally, the tokenization process may be enhanced. Since the conversion of the tokens to SQL is highly dependent on the tokenization phase, it could be further improved if the system can distinguish two or more attributes used as conditions.

5. Reference

Androutsopoulos, G.D., Ritchie, & P. Thanisch, "Natural Language Interfaces to Databases – An Introduction". *Natural Language Engineering*, 1(1), Cambridge University Press (1995).

CBMS Brochure (2003). Philippine Institute for Development Studies. Makati, Philippines.

CBMS Network Conference Overview.(2006) The CBMS Network Conference 2006: Philippine Institute for Development Studies. Makati, Philippines.

CBMS Network Coordinating Team (2005). Gaining Insights on the CBMS Application: The Case of the Philippines Proceedings of the 2004 National Conference on CBMS. *2004 National Conference on CBMS* (pp. 2-51). Makati City: CBMS Network Coordinating Team.

CBMS Network Project Progress Report. (2004). Angelo King Institute for Economic and Business Studies, De La Salle University, Manila.

Gala, S. (2005). *Translation From English to Sql Using Statistical Machine Translation and Finite State Transducers*.[online] Available: http://tangra.si.umich.edu/clair/shyamg/project.pdf