Enriching Data-Intensive Domain-Ontologies for Useful Transformation of Natural Language Queries

S. M. Abdullah Al-Mamun † **and Mohammad Moinul Hoque** ††

Department of Computer Science and Engineering
Ahsanullah University of Science and Technology, Dhaka, Bangladesh

Abstract
This paper presents a scheme for purposeful enhancement of domain knowledgebases which are supposed to be repositories of public information and also to be subject to frequent interaction by non-technical people through natural languages. In place of rigorous syntactic study of natural language queries, an acceptable alternative in the form of gradual simplification and recognition of useful semantic structures is proposed. Localization of matching procedures by isolating significant phrases and disambiguation of complicatedly ordered phrase components by heuristic rules are emphasized. Use of finite state machines for recognition and generation of data structures is fruitfully accommodated. Besides, techniques of inexact string matching for mapping query elements to database details have been incorporated to justify the usability of the enhancements.

Index Terms:
Domain Ontology, Natural Language Query, Semantic Structure Recognition, Approximate String Matching.

1. Introduction

Making information easily accessible to wide mass in our era of internet is very much obvious. Achievements of human endevour in varieties of domains and in various corners of the Glob are being published every now and then. It is within the virtual vicinity of many, but not within the reach of all of them. Representational differences with respect to conceptualization and languages play a vital role here, in our consideration. Enumerable useful information repositories of scientific and business interest stay out of reach of or seem very complicated to a lot of people. In this context age old efforts to share information and knowledge through automated natural language interfaces appear to remain in the fore front and open ended. Our basic assumptions regarding these issues are as follows:

a. A database system or a domain knowledgebase is better to be treated as a domain ontology with most of the characteristics and consequences.
b. General tools for enriching domain ontologies are to be sought after to provide easier implementation and access.
c. Adoption of the natural language basis in development and querying common purpose information repositories has a proven record of success.
d. Search for effective and efficient techniques for interpreting natural language queries is a real example of quest for accuracy, generality and simplicity.

A domain ontology, at least in its content, resembles a database system. Their resemblance is very much apparent with various components they include: individuals, classes, attributes, relations, function terms, events, restrictions, etc., [11, 19, 20]. As far as the ultimate goal to develop those is concerned, we believe that it is the useful information people are always after. Moreover, 'ontology' in principle better suits the purpose in terms of sharing.
On the other hand, general tools are essential to resolve linguistic and conceptual incompatibilities which inevitably arise as diverse groups are involved in sharing and translating varieties of information, [9, 11]. Accommodating advancements in formal languages, automata and text search seems very much in line with the need, [7, 22]. Research results in artificial intelligence and intelligent user interface, we think, also provide a good promise in this regard, [12, 21].
Further, developing natural language based shared terminology for effective interaction with information repositories is a quite old research trend, [29], that survives vigorously till our days, [3, 4]. Discovering and popularizing resources for overcoming language barriers can also be cited as great efforts here, [8]. Many works are just dedicated to make information more useful, [5, 14, 15].
Our last assumption is regarding efforts for appropriate ways of processing natural language queries to information repositories. Methodologies based on exhaustive tokenization and rigorous syntactic analysis have a deep root in the field, [13]. Systems assuming customization of grammatical parsers, like Microsoft English Query, are also available. Alternatively, systems emphasizing semantic analysis enjoy not less a popularity in the community, [24, 27].
We decided to stick more to pragmatic note of the problem exploring novel ways. Practically, it went down to applying useful semantic structures avoiding extreme

Manuscript received July 5, 2010
Manuscript revised July 20, 2010

syntactic analysis. We were inspired by a number of works in this respect, [10, 16, 23]. Some general references to relevant problems did also help us build up necessary confidence, [26, 28]. We also note that avoiding the potentially intractable problem of exhaustive string matching was very much in our mind. And we consulted general approaches, [18], as well as more specific ones, [17, 25], to be pragmatic through approximation. Moreover, we went for heuristics to help us. Some of the intermediate results of our efforts have been presented in international conferences recently, [1, 2].

Rest of this paper is organized in 8 sections followed by a list of references. Section 2 characterizes domain knowledgebases putting them before users interacting in natural languages. Section 3 has a detailed description of the significant phrases that we distinguish in a natural language query (NLQ), while section 4 illustrates how those phrases can be isolated in case of English queries. Section 5 is dedicated to the proposed enhancements of a database which are supposed to enable it to interpret NLQs. Section 6 sketches possible ways of mapping phrases isolated in NLQs to database details, and section 7 describes the fundamental components of the required inference engine. Experimental verification of our propositions is presented in section 8. Section 9 is a discussion that elaborates our concluding remarks and possible future study in the field.

2. Peculiarities of a Domain Knowledgebase and Natural Language Queries to it

Common components of a domain knowledgebase are the database and the inference engine. Usually, the database is that part of the domain knowledgebase which contains descriptions of numerous similar entities of the domain and also the descriptions of the specified relationships among various entities. So, essentially, an NLQ to a database must contain explicit or implicit narratives of some entities and/or entity relationships. The inference engine, on the other hand, usually contains predefined procedures for delivering decisions or hypothesizing over the existing database. Those predefined procedures may be enlisted and made available to the users in a manageable fashion for the users. We here are interested in some additional generic procedures that may be incorporated to the inference engine for effective recognition of general purpose NLQs addressing mainly the database content. In case an NLQ maps onto the functionality addressed to by an existing procedure, it may also be recognized so that the procedure can just be invoked in time. We will restrict the predefined procedures to those that correspond to most common and general types, like ‘Show the total number of students.’ Or ‘How many courses are available?’, and thus avoid considering those types of NLQs. Here and hereafter we will use query examples from a simple database domain, ‘Result Processing System’ (RPS), that assumes very common scenarios of course evaluation of undergrads.

For simplicity, we bind our presentation with the terminology of relational databases. NLQs, to which we direct our investigation, are just restricted types of English Interrogative and Imperative sentences. Such a sentence thus necessarily has to be rich with phrases, identifying objects of the domain, their attributes and attribute values. And those are the significant phrases which can be isolated indeed, [2]. If we consider the queries, ‘What is the address of the student with id 050104019?’ or ‘Display the address of the student with id 050104019.’, we find that important are the phrases [address], [student] and [id 050104019] only. And the order of the phrases found is also important. The header phrases here just fall short of having any importance in forming the answer expected by the user. The same fate awaits the punctuation marks at the end or the determiners used here and there. Potential delimiters and contents of some ‘significant phrases’ only remain for paying attention to.

3. Significant Phrases in Natural Language Queries

Discussion in the preceding section reveals that unlike common sentences, NLQs to databases are supposed to contain some phrases that are and only that are important for retrieval of information. We further postulate that instead of analyzing them, from the very beginning, in terms of grammatical phrases like noun phrases, verb phrases, adjective phrases, etc., it is more effective to analyze them in terms of phrases describing various aspects of objects, in database or ontological sense of the term. Our assumptions, in line with the ideas underlying formal query languages, also include the fact that there are a very few types of such phrases. We take it as a clear indication of an advantage over the methodologies taking into consideration the numerous grammatical phrases.

3.1. Attribute Phrase (AP)

The first and foremost of the phrases that are to be taken into account are the Attribute Phrases. An Attribute Phrase denotes an ‘attribute’ in a ‘relation’ or ‘table’. This type is most common, but obvious of the parameters used in a formal query language for information retrieval. ‘student ID number’ or ‘roll’ may simply denote the attribute ‘id’ in a possible ‘student’ relation. Attribute Phrases here represent those object attributes, values of which the user of the Natural Language Interface wants to see as the output of his or her query.

3.2. Attribute Value Phrase (AVP)

Next come Attribute Value Phrases, which specify the values associated to some attributes for addressing specific objects. The attributes are very much likely to be provided within the query, although they may remain implicit in some cases. Natural language flexibility may also allow those values and the corresponding attributes to be in a heavily messed up form. This leads to the need for identification of phrases that denote objects through values of the attributes. We eventually come to the concept of Attribute Value Phrases. Let us take the query, 'What is the CGPA of the student bearing identification number 050204003?'. The phrase 'identification number 050204003' is a simple AVP, which contains both an attribute portion and its value. While 'newly enrolled female' in 'Display the rolls and names of newly enrolled female students in the department of CSE.' is an example of a compound and complicated AVP. Moreover, negation descriptors are also usually placed in the AVP portion. So, the system needs to isolate the AVP portion including the negation descriptors which may have any of the forms like 'not', 'do not', 'have not', 'not in', 'was not' etc. The process of isolating AVPs thus requires special attention.

3.3. Object Identifying Phrase (OIP)

The concept of Object Identifying Phrases is to be introduced for coping up with those nuances of NLQs that essentially include pointers to specific relations of the database in terms of typical descriptors, which are common to multiple relations. Distinguishing 'male patient' and male doctor' in a Hospital Information System is a must. If an OIP is explicitly mentioned in the NLQ, it helps us to identify easily possible relations from where the data must be fetched. For example, in the NLQ, 'List the names of those students who are from CSE department', we see that 'those students' is a simple OIP which seems to have no big effect. But, during analysis of the NLQ for generating a response, this can provide enough hints in favor of the fact that information be fetched from the 'student relation'. More simple use of OIPs can be demonstrated in the query 'List the students of third year 1st semester'. Here no AP is specified, but records of students are to be fetched, although it is unlikely that an ordinary Data Dictionary has an option for that.

3.4. Aggregate Function Phrase (AFP)

Aggregate Function Phrases are common in queries as well. There is no doubt that we are going to have phrases containing words like average, total, sum, highest, lowest, best, first, last, etc., and we refer to them as AFPs. It also comes out very interesting to isolate AFPs and recognize them in combination with APs, AVPs and OIPs present in a query. For example, in the NLQ 'What is the average CGPA of the students of 3rd year 1st semester?' the AFP marking word 'average' needs to be taken care of in association with the AP 'CGPA'. But in the NLQ 'List the students who scored highest grade in CSE101.' we see that in 'scored highest grade' the AFP is mixed up with an AVP and it refers to the OIP 'students'. So, we have all the reasons to consider seriously these phrases mentioned in various places of NLQs.

For illustration of the various significant phrases described above let's consider the query, 'What are the names, resident addresses and CGPAs of the students bearing id numbers 050204001 and 050204009?'. The structure of this query can be presented as a syntactic rule like the one below.

$$\text{Query} \rightarrow HW_1\ AP_1\ ,\ AP_2 \text{ and } AP_3 \text{ of } OIP_1\ oip\_del_1\ AVP_1 \text{ and } AVP_2$$

HW_1 (Head Word 1) here stands for 'What', AP_1 for 'names', AP_2 for 'resident addresses', AP_3 for 'CGPAs' OIP_1 for 'students', oip_del_1 (OIP delimiter 1) for 'bearing' (a delimiter from a set of delimiters that may possibly appear after an OIP), AVP_1 for 'id numbers 050204001' and AVP_2 for '050204009'. We further add that HW_1 (Head Word 1) could take the value 'List', 'Display', 'Show', etc. with no noteworthy change in semantics.

Let's have another illustrative example that contains an aggregate function phrase, 'Show the average grade points of 1st year 1st semester CSE students in CSE103'. The corresponding rule will have the following form.

$$\text{Query} \rightarrow HW_2\ AFP_1 \text{ of } AVP_1 \text{ in } AVP_2$$

AFP_1 here stands for 'average grade points', and AVP_1 for '1st year 1st semester CSE students', which is a compound and quite complicated attribute value phrase.

4. Isolating Significant Phrases in NLQs

The boundaries of useful phrases in NLQs that we are set to isolate need to be marked first. It means that we have to localize possible APs, AVPs, OIPs, etc. From our discussion above we reach to the point that there are some words or phrases in an NLQ that do not play an important role in interpreting it and responding to it. Thus we come to the idea of preprocessing or normalizing an NLQ first, which enable us to have only a headword in a more general form, some words as delimiters of potential significant phrases that we refer to as 'Non-Delimiter Phrases' (NDPs) and the NDPs, of course. The normalized NLQ is then passed to an algorithm, [2], the output of which is a

phrase structure (PS). A PS is a sequence of pairs containing NDPs and NDP classes. An important feature of the algorithm is that it uses a set of Finite Automata, referred to as DFAs, to recognize the delimiter sequence in the normalized NLQ. Another important feature is that a set of Moore Machines (MMs) is used to generate the PS consisting of the NDP-NDP class pairs.

It happens that normalizing an NLQ for filtering out words that do not play any role in formation of an executable query simplifies the overall process to a great extent. It is in line with the idea of disregarding 'words not semantically important', [13]. We have marked them as 'insignificant'. The words 'is' and 'the' may be marked insignificant in the NLQ 'What is the CGPA of the 3rd year student Abdul Karim?'. The sequence is supposed to undergo further refinement that frees it from words that may come into conflict with predefined phrase delimiters. While 'of' is a very important phrase delimiter in our objects of investigation, this very 'of' in the phrase 'department of CSE' will be a misleading delimiter.

The normalization process includes a number of steps, most important of which are head word modification, removal of semantically not important words, distinguishing and generalizing delimiters and standardization of mathematical and temporal expressions. A simple head word 'display' can replace 'What', 'I would like to see' or 'Please tell me', while expression '<=' may replace any of 'not over', 'not greater than', 'did not get over', etc. Similarly, changing 'current semester' with 'semester spring 2010' may help a lot.

After the normalization process the query is ready for probation of its delimiter sequence using a DFA from the predefined DFA set. The query, 'What are the names, resident addresses and CGPAs of the students bearing id numbers 050204001 and 050204009?' is normalized to 'display names and resident addresses and cgpas of students with id numbers 050204001 and 050204009'. The delimiter sequence, 'display and and of bearing and' is accepted by the following DFA (Fig. 4.1).

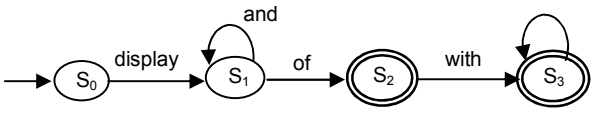

Fig. 4.1. DFA that recognizes delimiter sequence DS_1

We note here that the DFA shown in Fig. 4.1 can equally successfully recognize numerous instances of delimiter sequences of queries of different size and content. Once the delimiter sequence is accepted, a rule is generated to narrate the content of the given query highlighting the isolated significant phrases. Here it takes the following form:

$$\text{Query} \rightarrow HW_1\ AP_1 \text{ and } AP_2 \text{ and } AP_3 \text{ of } OIP_1 \text{ oip\_del}_1\ AVP_1 \text{ and } AVP_2$$

The rule in its turn invokes an MM (Moore Machine) from the predefined set of MMs, and the MM has the transition diagram like the one in the following figure (Fig. 4.2).

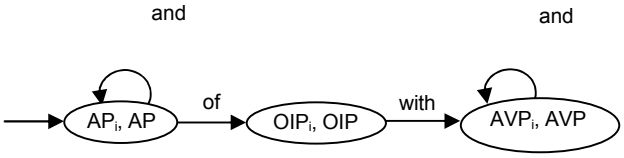

Fig. 4.2: Moore Machine that returns Phrase Structure PS_1

Some typical queries, including the one analyzed above, and their Phrase Structure Output (PSO) generated by MMs are shown in the Table 4.1.

5. Database Enhancement for Recognition of Significant Phrases

Capability of a domain ontology to respond properly to NLQs essentially requires additional information about the domain. This additional information is to encounter ambiguity & multiplicity of natural language elements. Our efforts here are directed to finding general forms of enhancement of the database embedded in the ontology. As we are supposed to have isolated phrases in the NLQs that are to be mapped to database contents, we concentrate on possible forms of a number of tables that have the potential to capture the representational diversity of those phrases.

First of all, for every specific database domain, we need an **AP Table** that contains the attribute phrases in the justifiable spectrum of their possible synonyms associated to the attributes from different relations. The synonyms may appear interchangeably in the NLQs for the corresponding attributes under that domain. Values in the AP Table are domain specific. For example, the synonym 'id number' referring to the attribute 'StudentId' in RPS may be related to a different attribute named 'EmployeeId' in a different database domain. Some entries of a possible AP Table for the RPS are shown in Table 5.1.

In our Enhanced data dictionary we also need a **Typical Value Table (TVT)** as shown in Table 5.2. It contains typical values and their synonyms in association with the attributes and relations they refer to. This table is essential for that, some NLQ are very much likely to have sketched an AVP using only a commonly understood value of the relevant attribute, and not mentioning the attribute itself. The typical value 'male' always refers to the attribute that represents 'Gender' and the typical value 'CSE' or 'EEE' refers to an attribute that may be named as 'DeptName'. When the system needs to recognize an AVP, the TVT is

consulted first. If a match is found in the TVT, partial matching of the possible AP portion of the AVP becomes redundant.

Table 4.1 Examples of isolated significant phrases

NLQ	PSO
What are the names, resident addresses and CGPAs of the students bearing id numbers 050204001 and 050204009?	((names, AP), (resident addresses, AP), (cgpas, AP), (students, OIP), (id numbers 050204001, AVP), (050204009, AVP))
List the female students of CSE department who got A+ in CSE101.	((students, OIP), (female, AVP), (cse department, AVP), (a+, AVP), (cse101, AVP))
Find the average marks of the 2nd year 1st semester students of the department. of EEE.	((average marks, AFP), (students, OIP), (2nd year, AVP), (1st semester, AVP), (department eee , AVP))
Please show the names and ids of students whose names has 'Hasan' as a part.	((names, AP), (ids, AP), (students, OIP), (names hasan, AVP))
Can you please display the total number of courses offered by the CSE department?	((total number courses offered, AFP), (cse department, AVP))

TABLE 5.1. Sample entries to an AP table for RPS

Attribute Phrase	Attribute	Relation
id number	StudentId	Student
roll number	StudentId	Student
course id	CourseNo	Course
credit hour	Credit	Course
course number	CourseNo	Course
student number	StudentId	Student

TABLE 5.2. Sample entries to the Typical Value Table (TVT) for RPS

Typical Value	Attribute	Relation
Male	Gender	Student
female	Gender	Student
cse	DeptName	Department
eee	DeptName	Department
computer	DeptName	Department

To be sensible, we only consider those attributes for the TVT whose sets of possible values are of a very limited cardinality. There might be some other attributes of relations on which a user of the database system places queries containing AVPs without AP portion. For example, in the query, 'What is the address of the student Abdul Karim?', the AVP 'Abdul Karim' has no AP part mentioned. But the set of names of the students has a very high cardinality, and names thus cannot be stored as typical values in the TVT. Similar situation arises if there is just a student id number of the form 'std2010s543' in place of the name. To handle this sort of situations, we have used quite successfully a **set of useful patterns**, **common instances** and **templates** for only those attributes on which frequent NLQs may be thrown. In case of necessity, a partial match can turn out to be a very good support here.

We also need an **OIP Table** to map possible OIPs to database relations. This, comparatively small, table contains possible synonyms of the identifiers of the objects that each of the relations contains. An example of such an OIP table is shown in Table 5.3.

And to map AFPs we need an **AFP Terms Table** that contains possible synonyms of terms characteristic to phrases aggregating on different types of data. The terms are supposed to be accompanied by other phrases like APs or OIPs, which need to be recognized separately. We use here a kind of standardization by using replacement of the terms with generally interpretable keywords. A typical AFP Mapping table is shown in Table 5.4

TABLE 5.3. A sample OIP table for RPS

OIP	Relation
students	Student
persons	Student
courses	Course
subjects	Course

6. Mapping Phrases to Database Details

The useful transformation of an NLQ includes an obvious step, tracking down its important components to the domain knowledgebase. So, a Phrase Structure (PS) returned by the process of isolating significant phrases has got to be mapped to the concrete database details of the domain ontology. The extension of the databases described above is used for the purpose. The relevant algorithm, [1], is in fact an exhaustive matching procedure to associate database details to the previously marked phrases in the NLQ. It takes as input the PS, that is, the sequence of NDP-NDP class pairs, and returns the PS associated to the database object descriptors. The output has in it concrete references to object attributes, attribute values and generic functional terms in place of NDP class descriptors, and has been described above as PSO (Phrase Structure Output).

The idea underlying the procedure is a very simple one, although the implementation requires a delicate treatment involving inexact matching, heuristics etc. We need to associate various NDPs classified as APs, AVPs, etc. with the database details. To confirm, for example, whether an NDP classified as AP matches with any relevant entry of the AP table sufficiently, we compare it to the entries of the AP table. The lengthiest string that is common in the NDP and each of the entries (attribute synonyms) of the AP table is derived. If its length crosses a defined threshold value, the particular match is considered. The threshold value is set to a level that ensures maximum accuracy of the matching.

In case of an NDP classified as AVP or AFP, things stand pretty difficult to map. There the order and range of the AP portion and value portion or the aggregation terms also become important.

TABLE 5.4. Typical entries to an AFP Term Table

AFP Term	Keyword
highest	max
top most	max
lowest	min
total number	count
total	sum

Those parts may be heavily messed up in the phrase. This has a vital effect not only in efficiency, but also in effectiveness of the procedure trying to achieve the desired mapping. Studying the suffixes and prefixes in collaboration with the domain specific knowledge about linguistic, representational and stylistic peculiarities of expressions comes to the rescue in most of the times. The assumptions like 'usually the attribute portion in an AVP precedes the value portion', 'the attribute portion of an AVP may be implicitly referred to in the subsequent AVP' or 'the aggregation term usually prefixes an attribute portion in an AFP' turns out to set things very much pragmatically as far as the disambiguation of AVPs and AFPs are concerned.

For illustration we here show the transformation of two typical queries shown in Table 4.1.

Query: What are the names, resident addresses and CGPAs of the students bearing id numbers 050204001 and 050204009?

PSO: ((names, AP), (resident addresses, AP), (cgpas, AP), (students, OIP), (id numbers 050204001, AVP), (050204009, AVP))

Recognized PSO: ((AP, Name, Student), (AP, Address, Student), (AP, CGPA, Student), (OIP, Student), (AVP, StudentId, =, "050204001", Student), (AVP, StudentId, =, 050204009, Student))

Query: Can you please display the total number of courses offered by the CSE department?

PSO: ((total number courses offered, AFP), (cse department, AVP))

Recognized PSO: ((AFP, count, CourseId, Course), (AVP, DeptName, CSE, Course))

7. Essentials of the Inference Engine

Major stages of the intended transformation of NLQs and the inherent complexity of the steps to be performed necessitate the important components of the inference engine. Our approach assumes in a series the processes already mentioned as normalization, demarcation of phrase boundaries, recognition of delimiter sequence, classification of separated phrases and mapping classified phrases to the domain database.

As we have already discussed, preprocessing or normalizing an NLQ prepare it to a great extent for effective demarcation of boundaries of phrases and efficient processing further. This frees the NLQ from unnecessary symbols and words, and reduces the size by replacing lengthy descriptions with short and more formal ones. The admissibility of the normalization is bound to possibility of matching words and word sequences to string entries of strictly small tables. Interactive input arrangement make things tolerable by allowing entry-time processing, including even spell checking. Anyway, fast string matching becomes a demand, [18, 25].

Maintenance of a set of DFAs for recognition of the delimiter sequences of the NLQs needs to be mentioned next. We have observed with appreciation that a few tens

of DFAs, not so complicated in nature, capture almost the whole spectrum of numerous NLQs that can be thought of about a general purpose domain. A DFA can be represented by its transition function, which obviously has a simple tabular form, [7]. So, the set of tables representing the set of DFAs becomes a charming addition to the domain knowledgebase. The set can be easily enhanced, and its elements can also be modified if there is a need.

Next comes the set of Moore Machines for classifying the phrases in the structure curved out of the NLQs in the process of targeted transformation. As we know, Moore Machines are also finite state machines each state of one of which may be associated with a classifier. So, once again we have a set of tables representing transition functions of MMs. They resemble DFAs in number and maintenance, while complement them greatly towards the common goal of interpreting NLQs.

After all, procedures for efficient string matching during various stages, especially the mapping of isolated phrases from the NLQs to database details, occupy the valuable core of the inference engine. Various approximation methods and useful heuristics need to be discovered and incorporated to this core. We have found that implementation of some five simple heuristic rules, [1], help in disambiguation of messed up AVPs almost to completeness. In cases of disambiguation of AFPs and OIPs even smaller number of such heuristic rules is very much apparent. On the other hand, suitable inexact or approximate string matching procedures are in a high demand for the core of the engine, [17, 22]. The adoption of the method of Longest Common Subsequence along with the whole collection of candidate-limiting techniques enables us, [1], to make potentially intractable matching procedures very much competitive computationally.

In addition to the above, supervised learning in various forms like upgrading enhanced data dictionary, sets of finite state machines etc. deserves special mentioning. The procedures are negligibly simple. Once the supervisor is convinced he is supposed to add or delete an entry in a particular table. Although, in case of a necessity a new finite state machine may be required to be designed, and it is a common song just from another opera. The process of introducing it to the system is again very simple. It is just including to the existing set a new table.

8. Empirical Justification

Experimental setup for justifying our ontology enrichment scheme was designed based on three easily comprehensible database systems, namely, RPS (Result Processing System for course evaluation of students of a university), ATIS (Air Travel Information System of a travel agency) and EIMS (Employee Information Management System of an organization). The quantitative outline of the data set we used is shown in Table 8.1. It has been noted that although we tried to be quite extensive and exhaustive in finding possible relations, attributes and their synonyms in each of the systems, the number of entries in the TVT, OIP table and AFP terms table remained consistently and comprehensibly small.

We were guided by the fact that our approach differed from other known approaches substantially, and thus we would concentrate on showing usefulness of the transformation we offer for NLQs and also demonstrating effectiveness and efficiency of the process.

TABLE 8.1. Experimental Data Sets Cardinalities

System	Relations	Distinct Attributes	Attribute Synonyms	Typical Values	Entries in OIP Table	AFP Terms
RPS	8	28	112	18	25	16
ATIS	6	19	123	10	24	14
EIMS	14	36	155	26	48	20

As far as the usefulness of the transformation is concerned, the recognized Phrase Structure Output (PSO) that we ultimately get can be easily shown to have distinctive relationships with, for example, expressions in SQL (Structured Query Language) like query languages for information retrieval from a data repository. We here show the two typical examples of transformation cited in section 6 with possible SQL expressions corresponding to them.

Query: What are the names, resident addresses and CGPAs of the students bearing id numbers 050204001 and 050204009?

Recognized PSO: ((AP, Name, Student), (AP, Address, Student), (AP, CGPA, Student), (OIP, Student), (AVP, StudentId, =, “050204001”, Student); (AVP, StudentId, =, 050204009, Student))

Possible SQL expression: select Name, Address, CGPA from Student where (StudentID = "050204001") or (StudentID = "050204009")

Query: Can you please display the total number of courses offered by the CSE department?

Recognized PSO: ((AFP, count, CourseId, Course), (AVP, DeptName, CSE, Course))

Possible SQL expression: Select count(CourseId) from Course where DeptName = "CSE"

To check the effectiveness we used batches of sample NLQs for each of the three database systems keeping in mind diversity of linguistic terms and complexity of formulation. Available benchmark data sets were also consulted in this regard. Representative phrase detection with PSO recognition outcome for RPS is shown in Table 8.2. Observations show that the failure cases are mostly related to very uncommon styles of query formulation. Effectiveness of the system is thus quite explicit.

Success in detection of AVPs in EIMS, for example, using heuristic rules discussed earlier is shown in the Figure 8.1.

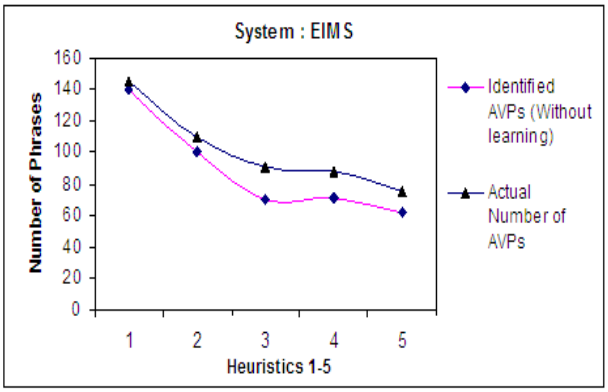

Fig. 8.1. AVP detection in EIMS (before training cycles)

Improvements done involving supervised learning (training) in average error reduction in ATIS, for example, is shown in Figure 8.2.

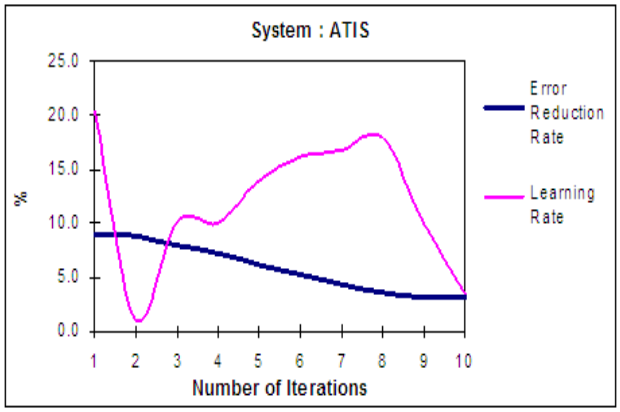

Fig. 8.2. Error reduction and learning rates in ATIS

For assessing efficiency we depended on average response time in returning recognized phrase structures. If the normalization is considered an interactive entry-time process, then the response time, even for a slow personal computer, is not at all noticeable; otherwise it is just a small fraction of a second. We show the result, taking normalized queries, some of them replicated, and executing in a 2.60 GHz personal computer, for all the three systems, in Table 8.3.

We find that the time required for returning a recognized phrase structure is very small in comparison to time required for entering a query or formulating and executing an SQL like query.

Table 8.2. Sample phrase detection and phrase structure recognition in RPS

Query Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Actual APs	1	2	3	1	2	1	3	2	2	2	1	2	2	4	5
Identified APs	1	2	3	1	2	1	3	2	1	2	1	2	2	4	5
Actual AVPs	3	2	5	2	2	1	2	6	2	3	1	2	1	2	3
Identified AVPs	3	2	4	2	2	1	2	6	2	3	1	1	1	2	3
Recognized PSO	+	+	-	+	+	+	+	+	-	+	+	-	+	+	+

Table 8.3. Average time required for returning recognized phrase structures in RPS

System	Total Number of NLQs	Total Time for Phrase Structure Recognition	Average Time Per Phrase Structure
RPS	500	48.20 sec	0.0964 sec
ATIS	400	34.24 sec	0.0856 sec
EIMS	400	24.68 sec	0.0617 sec

8. Discussion

A methodology for enrichment of domain knowledgebases is the object taken for illustration in the paper. A domain knowledgebase has been supposed to resemble a database system that has all components of a domain ontology. As information repositories for common people are being considered, so retrieval techniques appropriate for commoners follow. A system in such an environment needs to be natural language oriented where domain knowledge is easy to interpret and share in bigger domains. This is how reference to domain knowledgebase and ontology came to the scene.

Useful transformation of queries in English posed to a domain knowledgebase has been the objective of the investigation. It is evident that a relatively small number of basic structures involving significant phrases are there in natural language queries. This was the stimulation for going to search for some new solutions to an old problem.

Further, we believe that extensive use of natural language grammars may be avoided. If it is so, then we need to handle less ambiguity. And portability across domains as well as natural languages seems more easily achievable by dealing with phrase structures in sentences avoiding grammatical nuances. Use of finite state machines for effectiveness and efficiency appears very appropriate for the purpose.

It is also evident that we can avoid exhaustive token matching by isolating the significant phrases first. Token matching becomes intractable in wider range. Nontraditional techniques like partial matching, heuristics in the form of studying prefixes and suffixes, etc. help to a great extent in this regard. May it be the case of finding an attribute value phrase or mapping an aggregation term, heuristics regarding the structure, content and size of the target phrases play an important role in inexact matching.

We also find that a kind of generality may be achieved by accommodating techniques that find phrase structures from the types of sentences other than those we investigated. Thus the methodology can be tried as a tool to shed an insight into understanding sentences in general. We may consider the representation of the world to be a collection of semantic networks or domain ontology of various things. So, beside efficiently constructing intelligent interfaces to databases for mass use, we may think of pure linguistic research.

The languages which allow deliberate separation of significant phrases are most suitable as targets. English and Universal Natural Language (UNL), [8], are alright in this respect. To this end, we think that many a common purpose databases may be designed as individual domain ontology in UNL, which is very much English like. In that case, all people in the community of languages associated with UNL can have the benefit.

One of the most important observations is that such a system can be easily trained in the development phase to perform remarkably well. Failure cases can just be gathered and new entries for the tables of the extended database may be looked for. Finite state machines are really portable across domains. If necessary a new one can just be added. Still, it may not be possible to cover all possible variations of the sentence patterns. Additional measures have to be thought of.

Our plan of future works in this field includes, beside other relevant matters, investigation of transportability of our phrase structure analysis techniques across various types of sentences, domains and languages. Moreover, speech recognition can also be added to the proposed one so that it can process Natural Language Queries by listening from the user. However, such systems are supposed to be of very complex nature, and some of the linguistic challenges will have to be addressed first. We are interested to carry on with our research and add efforts to developing natural language components for domain knowledgebases.

References

[1] M.M. Hoque, S.M.A. Al-Mamun. Recognition of Attribute Value Phrases in Natural Language Queries to Databases. Proceedings of 3rd International Conference on Data Management (Innovations and Advances in Data Management), pp. 1-14, Ghaziabad, India, March 2010.

[2] M.M. Hoque, M.S. Mahbub, S.M.A. Al-Mamun. Isolating significant phrases in common natural language queries to databases. Proceedings of 11th International Conference on Computer and Information Technology, pp.554-559, Khulna, Bangladesh, December 2008.

[3] P. Cimiano, P. Haase, J. Heizzmann. Porting Natural Language Interfaces Between Domains: an experimental user study with the ORAKEL system. Proceedings of 12th International Conference on Intelligent User Interfaces, pp.180-189, Honolulu, Hawaii, USA, 2007.

[4] Y. Li, I. Chaudhuri, H. Yang, S. Singh, H. V. Jagadish. DaNaLIX: a domain-adaptive natural language interface for querying XML. Proceedings of the 2007 ACM SIGMOD international conference on management of data, Beijing, China, pp. 1165-1168.

[5] O. Küçüktunç, U. Güdükbay, Ö. Ulusoy. A Natural Language-Based Interface for Querying a Video Database. IEEE Multimedia, Vol. 14, pp. 83-89, Jan-Mar 2007.

[6] E. Kaufmann, A. Bernstein, R. Zunstein. Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. Proceedings of 5th International Semantic Web Conference, pp.980—981, Springer, November 2006.

[7] J.E. Hopcroft, R. Motwani, J.D. Ullmann. Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 3rd edition, 2006.

[8] H. Uchida, M. Zhu, T.C.D. Senta. Universal Networking Language, UNDL Foundation, 2nd edition, Geneva, 2005.

[9] A. Gomez-Perez, M. Fernandez-Lopez, O. Corcho. Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer, 2004.
[10] N. Stratica, L. Kosseim, B.C. Desai. NLIDB Templates for Semantic Parsing. Proceedings of 8th International Conference on Applications of Natural Language to Information Systems, pp. 235-241, Berg, Germany, June 2003.
[11] B. Smith. Ontology. Blackwell Guide to the Philosophy of Computing and Information, pp. 155–166, Oxford: Blackwell, 2003.
[12] S.J. Russell, P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2nd edition, 2003.
[13] A.M. Popescu, O. Etzioni, H. Kautz. Towards a Theory of Natural Language Interfaces to Databases. Proceedings of the 8th international conference on Intelligent user interfaces, Miami, Florida, USA, pp. 149-157, 2003.
[14] M. Samsonova, A. Pisarev, M. Blagov. Processing of natural language queries to a relational database. Bioinformatics, Vol. 19, pp i241-i249, Oxford University Press, January 2003.
[15] M. Dittenbach, D. Merkl, H. Berger. A Natural Language Query Interface for Tourism Information. Proceedings of 10th International Conference on Information Technologies in Tourism, pp.152-162, Helsinki, Finland, January 2003.
[16] H.H. Mengg, K.C. Siu. Semiautomatic Acquisition of Semantic Structures for Understanding Domain-Specific Natural Language Queries. IEEE Transactions on Knowledge and Data Engineering, pp. 172-181, Jan–Feb 2002.
[17] G. Navarro. A guided tour to approximate string matching. ACM Computing Surveys, Volume 33, pp. 31–88, 2001.
[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, S. Clifford. Introduction to Algorithms (Chapter 32: String Matching), pp. 909-993, MIT Press and McGraw-Hill, 2nd edition, 2001.
[19] A. Silberschatz, H. F. Korth, S. Sudarshan. Database System Concepts, McGraw-Hill, 3rd edition, 1997.
[20] M. Uschold, M. Gruninger. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, Vol.11, pp. 93—136, 1996.
[21] I. Androutsopoulos, G.D. Ritchey, P. Thanisch. Natural Language Interface to Databases – An Introduction. Journal of Natural Language Engineering, Vol. 1, pp. 29-81, 1995.
[22] R. Baeza-Yates, G. Gonnet. A new approach to text searching. Comm. ACM, 35, pp. 74-82, 1992.
[23] H.W. Beck, S. Navathe. Integrating natural language, query processing and semantic data models. Digest of Papers, 35th IEEE Computer Society International Conference (Compcon '90), pp. 553-543, 1990.
[24] D.G. Shin. Semantics modeling issues for processing natural language database queries. Proceedings of the 1990 ACM annual conference on Cooperation, pp. 8-14, Washington D.C., United States, 1990.
[25] G.M. Landau, U. Vishkin. Fast String Matching with k-difference. Journal of Computer and System Sciences. Volume 37, pp. 63-78, August 1988.
[26] H. Ishikawa, Y. Izumida, T. Yoshino, T. Hoshiai, A. Makinouchi. KID: Designing a knowledge-based natural language interface. IEEE Expert, Vol. 2, pp.57-71, 1987.
[27] J.M. Janas. The semantics-based natural language interface to relational databases. Cooperative interfaces to information systems. Springer-Verlag, New York, 1986.
[28] F.J. Damerau. Problems and some solutions in customization of natural language database front ends. ACM Transactions on Information Systems (TOIS), vol.3, pp.165- 184, April 1985.
[29] S.J. Kaplan. Designing a Portable Natural Language Database Query System. ACM Transactions on Database Systems (TODS), vol.9, pp. 1-19, March 1984.