Precisiating Natural Language for a Question Answering System

Marcus THINT
Intelligent Systems Research Unit, British Telecom (BT Americas)
Boca Raton, FL 33498, USA

and

M. M. Sufyan BEG
Department of Computer Engineering, Jamia Millia (A Central University)
New Delhi 110 025, India

and

Zengchang QIN
Berkeley Initiative in Soft Computing, EECS Department, UC Berkeley
Berkeley, CA 94720, USA

ABSTRACT

We report on an application of Precisiated Natural Language (PNL) concepts and protoformal deduction, which are integral to Computational Theory of Perception, and Computing with Words, as developed by Lotfi Zadeh. A semi-automated precisiation process is part of an information extraction module for a question answering system. Simplified natural language statements (containing a single verb phrase) are first subjected to part-of-speech tagging to identify the verb phrase, subject phrase and the object phrase, if any. If the verb phrase is an "is-form" (covering all modalities and tenses of the "to be" verb) we dwell into further analysis of this sentence being one of the various PNL protoforms, such as *X isr A*, *Y isr (X+B)*, *QAs are Bs*, and *f(X) is A*. Via protoformal deduction, more precise answers can be computed for a subset of a knowledge corpus (e.g. critical or frequently-asked topics) where fuzzy set definitions of vague terms are provided. For sentences without an "is-form" verb phrase, supplemental analyses detect causal facts, if-then rules, procedures, or simple propositions, and phrase-based deduction is subsequently applied where possible. Analyses are extended to query-type classification which is used to refine answer ratings.

Keywords: Precisiating Natural Language, Protoformal Deduction, Question Answering.

1. INTRODUCTION

Concepts of Precisiated Natural Language (PNL) [1] and the process of language precisiation provide a new approach towards semantic interpretation of natural language. Traditionally, linguists depend on syntax and grammatical structure of language in order to analyze and understand the conveyed message. Zadeh supports perception-based approach to describe propositions, which can be subsequently represented and manipulated based on a generalized theory of uncertainty [2]. Consider a conceptual space where language provides a practical means of communicating so that an audience can understand the concepts and messages expressed by the speakers. Words usually have multiple meanings (or senses) and even a single word can map to multiple sites in the conceptual space; hence a 'bag of words' by itself is not useful for semantic interpretation. Thus, we often use a sentence, i.e. ordered set of words according to a particular grammar, to convey a message more precisely. Sometimes, using only one sentence is not enough to describe a complicated concept/message, so we may require a paragraph, an article, or even a book to convey a complete description. More words generally provide more constraints in describing a concept. In the conceptual space, these constraints reduce the number of concept sites (alternate interpretations) to help the audience understand the conveyed message without confusion. Language precisiation is such a process of adding constraints in order to clearly describe a complicated or ambiguous concept in natural language.

In the generalized constraint expression *X isr R*, *X* is the constrained variable, *R* is the constraining relation, and *r* is a discrete valued modal variable. The "is" in *isr* is simply its natural meaning – the conjugated verb "to be". Thus, the expression *X isu R* means *X* is usually *R*, (i.e. *r=u*) and other defined modalities include: probabilistic (*r=p*); veristic (*r=v*); random set (*r=rs*); fuzzy graph (*r=fg*); bimodal (*r=bm*); Pawlak set (*r=ps*), and possibilistic (*r* is blank).

2. APPLICATION CONTEXT

We apply the precisiation concepts and a method for semi-automated detection of PNL protoforms in the context of a problem domain that require advanced analysis of natural language statements, and where a 'bag of words' representation of a text document is insufficient.

While receiving a relevant list of documents from a search engine is a tremendous value when searching a vast resource such as the Web, more direct answers are expected from domain restricted information portals such as corporate intranets and customer (self-) service websites. For example, if a customer asks "Where can I find the reset-button on product Alpha", a directly relevant fact such as "The reset-button is located on the rear panel, inside the hole adjacent to the power adaptor connector" should be returned, rather than pointing to a set of assembly-, user-, and service-manuals about product Alpha, which the customer must download and further search. Thus, the problem addressed is a design of a restricted-domain automated Question Answering (QA) system, whose knowledge sources are unstructured natural language text (e.g. product or service description).

The high-level system block diagram is shown in Figure 1 below. Part-of-speech (POS) tagging [3] and PNL fact-type analyses are performed in the front-end modules as part of the preprocessing and information extraction phase.

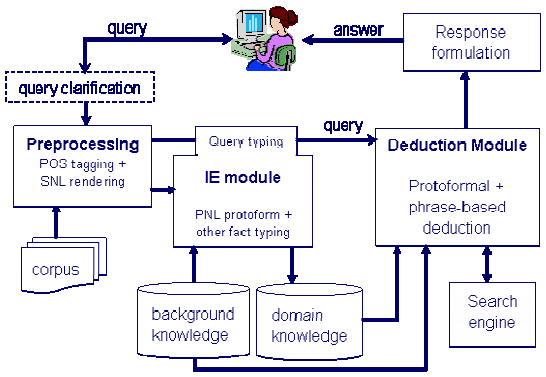

Fig. 1. Block-diagram of PNL-enhanced QA system

An important design goal is a re-usable, domain-independent design that minimizes effort for an application developer to analyze and prepare domain knowledge. Accordingly, design approaches that require hand-crafted, domain-dependent ontologies are avoided, including manual mapping of domain knowledge chunks to pre-defined ontology nodes. However, due to current limitations of automated Natural Language Processing (NLP) technology, we cannot eliminate the need for human assistance in QA application development. Thus, we reserve application developer's assistance mainly for clarification of semantics in domain-specific concepts and provision of limited background knowledge. Additional description about the overall system, and phrase-based deduction may be found in [4] and [5] respectively. Previous QA research have been based solely on classical natural language processing technology [6], and some recent publications have discussed leveraging the Web as a knowledge resource [7]-[11].

3. PRECISIATION PROCESS

Since QA problems require deeper reasoning (e.g. deductive reasoning in addition to keyword matching of search engines) system knowledge is represented as various fact-types, including PNL protoforms and other (causal, rules (if-then), procedures, or plain 'facts').

3.1 Background

In Zadeh's formulation of PNL theory and the associated examples provided [2], it is generally expected that a human will interpret the semantics of text phrases and convert their representations into GCL (Generalized Constraint Language). Such manual process of precisiation would yield the best quality result (as with any task involving natural language understanding), but we investigate the feasibility of a semi-automated precisiation process and its application towards problems seeking automated solutions in the real-world.

Three key issues provide the setting and support for this research (i) timing – availability of theories on Computing with Words [12] and PNL, together with stable search engine technologies provide the opportunity to investigate novel and higher-risk approaches with value-added benefits; (ii) nested processing design – exploitation of the above by triggering phrase-based deductive reasoning when fact-types do not fit PNL protoforms, and further integration with standard search engine to provide known baseline performance; and (iii) tolerance – maturation of search technology and user expectations when dealing with natural language-based information sources have led to better acceptance of imperfect solutions, as long as they are useful improvements over alternative means.

3.2 PNL Protoform Detection

As an initial effort to automate the recognition of PNL protoforms, input to the precisiation (IE) module is limited to simple sentences containing a single verb phrase – denoted as simplified natural language (SNL). Each sentence from the corpus is first subjected to POS tagging to identify the verb phrase, subject phrase and the object phrase (if any). Verb phrase comprises the string between the first and last occurrence of the tags that denote various verb forms (e.g. starting with "/VB") including English modal words {*can, could, may, might, will, would, shall, should, must, ought to*} and additional qualifiers {*usually, often, seldom, probably, likely, partially, possibly*}. Reverse ordering of the modal words and qualifiers are also considered, combined with all (1st, 2nd, 3rd) person forms and tenses of the "to be" verb. A verb phrase matching one of the key phrase combinations above are denoted as an "is-form" verb phrase, and after it is extracted, the subject and object phrases are identified as the phrases preceding and following the verb phrase respectively. It is possible that for intransitive verb phrases, there is no object phrase.

Simple compound sentences (two or more SNL joined by *and*, *or*, *but*, *however*, *so*, *while*, *yet*) can be converted into SNL by a program that examines POS tags, and reconstructs multiple simple sentences. In the present system design, complex sentence structures that cannot be precisiated or rendered into SNL are processed as "other" fact types by the deduction module.

If the verb phrase is an "is-form" the IE module proceeds to further analyze a sentence as being one of the various PNL protoforms, such as *X is A*, *Y is (X+B)*, *QA's are B*, *Q(A's and B's) are C*, and *f(X) is A* [1], [2].

X is A: Facts such as "Average price of crude oil in 2005 was around 50 dollars a barrel" is in the form

X_1 *is* A_1 where

X_1 = *average price of crude oil in 2005*, and

A_1 = *around 50 dollars a barrel*,

separated by the verb phrase "was."

Y is (X+B): Similar to the above, the fact "Average price of crude oil in 2006 was moderately higher than average price of crude oil in 2005" is identified as the form *Y is X + B* where

Y = average price of crude oil in 2006,

X = average price of crude oil in 2005, and

B = moderately higher

but additional processing is required to reach this conclusion. The above sentence must first be recognized as an

X_2 *is* A_2 where

X_2 = *average price of crude oil in 2006*, and

A_2 = *moderately higher than average price of crude oil in 2005*.

Subsequently, the keyword "than" in A_2 triggers further analysis to subdivide A_2 into

B = *moderately higher*, and

A'_2 = *average price of crude oil in 2005*.

Only if A'_2 is equivalent to a previously known X, it can be concluded that *Y is X + B* protoform has been found. In this example, A'_2 is found to be exactly the same as X_1 but an exact string match is not necessary, as approximate matches are supported by the concept matching module (Section 5.1). Note that if phrase *B* had been "a little less" rather than "moderately higher", the protoform notation *Y is X + B* is still valid, as long as the fuzzy set definition of "a little less" produces a subtraction effect. Further implications and use of this result is discussed in Section 4.

***QA's are B*:** Propositions of the form "most service cancellations are price related" is in the form Q_1A*'s are B* where

Q_1 = *most*

A = service cancellations

B = price related.

This sentence is also first recognized as an *X is A* form and components are re-labeled if a "quantity term" initiates the *X* part. Quantity term *Q* includes numbers, text description of numbers, and fuzzy quantifiers such as {*some, few, none, many, most, a little, a lot*}. Note that text description of numbers can be recognized as some combination of a limited keywords including *one* to *twenty*, *thirty* to *ninety* (by tens), *hundred*, *thousand*, *million*, *billion*, etc.

Similarly, a sequel proposition of the form "some price related service cancellations are permanent" is in the form Q_2*(A and B)'s are C's* where

Q_2 = *some*

(A and B)'s = price related service cancellations

C = permanent.

More precisely, *(A and B)'s = (B and A)'s* where the expression contains both *A* and *B* phrases.

***f(X) is A*:** Automated extraction of the *f(X)* concept is an especially challenging problem – e.g. recognition that the semantics of "product-X is expensive" is actually "cost of product-X is expensive" or the semantics of "the room is cold" is actually "temperature of the room is cold", etc. As PNL protoforms, the latter results can be expressed as *cost(product-X) is expensive*, or *temperature(room) is cold*, where *cost()* and *temperature()* are the abstracted functions.

Whether such abstract concepts can be learned by machine learning techniques and contrived examples pose an interesting theoretical exercise. However, it is relatively easy for humans to identify certain *f(X)* concepts, and we initially rely on this human-assisted approach which can subsequently be automated.

A limited set of hashtables are created wherein the metric terms for *f(X)* concepts are keyed with descriptions of function *f*, such that, during analysis, if a metric such as "expensive" is found in the value array, its key term "cost" is returned as function *f*. Note that only certain concepts (e.g. *speed, age, cost, weight, size, temperature, scent, texture, time, weather*, etc.) are linked with metric terms that map to the function *f* with high probability are pre-stored as system core knowledge. For instance, terms such as *fast, slow, quick, rapid* invariably refer to a "speed" concept of an entity or a process. Similarly, the terms *small, large, big, huge, tiny, mini* reliably refer to the "size" of something. It is true that "large talk" or "living large" imply a grand or ostentatious "sense" and is different from a "large box", but in most cases, the associated *f(X)* keys do apply.

Note that some metric terms are less reliable indicators of a corresponding function compared to its peers. For example *hot, cold, warm, cool, freezing, boiling* are commonly good descriptors of temperature concept, but "hot", and "cool" are also frequently used in other contexts that do not imply temperature (e.g. "hot car", or "cool design!"). In such cases, only select subset of

the metrics terms are used.

This approach does not provide perfect coverage or performance (very few rules applied to natural language do) but due to the nested processing design, a missed detection of *f(X)* concept may result in a missed opportunity to provide a more precise response, but a default response based on concept matching or standard search is available. *f(X)* detection is invoked only if an *X is A* form is first recognized as in other cases.

3.3 Negation

It is known that one or more "not" words in a sentence fragment can alter its meaning and deduction results accordingly. Thus far, according to the sentence segmentation approach described above, it is in fact possible that a sentence identified as *X is A* is in the form *X is not A*, where "not" has been embedded as part of *A*. Thus we explicitly check for "not" in the original sentence and include the findings as part of the data structure passed to the deduction engine. Presently, the deduction engine is informed whether a single "not" is detected or if more than one occurrence was found. Consecutive multiple "not" occurrences are readily resolved, but other cases require deeper analysis of the sentence structure and has not been completed.

3.4 Non-PNL Fact Types

Although not strictly a precisiation process (into PNL protoforms) general fact-type analyses is performed in the IE module, and is summarized here for completeness. For sentences without an "is-form" verb phrase, supplemental analyses are performed to detect *causal* facts, *if-then* rules, *procedures*, sentence *fragments*, or simply 'fact' (default). Causal facts (e.g. detected via *due to*, *lead to*, *cause*, *because* and *since* keywords between two phrase segments) and if-then rules are useful towards answering why-type questions, since portions of the sentence can be identified as the "cause" and "effect" fragments. Procedures, detected by list elements (e.g. numbering, bullets/dashes) plus imperative phrases are useful towards answering how-type questions. Sentence fragments are ignored unless included as part of a procedure. Facts containing quantity descriptors (see *QA's are B* section) are also marked.

4. PROTOFORMAL DEDUCTION

A key motivation for the precisiation efforts is to leverage the benefits of protoformal deduction and augment system knowledge. Without using PNL and protoform detection concepts, automated deduction with natural language statements require complex NLP and computational linguistic analyses. Protoformal deduction illustrates that for a subset of natural language forms, automated reasoning with natural language statements can be simplified to simpler string substitution and known fuzzy arithmetic computations.

As explained in Section 3.2, sentences like "Average price of crude oil in 2005 was around 50 dollars a barrel" maps to *X is A* protoform, and "Average price of crude oil in 2006 was moderately higher than average price of crude oil in 2005" maps to a *Y is X + B* protoform. Protoformal deduction then leads to a conclusion that *Y is A + B*, which yields the text expression "Average price of crude oil in 2006 is around 50 dollars a barrel PLUS moderately higher."

If fuzzy sets for "around 50 dollars" and "moderately higher" are not defined, the string expression as shown above will be returned as an answer (which can be interpreted and understood by a human user). If, however, the term 'around 50 dollars' is defined by a triangular fuzzy set centered on 50 and the term 'moderately higher' is defined by a percentage (e.g. 25%) on the range of fuzzy set 'around 50 percent', the protoformal deduction performs the following fuzzy addition:

TriFuzzy(50, 5) + 50 × 25% = TriFuzzy(62.5, 5)

where TriFuzzy(*a*, *b*) represents a triangular fuzzy set whose center is at *a* and the width is *b*. The final answer is returned as "about 62.5" where 62.5 is the defuzzified value of the composite fuzzy set. Note that in practice, application developer can specify that for all expressions {"around *N*", "about *N*", "approximately *N*"} where *N* is a real number, its fuzzy set is defined by

TriFuzzy(*N*, α*N*)

where α is a pre-specified constant (e.g. α=0.1), such that fuzzy set definitions can be automatically created during preprocessing stage whenever matching string patterns are encountered. Similarly, application developer can pre-specify fuzzy set definitions for {"a little higher", "moderately higher", "much higher", "a little lower", "moderately lower", "much lower"} semi-automatically, rather than manually defining each one.

QA systems built only on standard search technology cannot provide the computed response "about 62.5" based on the original two sentences. Although every concept in a corpus would not be pre-defined with fuzzy set representation, for a particular subset of knowledge and key terms where they are defined, PNL-based computing offers value-added performance.

Following the *Q_1A's are B* and *Q_2(A and)'s are C's* example from Section 3.2 with the phrases "most service cancellations are price related" and "some price related service cancellations are permanent" respectively, it can be deduced that *(Q_1 x Q_2) A's are (B and C)'s* , or

(*most* x *some*) *service cancellations* are (*price related* and *permanent*).

Thus if, for example, centroid of the *most* fuzzy set is 0.65 (65%) and centroid of the *some* fuzzy set is 0.3 (30%), and a question "What fraction of all service cancellations are price related and permanent?" the following answer

would be produced via protoformal deduction:

About 0.2 of all service cancellations are price related and permanent.

Consider another example in applying the *f(X)* concept combined with deductive reasoning. As explained above some *f(X)* protoforms (e.g. cost(a flat screen monitor)) can be identified from phrases like "A flat screen monitor is expensive", and the deduction module supports associated computations. Given the following facts:

A mouse is cheap.
A computer monitor is two hundred US dollars.
A keyboard is fifty US dollars.

If asked "What is the cost of a computer monitor plus a keyboard?" the QA system outputs:

This is a 'compound' type question (e.g. f(X) plus f(Y)).
Reasoned from sentence:
"*A computer monitor is two hundred US dollars*."
AND
"*A keyboard is fifty US dollars*."
We can obtain:
---
cost(A computer monitor) PLUS cost(A keyboard) IS (two hundred US dollars)+(fifty US dollars)

A traditional (non-PNL enhanced) QA system could not synthesize the final output sentence shown above. If we ask for "the cost for a mouse plus a keyboard," the answer will be *fifty US dollars PLUS cheap*. Once again, the defuzzified value "about X" will be returned if *cheap* is predefined by a fuzzy set and the value of X can be computed; else, only the linguistic answers as shown above is returned.

Other protoformal deductions are computed in similar manner. For example, given: *Q1A's are B's*, *Q2(A's and B's) are C's*, the result after protoformal deduction is that *Q3A's are (B's and C's),* where $Q3 = Q1 \bullet Q2$, and $\bullet$ is a product in fuzzy arithmetic. In general, the computational rules as shown in Table I apply to the protoforms. Each of the forms in the left column of Table I can be also extended with respect to different modal constraints (probabilistic, usuality, bimodal interpolation, fuzzy graph interpolation, etc.).

TABLE I
PROTOFORMAL DEDUCTION PRINCIPLES

Given:	*Then:*
X is A, (X, Y)is B	*Y is C*, where $\mu C(v) = \max_u (\mu A(u) . \mu B(u, v))$
Q1A's are B's and *Q2(A&B)'s are C's*	*Q3A's are (B&C)'s* where $(Q3 = Q1 \bullet Q2)$
X is A	*g(X) is B*, where $\mu B(v) = \sup_u (\mu A(u)), v = g(u)$
f(X) is A	*g(X) is B*, where $\mu B(v) = \sup_u (\mu B(f(u))), v = g(u)$

5. OTHER KEY ISSUES

5.1 Concept Matching

During protoform detection, it is necessary to find equivalent phrases (e.g. between *X* and *A* arrays to conclude *Y is X + B*). Phrase matching is a common capability required for many applications that involve descriptions expressed in natural language, and simple string matching is insufficient. As natural language allows different ways to express the same concepts, phrases must be compared at the concept level, rather than comparing strings (or stemmed versions). Expressions "price of X" and "cost of X" match conceptually, but not during string comparison. A synonym generator must be employed to find a match between these phrases, but that is also not trivial, since many words have multiple "senses" and it is unclear which sense should be used for comparison. The next level of complication arises when comparing phrases like "cost of calls to Zagreb, Croatia" and "cost of phone calls to Zagreb". After stemming and stopword removal, the two phrases become {cost call Zagreb Croatia} and {cost phone call Zagreb}. Concluding their match requires knowledge that in this context, "call" and "phone call" are equivalent, and "Zagreb Croatia" and "Zagreb" are also equivalent, which cannot be solved solely via synonym look-up.

Due to such challenges, we have not achieved a complete solution, but have developed a routine that returns a fuzzy metric depending on different type/degree of string matches: exact, partial match at beginning, partial match at end, synonym match using Wordnet [13], and percentage of keyword match. For the above (Zagreb) example, this concept match function would return 0.75 similarity value, which is sufficient to propagate processing to further stages.

5.2 Query Processing

Query analysis is a complex topic in itself, which usually includes query clarification through a dialog with a user to ascertain intentions and context. At this juncture, however, our focus has been merely on generating the appropriate answers assuming the query is relevant and clearly stated. Currently, query analysis is limited to extending techniques used by the preprocessor to detect query types.

Query types are detected by spotting key phrase patterns in the query. When reliably detected, this knowledge helps refine the ranking of most relevant facts, but it is not critical, since the subject matter of the query is the primary criteria for selecting relevant subset of knowledge.

Phrases starting with *why* or *what is the reason/cause/rationale*, *due to what*, *because of what* predict why-type (qtWHY) queries with good reliability. Phrases like *how do I, how do we*, *what are the*

procedures/methods/steps, *explain how*, frequently start how-type (qtHOW) questions, and queries starting with *where is*, or containing *whereabouts*, *locate* or *location* are indicative of where-type (qtWHERE) questions. Usually, when-type (qtWHEN) queries start with *when is*, *when do*, forms or contain *before/after/during* keywords. Quantity-type (qtQUANTITY) queries frequently start with *how much*, *how many* or *what quantity*. If none of the above are detected, query defaults to what-type (qtWHAT).

A query like "Will product-X be available in September" is actually a "When will product-X be available" question, and many other 'alternate phrasing' could cause wrong typing via simple key phrase spotting approach. But obvious and common expressions are detected and used to refine answer ranking. Often, multiple facts will be found relevant to a query, but if a query is type qtWHY, then ranking of causal facts and if-then rules are incremented. If a query is type qtHOW, ranking of procedure facts are incremented, and likewise for qtQUANTITY and facts containing quantities. This type of secondary refinement helps move most appropriate answers to the top of the list.

6. SUMMARY AND DISCUSSION

We discussed some novel concepts from PNL-based computing and the benefits offered by protoformal deduction towards a QA application. We also note some limitations of PNL-based approach as follows. In the present paper and Zadeh's theories, precisiation has been discussed only for propositions in α *is* β form, where α may vary from *X* to *QAs* to *f(X)* etc. Nevertheless, propositions in the form "something is another-thing" are only a small subset of natural language expressions, and thus provide limited utility in the overall problem space. Hence, we employ a nested processing design in which non-PNL protoforms are processed via phrase-based deduction module [4] and [5].

It is also interesting to note that even a simple statement like "Mary is young" can be precisiated in multiple ways: Age(Mary) is young, and Person(young) is Mary. Thus, some background or context knowledge is required for full clarification. Also, we mentioned that to derive the full benefit of PNL-based computing, fuzzy terms in the natural language statements must be associated with pre-defined fuzzy sets. Since human-assistance in this effort is likely to be reserved only for an important subset of a corpus, it would be helpful to automate the fuzzy set definition process as much as possible.

A key challenge, however, is that perceptions of fuzzy concepts vary in general. For example, the concept of "tall" depends on whether the observer is a short person or an NBA basketball player, and whether the domain is about human height or buildings in NY city. If the personalized tuning aspect is made optional, most concepts can be pre-defined with a three to five fuzzy set "family". Methods described in Section 4 can be extended to enable fast definition of an entire family of fuzzy sets for a concept, by simply specifying the number, universe of discourse, and fuzzy set label names.

In view of the complexities involved, we designed and investigated a semi-automated solution, which appears to remain as the practical route in the near term. Both the research prototype and underlying theories/technologies remain under development, and planned extensions include: improvements to concept matching module, investigation of alternate knowledge representation formats like RDF and OWL [14], and standardized testing using TREC data sources as the system matures.

7. REFERENCES

[1] L. A. Zadeh, "Precisiated natural language," *AI Magazine*, 25(3), 2004, pp. 74-91.

[2] L. A. Zadeh, "Toward a generalized theory of uncertainty (GTU) - an outline,"in *Information Sciences*, 172, 2005, pp. 1-40.

[3] K. Toutanova, D. Klein, C. Manning, and Y. Singer, "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network,". *in Proc. of HLT-NAACL 2003*, pp. 252-259..

[4] M. Thint, M. M. S. Beg, Z. Qin, "PNL-enhanced Restricted Domain Question Answering System," *IEEE International Conference on Fuzzy Systems,* London, UK, July 2007, submitted for publication .

[5] Z. Qin, M. Thint, M. M. S. Beg, "Deduction Engine Design for PNL-based Question Answering System," *World Congress of the International Fuzzy Systems Association* , 2007, to be published.

[6] E. Vorhees, L. Buckland (Eds), *Proceedings of the Eleventh Text Retrieval Conference*, NIST Special Publication: SP 500-251, 2002.

[7] E. Brill, S. Dumais, and M. Banko, "An analysis of the askMSR question-answering system," in *Proc. of 2002 Conf. on Empirical Methods in Natural Language Processing*, 2002, pp..257-264.

[8] O. Tsur, M. de Rijke, and K. Sima'an, "Biographer: biography questions as a restricted domain question answering task," in *Proc. of the ACL 2004 Workshop on Question Answering in Restricted Domains,* July 21-26, 2004.

[9] F. Benamara, "Cooperative question answering in restricted domains: the WEBCOOP experiment," in *Proc. of the ACL 2004 Workshop on Question Answering in Restricted Domains*, July 21-26, 2004.

[10] D. Azari, E. Horvitz, S. Dumais, and E. Brill, "Web-based question answering: a decision making perspective," in *Proc. of the Conf. on Uncertainty and Artificial Intelligence, 2003,* pp. 11-19.

[11] H. Chung et. al., "A practical QA system in restricted domains," in *Proc. of the ACL 2004 Workshop on Question Answering in RestrictedDomains,* July 21-26, 2004.

[12] L. A. Zadeh, "From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions," in *Int. J. Appl. Math. Comput. Sci*., 12(3), 2001, pp. 307-324.

[13] G. Miller, "Wordnet: a lexical database," *Communications of the ACM*, 38(11), 1995, pp. 39-41.

[14] Available: http://www.w3.org/TR/owl-ref