International Journal of Computer Applications (0975 – 8887)
Volume 107 – No 5, December 2014

An Overview of NLIDB Approaches and Implementation for Airline Reservation System

Manju Mony
M.E. Student (Comp Engg.)
K. J. Somaiya CoE
Vidyavihar, Mumbai

Jyothi M. Rao
Dept. of Computer Engineering
K. J. Somaiya CoE
Vidyavihar, Mumbai

Manish M. Potey
Dept. of Computer Engineering
K. J. Somaiya CoE
Vidyavihar, Mumbai

ABSTRACT
Relational databases are queried using database query languages such as SQL. Natural language interfaces to databases (NLIDB) are systems that translate a natural language sentence into a database query. In this modern techno-crazy world, as more and more laymen access various systems and applications through their smart phones and tablets, the need for Natural Language Interfaces (NLIs) has increased manifold. The challenges in Natural language Query processing are interpreting the sentence correctly, removal of various ambiguity and mapping to the appropriate context. Natural language access problem is actually composed of two stages - Linguistic processing and Database processing. NLIDB techniques encompass a wide variety of approaches. The approaches include traditional methods such as Pattern Matching, Syntactic Parsing and Semantic Grammar to modern systems such as Intermediate Query Generation, Machine Learning and Ontologies. In this report, various approaches to build NLIDB systems have been analyzed and compared along with their advantages, disadvantages and application areas. Also, a natural language interface to a flight reservation system has been implemented comprising of flight and booking inquiry systems.

General Terms
Natural Language Processing (NLP), Natural Language Interface to Databases (NLIDB)

Keywords
Natural Language Interfaces, Mapping of SQL

1. INTRODUCTION
Interaction with a database based application requires either a GUI or expertise in database query language. A simple conversational system in natural language will be a great boon to the vast majority of application users. Natural language interfaces to databases (NLIDB) systems allow a user to communicate with the database directly by entering the query in the form of a natural language question. The NLIDB system maps the natural language query to the appropriate SQL by processing the information in the query and correlation with the system and domain metadata. NLIDB can be considered as a classical problem in the field of natural language processing. Although, the earliest research has started since the late sixties, NLIDB remains as an open research problem. The solution to the NLIDB problem can be obtained in two stages i.e. Linguistic processing and Database processing. In the first stage of linguistic processing, the natural language query is mapped and translated into the corresponding SQL query by appropriate mapping functions. In the second stage of database processing, database management and access is performed and the SQL query is executed by the system.

The main challenges associated with Natural language processing are Modifier attachment, Quantifier scoping, conjunction and disjunction, nominal compound problem, anaphora resolution, elliptical sentences, extra grammatical utterances, ambiguity and word sense disambiguation [1] [2].

There are generally five steps in NLP:

- Morphological Analysis: Break up the sentence into word tokens and identify the morphemes i.e. prefix, stem, suffix, punctuations etc. for each word.
- Syntactic Analysis: Construction of parse tree which groups the related words and shows the relationship between them as noun phrase, verb phrase, prepositional phrase etc.
- Semantic Analysis: Generate meaning for the whole sentence on basis of the results of the syntactic analysis which applies to individual words and groupings.
- Discourse integration: Interpret the sentence with reference to statements in the conversation preceding it. For ex. pronouns in the sentence.
- Pragmatic Analysis: Based on the above findings, try to again interpret the sentence as a whole to find its actual intended meaning or action. [2]

Some advantages of NLIDBs are [1]:

- User is not required to learn an artificial communication language.
- Better for some questions involving negation or quantification.
- Meaning of each question is complemented by the discourse context.

Some disadvantages of NLIDBs are [1]:

- Users find it difficult to understand what kinds of questions the NLIDB can or cannot cope with.
- It is often not clear whether a rejected question is outside the system's linguistic coverage, or whether it is outside the system's conceptual coverage.
- Users assume intelligence.
- It has been argued that natural language is not an appropriate medium for communicating with a computer system.
- NLIDBs usually require tedious and lengthy configuration phases before they can be used.

The purpose of this paper is to compare the different approaches of Natural Language Interfaces for Databases and study the various advantages and disadvantages. This can help to make a decision in selection of a suitable approach depending upon the required application system. The remainder of the paper is organized as follows. Section 2 gives an overview of selected academic, traditional and modern approaches. Section 3 details the analysis and comparison of various approaches. Section 4 uses one of the approaches to depict the implementation for a flight reservation database.

2. APPROACHES FOR NLIDBs

2.1 Earlier Approaches for NLIDB Systems

Since the end of 1960s, there has been large number of research works introducing the theories and implementations of NLIDBs. Androutsopoulos [1] mainly introduced four kinds of NLIDBs framework:

Pattern-Matching Systems: This framework is based on pattern matching and typical applications of this type of framework include SAVVY [2]. It uses rules to match patterns to the data.

For e.g. given a list of countries and their capitals

pattern: . . . “capital” . . .<country>

action : Report Capital of row where Country = <country>

pattern: . . . “capital” . . . “country”

action : Report Capital and Country of each row

Syntax Based Systems: In syntax-based systems, the user’s question is parsed i.e. analyzed syntactically using a syntactic parser. Syntactic NLIDB systems use a syntactic grammar to map the resulting parse tree into an expression in a database query language. One of the best-known syntax-based NLIDB systems is LUNAR [3].

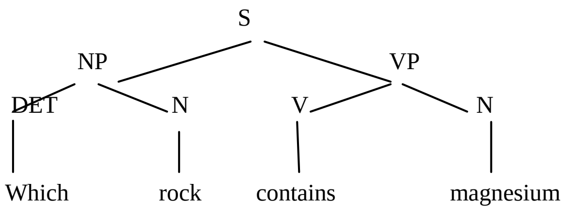


Figure 1: Parse Tree

S -> NP VP

NP -> Det N

Det -> “what” | “which”

N -> “rock” | “specimen” | “magnesium” | “radiation” | “light”

VP -> V N

V -> “contains” | “emits”.

Figure 2: An example of the syntactic grammar

Semantic Grammar Systems: The third one is based on semantic grammar. A semantic grammar NLIDB system is similar to a syntax-based NLIDB system. The user inputs will be translated into semantic tree by semantic analysis, then it will be translated into SQL and a typical example is PLANES [4] and LADDER [5].

S -> Specimen question | Spacecraft question

Specimen question -> Specimen Emits info | Specimen Contains info

Specimen -> “which rock” | “which specimen”

Emits info -> “emits” Radiation

Radiation -> “radiation” | “light”

Contains info -> “contains” Substance

Substance -> “magnesium” | “calcium”

Figure 3: An example of the semantic grammar

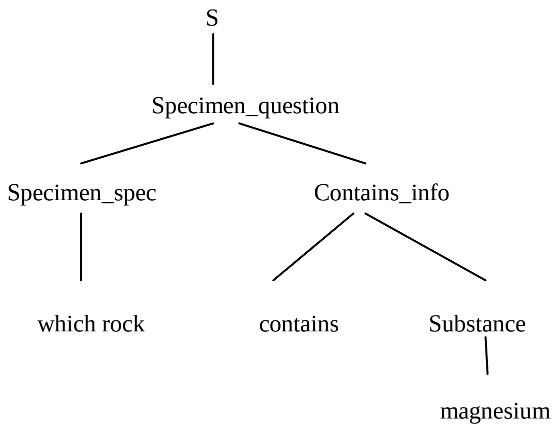


Figure 4: Semantic Tree

Compared to syntax analysis, semantic analysis uses semantic domain instead of grammar concept. The grammar’s categories (i.e. the non-leaf nodes that will appear in the parse tree) do not necessarily correspond to syntactic concepts.

Intermediate Representation Languages: Most current NLIDBs first transform the natural language question into an intermediate logical query, expressed in some internal meaning representation language. One typical example of this type is MASQUE/SQL [6].

2.2 NLIDB systems developed by universities

PRECISE: PRECISE is a system developed at the University of Washington by Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates (2004). The target database is in the form of a relational database using SQL as the query language [8]. The PRECISE work identifies a class of semantically tractable queries. It introduces the idea of semantically tractable sentences which are sentences that can be translated to a unique semantic interpretation by analyzing some lexicons and semantic constraints.

WASP: Word Alignment-based Semantic Parsing (WASP) developed at the University of Texas, uses predicate logic (Prolog) as the formal query language. WASP learns to build a semantic parser given a corpus of a set of natural language sentences annotated with their correct formal query languages [9].

NALIX: NALIX (Natural Language Interface for an XML Database), an NLIDB system developed at the University of Michigan uses extensible markup language (XML) database with Schema-Free XQuery as the database query language. NALIX implements a reversed-engineering technique by building the system from a query language toward the sentences [10] [11].

2.3 Modern Techniques

Intermediate Query Generation – A Modern perspective: Amisha Shingala et al [12] [13] [14] [15] have developed an intermediate query approach which performs lexical analysis, converting the lexeme to syntactic and semantic analysis and finally convert it into structured query language output. The semantic interpreter transforms the parse tree into intermediate query using semantic rules. The intermediate logic query generated by semantic interpreter, does not specify how to search the database to retrieve specific information. In order to retrieve, desired output required by user, the intermediate representation of query can be converted to some database query language which can be in form of any database system [12]. Various NLP tools such as Stanford Core NLP and Stanford Typed Dependencies are used for input NL query processing [23] [24].

Machine Learning Techniques: In the Machine Learning approach of Giordani and Moschitti [16] [17], the sentences are represented by parsing trees. The training pool consists of a pair of parsing trees: the first tree i.e. Natural Language Tree (NLT) for the sentence in natural language, the second tree i.e. Structured Query Tree (SQT) is for the sentence in SQL. There is a knowledge base to store the relationships between the nodes of NLT and SQT. In this approach, the mapping between natural language and database programming language is exploited at syntactic level and machine learning algorithms are applied to derive the shallow shared semantics. For this purpose, a dataset of relational pairs containing syntactic trees of questions and queries is designed and then encoded in Support Vector Machines by means of kernel functions. Kernel Methods refer to a large class of learning algorithms based on inner product vector spaces, among which Support Vector Machines (SVMs) are one of the most well-known algorithms [16] [17].

Ontology Based Approaches: The Ontology model is encouraged by advances in the semantic web research. It explicitly highlights the concepts and relations and not the words used to describe them. It is a simple and effective method of knowledge comprehension and concepts discovery. Ontology Concept Mapping (OCM) is intuitively derived from the way human beings reason and derive reasonable additional knowledge in order to formulate the question fully and attempt to answer it [18][19]. The relational database is converted into ontology. Concepts from the NL query and database ontology are represented and mapped. SPARQL Protocol and RDF Query Language (SPARQL) queries can be used to obtain information from Resource Description Framework (RDF) sources such as the semantic ontology for a Relational Database Management System (RDBMS). SPARQL is a structured query language that can query RDF sources.

Domain-specific semantic templates: Domain-specific semantic templates are used to implement a natural language interface to a virtual library by Niculae and co-researchers [20] [21]. Meta-knowledge of the database, namely the schema of the database is used as an additional resource to better interpret the question in a limited domain. The question is parsed through the Link Parser, and the ranked list of parse trees is matched with the semantic templates to find a match.

Lexico semantic patterns: Jung and Lee developed a multi-lingual question-answering system using lexico-semantic patterns to support multi-level grammars for query construction. Their system claimed to have high portability of languages, domains and databases. A semantic template is a pre-defined pattern and used in runtime to extract the semantic relationships between different objects identified from the input question [22].

3. ANALYSIS OF THE APPROACHES

Following is a table summarizing selected approaches for NLIDBs. The approaches chosen are unique and include both traditional and modern designs.

Table 1. Comparison of traditional NLIDB approaches and applicable systems

Technique	Advantages	Disadvantages	Applicable Systems
PATTERN MATCHING			
Pattern, Action	Simple and easy to be implemented	Does not perform for complicated queries. Pattern finding is tedious for large databases.	Works well for queries expressed as per pattern
SYNTAX BASED SYSTEMS			
Syntactic parse tree based on POS (Parts of Speech), Syntactic Grammar, Mapping Rules	Ability to express more complex patterns	It is difficult to devise mapping rules to transform parse tree to SQL	Application-specific database systems
SEMANTIC GRAMMAR			
Parse tree based on semantic concepts, Semantic Grammar, Mapping Rules	Semantic knowledge is included	New semantic grammar has to be written for different domain	Works well in restricted domains (i.e. small databases)
PRECISE			
Graph matching	High Precision. System detects ambiguous questions	Low recall	Works only for semantically tractable questions

Table 2. Comparison of modern NLIDB approaches and applicable systems

Technique	Advantages	Disadvantages	Applicable Systems
INTERMEDIATE QUERY APPROACH			
Domain-specific lexicon, Stanford Parser[23],	Enhancing disambiguation and better	Failure message cannot be deciphered by user. Semantic analysis and	Applications with exhaustive domain-specific

Stanford Typed Dependencies[24], Intermediate Query	context resolution. Linguistic component is database independent. NLIDB can be ported to a new DBMS.	post-processing required to convert to IQ.	lexicon.
MACHINE LEARNING			
Syntactic mapping of parse trees of NL and SQL, Semi-supervised learning approach	Syntax based approach, with less of semantic processing.	Cost of obtaining the corpus of NL/SQL pairs. Training has to be performed.	Applications having corpus readily available for training.
ONTOLOGY			
Construction of ontology, Feature extraction, Mapping Functions, RDF Triples, SPARQL Query	Supports knowledge comprehension and concepts discovery.	Ontology construction	Complicated queries. Resource-scarce languages with minimal linguistic processing activities.

4. IMPLEMENTATION

4.1 Flight Reservation System

A combination of Syntax Analysis and Intermediate Query approach is used for this system. Syntax Analysis performs syntactic processing and breaks the input sentence into its constituent parts and identifies the relations between the concepts. Intermediate Query approach allows to easily perform the mapping of concepts to an intermediate representation. The intermediate representation can be used even in case of database portability i.e. even if database is ported to another database.

The system is developed using Java, Spring framework, Hibernate ORM and MySQL database. The database has 2 tables for flights and bookings. Stanford Core NLP is used.

4.2 Steps

The algorithm is as follows:-

- The input sentence is tokenized to identify the various tokens.
- Stanford Parser is used to perform the parsing.
- Stanford Typed Dependencies is used for extraction of grammatical relations.
- The nouns and prepositions are extracted from the Tagger and Typed Dependencies.
- Semantic analysis is performed using nouns and prepositions.
- The nouns are mapped to the concepts of the database like tables and columns. For ex. Flight, Passenger.
- The prepositions and Typed Dependencies are used to map the governing and/or dependent tokens to the columns. For ex. From city, To city.
- The Natural Language query is mapped initially to an intermediate query by mapping rules for the various tables, columns and literals.
- After which it is then converted into SQL.
- Query is executed on the database and
- The results are displayed.

5. RESULTS

The software is tested by running various natural language queries. The results are evaluated. The output is checked with the actual records in the database and verified for correctness.

NL Query 1 –

Get the details of AIRINDIA flights from DEL

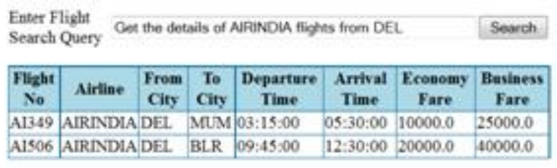
Table 3. Mapping of POS and Typed Dependencies for NLQ for Flight Details

Typed Dependencies	Tagged Parts Of Speech	Target Query

root(ROOT-0, get-1)
dobj(get-1, details-2)
nn(flights-5, AIRINDIA-4)
prep_of(details-2, flights-5)
prep_from(get-1, DEL-7)
	VB: get
	NNS: details
	IN: of
	NN: AIRINDIA
	NNS: flights
	IN: from
	DT: DEL
		SELECT F.*
		FROM FLIGHT_DETAILS F
		WHERE FROMCITY = 'DEL'
		AND AIRLINE = 'AIRINDIA'

Table 4. Mapping of nouns and prepositions to Database entities

Token	Database	Type
flights	FLIGHT_DETAILS	Table
DEL	FROMCITY	Column
AIRINDIA	AIRLINE	Column



Enter Flight Search Query: Get the details of AIRINDIA flights from DEL [Search]

Flight No	Airline	From City	To City	Departure Time	Arrival Time	Economy Fare	Business Fare
AI349	AIRINDIA	DEL	MUM	03:15:00	05:30:00	10000.0	25000.0
AI506	AIRINDIA	DEL	BLR	09:45:00	12:30:00	20000.0	40000.0

Figure 5: Flight Details Screen – Input Query and Output Rows

Table 5. Intermediate Query for NLQ for Flight Details

Translated Intermediate Query	Software Hibernate Query

SELECT: FLIGHT_DETAILS.*
FROM : FLIGHT_DETAILS
WHERE :
fromCity, DEL
airline, AIRINDIA
	select
	this_.FLIGHTNO as FLIGHTNO1_0_,
	this_.AIRLINE as AIRLINE1_0_,
	this_.ARRIVALTIME as ARRIVALT3_1_0_, this_.BUSINESSFARE as BUSINESS4_1_0_, this_.DEPARTURETIME as DEPARTUR5_1_0_, this_.ECONOMYFARE as ECONOMYF6_1_0_,
	this_.FROMCITY as FROMCITY1_0_,
	this_.TOCITY as TOCITY1_0_
	from FLIGHT_DETAILS this_
	where
	this_.FROMCITY=? and
	this_.AIRLINE=?

NL Query 2 –

Get the details of passengers travelling by AIRINDIA flights

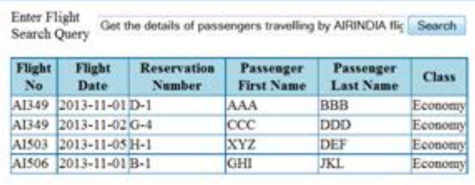
Table 6. Mapping of POS and Typed Dependencies for NLQ for Booking Details

Typed Dependencies	Tagged Parts Of Speech	Target Query

root(ROOT-0, get-1)
dobj(get-1, details-2)
prep_of(details-2, passengers-4)
partmod(passengers-4, traveling-5)
nn(flights-8, AIRINDIA-7)
agent(traveling-5, flights-8)
	VB: get
	NNS: details
	IN: of
	NNS: passengers
	VBG: travelling
	IN: by
	NN: AIRINDIA
	NNS: flights
		SELECT B.*
		FROM
		FLIGHT_DETAILS F
		JOIN
		BOOKING_DETAILS B
		WHERE AIRLINE = ‘AIRINDIA’

Table 7. Mapping of nouns and prepositions to Database entities

Token	Database	Type
flights	FLIGHT_DETAILS	Table
passengers	BOOKING_DETAILS	Table
AIRINDIA	AIRLINE	Column


Enter Flight Search Query
Get the details of passengers travelling by AIRINDIA flig
Search

Flight No	Flight Date	Reservation Number	Passenger First Name	Passenger Last Name	Class
AI349	2013-11-01	D-1	AAA	BBB	Economy
AI349	2013-11-02	G-4	CCC	DDD	Economy
AI503	2013-11-05	H-1	XYZ	DEF	Economy
AI506	2013-11-01	B-1	GHI	JKL	Economy

Figure 6: Booking Details Screen – Input Query and Output Rows

Table 8. Intermediate Query for NLQ for Booking Details

Translated Intermediate Query	Software Hibernate Query
SELECT : FLIGHTNO, FLIGHTDATE, RESERVATIONNO, CLIENTFNAME, CLIENTLNAME, FCLASS FROM : BOOKING_DETAILS FLIGHT_DETAILS WHERE : airline, AIRINDIA	select bookingdet1_.CLIENTFNAME as CLIENTFN2_0_, bookingdet1_.CLIENTLNAME as CLIENTLN3_0_, bookingdet1_.FCLASS as FCLASS0_, bookingdet1_.FLIGHTDATE as FLIGHTDATE0_, bookingdet1_.FLIGHTNO as FLIGHTNO0_, bookingdet1_.RESERVATIONNO as RESERVAT6_0_ from FLIGHT_DETAILS flight_det0_ inner join BOOKING_DETAILS bookingdet1_ on flight_det0_.FLIGHTNO= bookingdet1_.FLIGHTNO where airline='AIRINDIA'

6. CONCLUSION

NLIDB systems are very useful to non-technical people as a means of interaction with a database or an application (for e.g. flight inquiry, reservation systems etc.). Many different and varied approaches have been analyzed to tackle this challenging problem. While comparing earlier and modern approaches, it is evident that newer approaches take advantage of new technological breakthroughs such as Stanford Parses, Stanford Typed Dependencies, Word Net and Ontologies as well as 4GL programming languages and frameworks to save valuable time and effort and for ease of development. The NLIDB system usually comprises of syntactic and semantic analysis. Syntax analysis reduces to an NLP problem, while semantic analysis performs the mapping to a Database query. The semantic analysis approaches vary and are the main focus of NLIDB research. Pattern-matching approaches can be used in simple applications. Syntax-based systems require development of mapping rules and are suitable for application-specific database systems. Semantic grammar approaches do not give much importance to syntax and focus on the semantic meaning and are suitable for small, domain-specific applications. Machine learning approaches also focus on syntax and require a huge corpus of NL and SQL and also require training, learning and classification. For complicated queries, domain ontology approach seems to be the best. NLIDB problems are usually tackled by using logic-based approaches which involves morphological and syntactic processing followed by mapping between the query syntactic forms and SQL forms either through conversion to intermediate representation or directly. Combination of approaches depending on the particular situation at hand can also be used to tackle the NLIDB problem.

The future scope of NLIDBs can focus on achieving domain-independence of NLIDBs. Modifications to database such as updations and deletions have been given little importance and can be explored to achieve versatility. Also, extension to vernacular languages will spread the popularity of NLIDBs and will be of great use to the lay man.

7. REFERENCES

[1] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch, Natural Language Interfaces to Databases – AnIntroduction, Journal of Natural Language Engineering 1 Part 1 (1995), 29–81.

[2] T. Johnson. Natural Language Computing: The Commercial Applications. Ovum Ltd., London, 1985.

[3] Woods, W.A. "Progress in Natural Language Understanding: An Application to Lunar Geology", AFIPS Conf.Proc. 42, pp. 141-450, 1973

[4] D.L. Waltz. An English Language Question Answering System for a Large Relational Database. Communications of the ACM, 21(7):526–539, July 1978.

[5] G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a Natural Language Interface to Complex Data. ACM Transactions on Database Systems, 3(2):105–147, 1978.

[6] P. Auxerre. MASQUE Modular Answering System for Queries in English - Programmer's Manual. Technical Report AIAI/SR/11, Artificial Intelligence Applications Institute, University of Edinburgh, March 1986.

[7] Daniel Jurafsky and James H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2nd edition. Prentice-Hall, 2009.

[8] Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability, COLING (2004).

[9] Yuk Wah Wong, Learning for Semantic Parsing Using Statistical Machine TranslationTechniques, Technical Report UT-AI-05-323, University of Texas at Austin, Artificial Intelligence Lab, October 2005.

[10] Yunyao Li, Huahai Yang, and H.V. Jagadish, Nalix:an Interactive Natural Language Interface for Querying XML, SIGMOD (2005).

[11] Yunyao Li, Huahai Yang, and H.V. Jagadish, Constructing a Generic Natural Language Interface foran XML Database, EDBT (2006).

[12] Amisha Shingala, Paresh Virparia, "Enhancing the Relevance of Information Retrievalby Querying the Database in Natural form", International Conference on Intelligent Systems and Signal Processing ISSP, 2013.

[13] Amisha Shingala, Rinku Chavda & Paresh Virparia, "Natural Language Interface for Student Information System", Journal of Pureand Applied Sciences, Vol. 19:41-44, ISSN: 0975 – 2595, 2011.

[14] Amisha Shingala, Paresh Virparia,"Enriching Document Features for Effective Information Retrieval using Natural Language Query Interface", International Journal of IT, Engineering and Applied Science Research ISSN: 2319-4413, 2012.

[15] Anjali Ganesh Jivani, Amisha Shingala and Paresh Virparia, The Multi-Liaison Algorithm, International Journal of Advanced Computer Science and Applications,Vol 2, Issue ISSN 2156-5570 (Online), 2011

[16] Alessandra Giordani and Alessandro Moschitti, Semantic mapping between natural language questions and SQL queries via syntactic pairing, Proceedings of the 14th international conference on Applications of Natural Language to Information Systems NLDB 2009, 207-221

[17] Alessandra Giordani and Alessandro Moschitti, Generating SQL Queries Using Natural Language Syntactic Dependencies and Metadata, NLDB 2012, Lecture Notes in Computer Science LNCS 7337, 164-170

[18] Muchemi L, Popowich F. An Ontology-based Architecture for Natural Language Access to Relational Databases, SpringerLink Digital Library 2013.

[19] Khamis R., Shatnawi S., Toward enhanced Natural Language Processing to databases: Building a specific domain Ontology derived from database conceptual model, INFOS (2010), 1 – 8

[20] Niculae Stratica, Leila Kosseim, Bipin C. Desai, NLIDB Templates for Semantic Parsing. NLDB 2003, 235-241.

[21] Niculae Stratica, Bipin C. Desai, Schema-Based Natural Language Semantic Mapping. NLDB 2004, 103-113.

[22] Jung, H., Geunbae Lee, G., Multilingual Question Answering with High Portability on Relational Databases, IEICE transactions on information and systems, E86-D(2), 2003, 306-315.

[23] The Stanford Parser: A statistical parser, http://nlp.stanford.edu/software/lex-parser.shtml

[24] Marie-Catherine de Marneffe, Christopher D. Manning, "The Stanford typed dependencies manual" in Revised for Stanford Parser v1.6.2, February, 2010.