Web Queries in Protoform and RDF semantic

Chris Tseng[1, 2] Patrick Ng[1]

[1] CS department, San Jose State University, One Washington Square, San Jose, CA 95192, USA
[2] tseng@cs.sjsu.edu (corresponding author), Tel: 1-408-924-7255, Fax: 1-717-754-6112

Abstract

For a search engine to truly understand the web query, one needs to preprocess the query semantic before submitting to the world-wide-web to retrieve relevant web pages. We study two forms to abstract and precisiate web queries. RDF and Protoform are both useful for this purpose and share a similar structure in representing the query semantic in an analytical form. Protoform is found to be useful for representing queries that may involve fuzzy semantic. Once the query is in Protoform, it is ready for web search using fuzzy semantic.

Keywords: Protoform. RDF. Semantic Search. Fuzzy Logic. Search Engine.

1. Introduction

The current way of performing search on internet is being done in a crisp way. Popular search engines simply perform keyword searches blindly without understanding of the query statement to any extent. This paper presents the elements of paving the way to build a Fuzzy Semantic Search (FSS) engine that is capable to understand human natural language search statement.

Machine cannot understand natural language (NL) so there is a need to transform NL to a form that machine can interpret and compute. The idea of Precisiated Natural Language (PNL) [2,3] introduced by Prof. Zadeh provides a tool in bridging the gap between NL and machine readable language. In order to achieve this goal, there are intermediate steps. First, NL has to be transformed into a fuzzy-logic based language names as Generalized Constraint Language (GCL) [1,2,3]. Then, GCL is further converted into Protoform Language (PFL) [1,2,3] for computation. Once PFL formatted input is received by FSS, FSS can interpret the PFL structure and perform searches separately on normal keyword-based search engine and show ranked summary results [4].

We begin by presenting an overview on GCL, PFL and Resource Description Framework (RDF) in section 2. In section 3, we provide examples to illustrate the steps involved in processing a NL search statement in FSS. The tool named MontyLingua [4] that can help in the NL processing will also be described. In section 4, we will conclude with future research, related issues and limitations.

2. RDF, GCL, PFL

The basic idea of GCL is representing information by showing how a variable is being constraint by a certain kind of limitation. The basic syntax of a GCL representation is:
X isr R (2.1)
X is the constrained variable, R is the constraining relation and r is the way that R constrains X.
r can take on one of the following forms: possibilistic (r=blank), probabilistic (r=p), veristic (r=v), usuality (r=u), fuzzy graph (r=fg), etc.

PFL is an abstraction of the GCL. The basic structure is
A(B) is C. (2.2)

RDF is a standard framework in semantic web to represent metadata about web resources. It can be represented either as an XML document, triplets, or directed graph. It has a triplet structure of subject, predicate and object. The object and subject are related through a predicate relation. There is a lot of in common between RDF and PFL. Both can be used to represent a statement in semantic form that is well abstracted and precisiated. Once the statement is in RDF form, a lot of XML

related tools and analysis can be applied. However, one major advantage of representing in PTL is that the predicate relation can be fuzzy and is considered more applicable to a wider variety of real-life applications.

We would like to use the real estate search domain to illustrate our formats in our examples. Some comparison of how natural language statements can be transformed into PTL and RDF formats will be presented below.

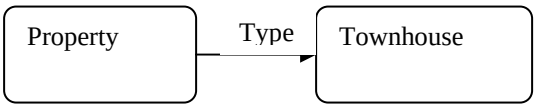
Example 2.1: Equality constraint (r = "="); X = R

Natural Language (NL):
"***The property is a townhouse***"

General Constraint Language (GCL):
X ise R, where X is Type(Property) and R is townhouse
Type(Property) is townhouse

PFL:
A(B) is C
A – Type
B - Property
C – Townhouse (a crisp keyword)

RDF:
Subject: Property
Predicate: Type
Object: Townhouse



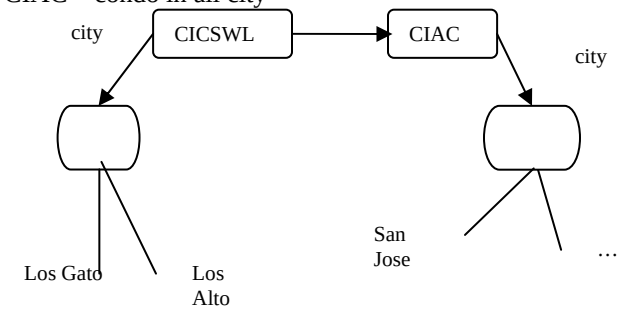
Example 2.2: Subset hood constraint (r="⊂"); X⊂ R

NL:
Show me condo in city name starts with "los"

GCL:
X is R, where X is name(city(condo)) and R is start_with_los
name(city(condo)) is start_w/_los

PFL:
A (B (C)) is D
A – name
B – city
C – condo
D – starts with "los"

RDF:
CICSWL – condo in city starts with "los"
CIAC – condo in all city



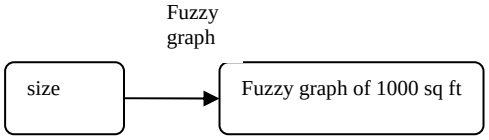
Example 2.3: Fuzzy graph constraint (r = fg); X is a function and R is its Fuzzy graph.

NL:
Show me townhouse around 1000 sq ft.

GCL:
X isfg R, where X is size(townhouse) and R is "around 1000 sq ft"

PFL:
A (B) is C
A – size
B – townhouse
C – around 1000 sq ft

PDF:
Size (townhouse) isfg (around 1000 sq ft)



Example 2.4: Usuality constraint (r = u); usually X is R

NL:
"House with swimming pool is expensive"

GCL:
X isu R, where X is "house with swimming pool" and R is expensive

Prob { with swimming pool (house) is expensive } is usual

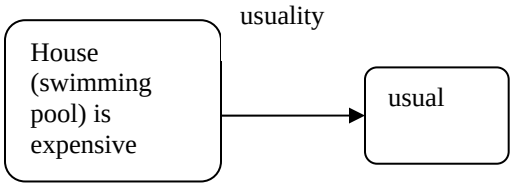
Count ((price(house w/ swimming pool) is expensive) /

(price (house w/ swimming pool) is median)) is most

PFL:

Prob { A (B) is C } is D
A – with swimming pool
B – house
C – expensive
D – usual

PDF:



The above examples show how a natural language statement can be transformed into GCL format and then to PFL or RDF. In example 2.1, the user wish to start searching on a real estate that is a townhouse. First, we want to find out the constrained variable, in this case, it is property. With the domain of property, it can have many attributes, like the age, type, size of the property, etc. Then, from the statement, next we want to identify the constraint, in this case, it is townhouse. As we know townhouse is considered as a type of property, so the whole constrained variable is considered as Type(Property). And the constraining relationship is equality in this case. By performing abstraction on the GCL format, we can come up with the PFL. As RDF shares a similar structure as PFL, a RDF diagram can then be constructed. Once NL got translated into PFL/RDF, we can perform search computation according to the structure. We now summarize the flow of the process of semantic search with PTL and FSS as follows. This is general flow of search process applied to the equality, inequality, subset hood and fuzzy graph illustrated in Examples 2.1-2.4.

Search Steps:

1. using B as the keyword to search
2. combine A & C, the constraint is applied to the outermost variable
3. perform Fuzzy Linguistic Search (FLS)/ Fuzzy Numeric Search (FNS) based on A & C

Please refer to Figure 2.1 for the search flow diagram.

Examples 2.2 to 2.4 are provided as references for other cases of constraint relationships.

3. From NL to PFL

In order to build an end-to-end FSS engine that will be easy to use and retrieve effective result, it is important for search engine to be able to take in natural language in plain English and understand it to some extents. There is a need to transform natural language statement to PFL format so that it can be processed with subsequent modules in the FSS engine. Focusing on a specific domain will help to simplify this effort and produce more sensible results. Therefore, in our case, we will focus on retrieving real estate information. There is a natural language processing tool created by Hugo Liu from MIT Media Lab called MontyLingua [5] that can facilitate the natural language processing task. MontyLingua is capable to extract phrases and subject/verb/object triplets from plain English statement. This is obviously a suitable pre-processing tool that can be used to prepare web query in a form ready for the PTL and RDF described in this paper. Figure 3.1 shows the how MontyLingua transform a natural language statement into the subject/verb/object triplets.

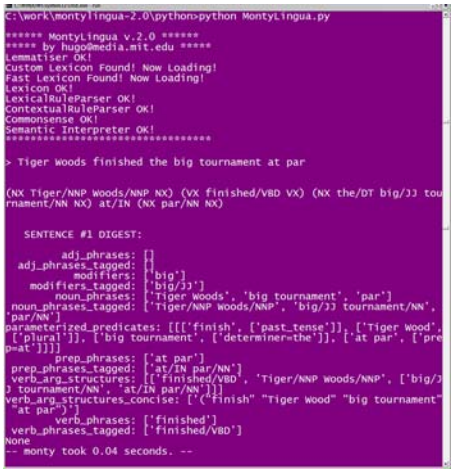


Figure 3.1 : A screenshot of MontyLingua output

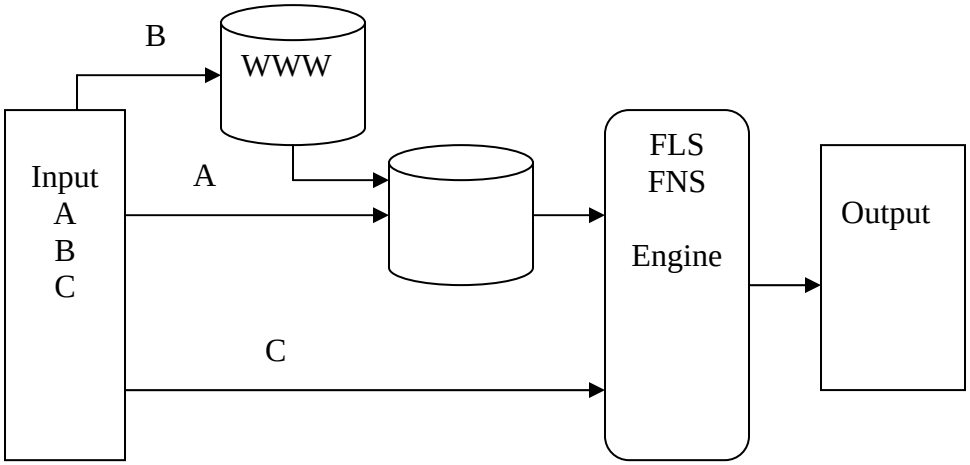


Figure 2.1 : Web query in Protoform for FSS

4. Conclusion

How web queries can be abstracted and precisiated in PTL and RDF is described. A comparison of few typical cases in both PTL and RDF is given. Statements in RDF form can tap on XML tools for further processing though it lacks the ability to abstract fuzzy relation as PTL does. Both RDF and PTL require the web queries be in a technical form before it can be transformed. The MontyLingua tool appears to be a suitable pre-processor for this need. More research is needed in automating the entire process for a search engine that can take on natural language statements and respond intelligently.

5. References

[1] Zadeh, L.A., "Toward a perception-based theory of probabilistic reasoning with imprecise probabilities", Journal of Statistical Planning and Inference, Volume 105, Issue 1 , 15 June 2002, Pages 233-264

[2] Lotfi A. Zadeh, "A New Direction in AI, Toward a Computational Theory of Perceptions", AI MAGAZINE, Spring 2001, Pages 73 – 84

[3] L A Zadeh, " From computing with numbers to computing with words. From manipulation of measurements to manipulation of perceptions.", Ann NY Acad Sci, Apr 2001; 929: 221-252.

[4] C. Tseng, T. Vu, "A perception-based web search with fuzzy semantic", *Fuzzy Logic in Semantic Web, Elsevier*, 2005

[5] Liu, Hugo (2004), "MontyLingua – An end-to-end natural language processor with common sense", http://web.media.mit.edu/~hugo/montylingua