Automated Goal Detection from Natural Language Constraints

Muhammad Khalid Mehmood
The Islamia University of Bahawalpur
Pakistan
momikh@gmail.com

M. Asif Naeem
Auckland University of Technology
Pakistan
mnaeem@aut.ac.nz

ABSTRACT**:** *This paper focuses on the development of an approach that can be used for automated detection of goals from software specifications text. Software specifications demonstration is the key analysis purpose of the provided analysis. Other analysis purpose is to examine that how goals are detected from business process texts. That is not only unambiguous and semantically limited but also machine process-able. The execution of the provided strategy for company procedure to NL and growth of model device that can instantly execute convert feedback NL company guidelines into sensible types and find dependencies among these guidelines is the main participation of this paper.*

Keywords: Ontology Engineering, Semantic Technology, Event Extraction, Natural Language Processing

Received: 8 January 2016, Revised 14 February 2016, Accepted 20 February 2016

1. Introduction

In software engineering discipline unified modeling language is well known modeling language which is used to intend and visualize a system in a regular way. The Unified Modeling Language (UML) offers a method to envision a system's architectural designs in a diagram together with elements such as: several jobs, mechanism of the scheme and how they can work together with other software interface, how the system will run, how entities communicate with others, external user interface. The Object Management Group (OMG) has designed a meta-modeling architecture to define the UML (OMG, 2009).

1.1 Constraints for UML Models

A package-able factor that is used to represent some limitation, condition or declaration relevant to some factor or several components. Constraint is generally accurate by Boolean statements which must evaluate to a true or incorrect. Constraint must be pleased by an appropriate design of the system. Constraints are usually used for a wide range of components on category blueprints.

Usually there are so many possible kinds of entrepreneurs for a restriction. Having factor must have availability to the restriction will choose when the restriction is to be analyzed. For example, function can have pre situation and publish situation restrictions. Constraint could have a recommended name, though usually it is unidentified. A software constraint is described in Example 1.1 as a written text series according to the following syntax:

Example 1.1

constraint : '{' [name ':'] Boolean-expression '}'

Unified Modeling Language specifications requirements do not limit languages which could be used to show restriction. Some illustrations of constraints languages are as follows:

1.2. Object Constraint Language

Object Constraint Language (OCL) is a language which is already defined in Unified Modeling Language but if is used to draw diagrams using some UML tools; any constraint language supported by that device should also be applied. For an element whose details are text statements for example a class attribute, the constraint text may pursue the text statements in curly braces.

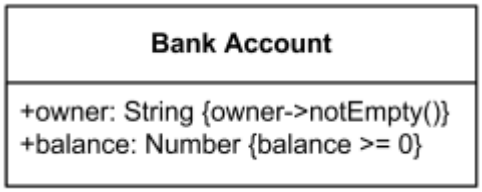

Figure. 1. A UML class

A constraint that is implements a factor e.g. an organization direction or a category, the constraint written text may be put near the icon for that factor, ideally near to the name, if any specified. A UML device must create it possible to determine the restricted written text claims. A constraint which is used to implement to two components such as two organizations or two sessions, the constraint may be proven as a dashed range between the components marked by the constraint sequence in wavy tooth braces.

It is shown in Figure 2 that Account owner is either corporation or person,is already defined in UML constraint. If the constraint is proven as a dashed range between two components, then an arrowhead may be located on one end.

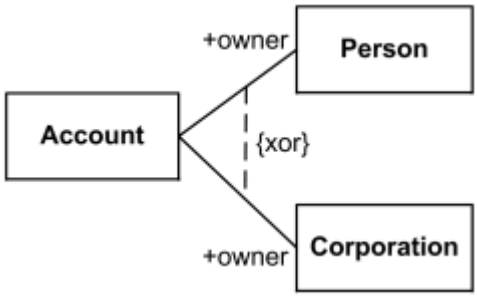

Figure 2. A UML class model

In Figure 2, the way of the pointer is relevant details within the constraint. The aspect at the end of the pointer is map to the first position and the aspect at the top of the pointer is planned to the second position in the constraint aspect selection.

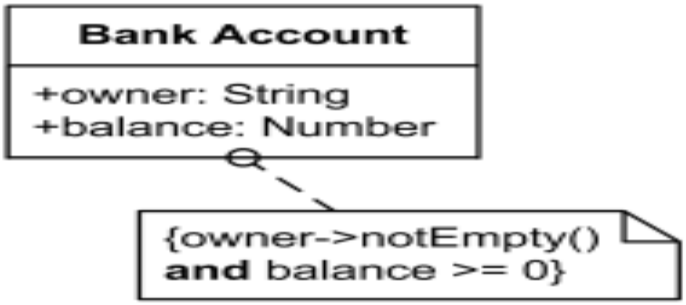

Figure 3. A UML class model

If there are three or more routes of the same type such as organization routes or generalization routes, the constraint may be placed to a dashed range passing all of the routes. The constraint text messages may be placed in a observe icon same as used for feedback and placed to each of the signs for the restricted text messages by a dashed range.

1.3. Goal Detection for NL Constraints

An approach that can be used for development for automated detection of goals from Natural Language specifications text. Natural Language specifications demonstration is the very important analysis purpose for the provided analysis. Other research objective is to analyses that how goals are detected from business process texts. That is not only semantically restricted and unambiguous but also machine process able act. The execution of the provided strategy for company procedure to NL and growth of model device that is instantly execute convert feedback NL company guidelines into sensible types and find dependencies among these guidelines is the main participation of this thesis.

A usual procedure in software development is software specifications. SS (Software specification) is generally performed by domain experts in NL. But a NL is syntactically uncertain and semantically ambiguous. That's why, software specifications (SS) written in Natural Languages also become uncertain and unreliable. Indecisiveness and irregularity in application specifications cause misunderstandings in last levels of application acting and growth. In Natural Languages there are two types of complications that may come across such as syntactic indecisiveness and semantic inconsistency. Here, a possible indecisiveness in a particular English sentence can be multiple interpretations of English constraints by different visitors of the specifications requirements and constraints.

2.Related Work

Ontology engineering is a process of capturing the related peculiarities of a particular domain at abstract level to represent meanings of a set of terminology. Since, for analysis of data in any domain, domain specific knowledge and background details are required. However, in ontology engineering a challenging task is to develop a mechanism that assists in developing ontological peculiarities in a methodical and articulated way. A similar scenario is construction of computational ontologies for a set of data is by understanding "the real world" driven by "formal ontology" insight (Guarino, 1995). A typical practice in ontology engineering can be reusing large sized information formally coded in formerly developed ontologies (Berners-Lee, 2001). Such practices can help in exemplifying the outcomes of a theoretical research.

In this paper, an approach is presented to address the above mentioned problem. To address this problem, an approach is designed that automatically identifies events from natural language constraints and then the extracted events are stored in an ontology specifically comprised to handle event semantics. The approach presented in this chapter that is capable of identifying the concepts of interest (i.e., concepts related to constraints events) that are further represented in the form of domain ontology.

In the presented approach, each concept stored in the ontology also contains context of the concept extracted from WordNet (Fellbaum, 1998). The presented approach performs concept identification by using event-related concepts available in WordNet to find out candidate events from natural language constraints. Here, lexical processing is performed such as word-groupings and lemmatization. Additionally, steps like word sense disambiguation are used to improve accuracy by defining a particular sense of words.Additionally, word grouping helps in identifying complete traces of terms and concepts. Then among the total list of phrases events related phrases

are extracted with the help of patter recognition of patterns. Once, all the possible event phrase are extracted, shown to user for final approval and the event phrases approved by the user are looked up in to existing ontology and Wikipedia and then stored into the domain specific ontology for software constraints.

3.Used Approach

Software specifications are mostly defined using natural languages. Software Specifications can be defined using mathematical or formal logic. Defining Software models through above said process is difficult. On the other side, NL based constraints are generally informal. Hence, software rules cannot define in organic 'languages' due to informalities of organic 'languages'. But defining software rules in organic 'languages' is easy and time saving. Therefore specifications Gatherings in organic 'languages' can be made more efficient and consistent if Informalities of organic 'languages' can be overcome.The use of OCL for detecting goals is syntactically unambiguous and semantically consistent.

Application requirements elicitation and software requirements specifications are significant task of the application development procedure. Application requirements (SS) are taken in organic terminology (NL) and NL are syntactically unclear and semantically untrustworthy that's why they are difficult to analyze as well as complex to machine process. To fix this problem we present the new technique to improve identify the organic terminology software objectives from software models. This strategy will take the Natural Language Application Restrictions as a feedback such as British terminology written text and instantly identify objectives from these constraints. The provided strategy works as the application professional information a piece of British requirements of software requirements and our provided strategy is capable to converts feedback written text to unambiguous software objectives. The recognition of software objectives based software requirement requirements involved some steps:

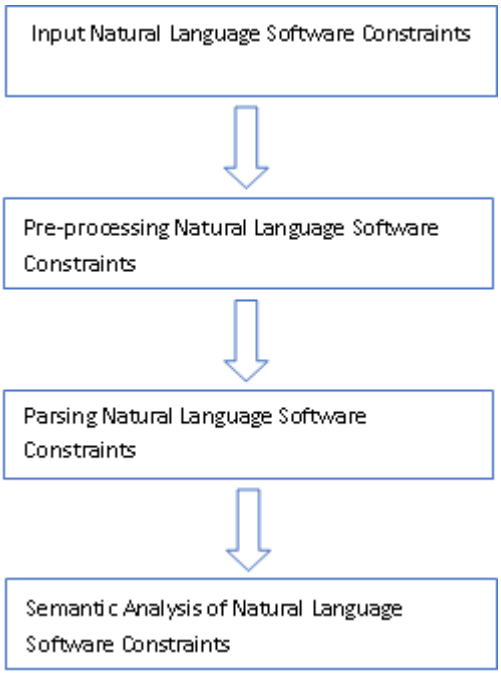

Figure 4. Proposed Architecture

The proposed framework used for goal detection from natural language constraints has four main steps as shown in Figure 4.

- Input Natural Language Software Constraints
- Pre-processing natural language constraints
- Parsing Natural Language Software Constraints
- Semantic Analysis Of Natural Language Software Constraints

Initially, input software constraints are lexically, syntactically and semantically parsed and then logical representation of these constraints is discussed. The last phase is analysis of these constraints some knowledge base rules. The output of this system is detection of goals.

3.1 Input Natural Language Software Constraints

The input of our system is Natural Language software constraints is given in a text file. Following is the example of inputs, we used in the experiments:

```
"Before a withdrawal, the balance of the account must be
greater than the amount being withdrawn".
```

3.2 Preprocessing Natural Language Software Constraints

The first phase of natural language processing is to pre-process input text. This step involved number of steps i.e.tokenization,POS tagging, stemming, and removal of stop words and logical representation of these rules.

A) Tokenization

Lexical processing of the input text is start with tokenization.In the sentence analysis text is read from left to right and group into groups .Tokens are series of characters with collective meaning.

```
[Before] [a] [withdrawal] [the] [balance] [of] [the]
[account] [must] [be] [greater] [than] [the] [amount] [being]
[withdrawn] [.]
```

B) Lemmatization

Lemmatization usually represents doing factors effectively with the use of a terminology and morphological research of terms, normally trying to eliminate inflectional being only and to come back the platform or vocabulary way of a term, which is known as the lemma. For example: A lemmatization program would manage related “car” to “cars” along with related “car” to “automobile”. In a more conventional online look for motor, related “car” to “cars” would be managed by arising, but related “car” to “automobile” would be managed by an individual system.

C) POS Tagging

In POS labelling each symbol from above level analyzed and categorized into its particular POS classification such as action-word, adverb, noun, pronoun, adjectives, assisting action-word, interjection, propositions, combination etc. The Stanford Part-Of-Speech tagger can identify these aspects of conversation tags.

```
Before/IN  a/DT  withdrawal/NN,  /,  the/DT
balance/NN  of/IN  the/DT  account/NN  must/MD
be/VB  greater/JJR  than/IN  the/DT  amount/NN
being/VBG withdrawn/VBN. /.
```

4) Parsing Natural Language Software Constraints

A number of phases are involved in parsing of natural language software constraints. These phases are described below:

4.1) Generating Parse Tree

A diagrammatic representation of the parsed structure of a sentence or string.

```
(ROOT
  (S
    (PP (IN Before)
      (NP (DT a) (NN withdrawal)))
    (, ,)
    (NP
      (NP (DT the) (NN balance))
      (PP (IN of)
        (NP (DT the) (NN account))))
    (VP (MD must)
      (VP (VB be)
        (ADJP
          (ADJP (JJR greater))
          (PP (IN than)
            (NP
              (NP (DT the) (NN amount))
              (VP (VBG being)
                (VP (VBN withdrawn))))))))
    (. .)))
```

Figure 5 shows graphicaltree that is generated using probabilistic LFG parse

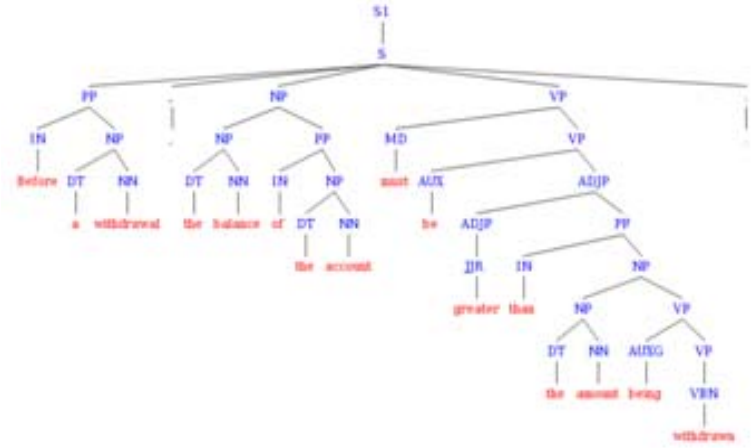

Figure 4. Tree of Example

Following typed dependencies are generated usin stanford parser.

4.2) Typed dependencies

A set of dependencies are generated for the parse tree shown above.

```
prep(greater-12, Before-1)
det(withdrawal-3, a-2)
pobj(Before-1, withdrawal-3)
det(balance-6, the-5)
nsubj(greater-12, balance-6)
prep(balance-6, of-7)
det(account-9, the-8)
pobj(of-7, account-9)
aux(greater-12, must-10)
cop(greater-12, be-11)
root(ROOT-0, greater-12)
prep(greater-12, than-13)
det(amount-15, the-14)
pobj(than-13, amount-15)
auxpass(withdrawn-17, being-16)
partmod(amount-15, withdrawn-17)
```

5) Semantic Analysis of Natural Language Software Constraints

Semantic analysis of natural language of software constraints are processed in the next stage after the parse tree and dependencies are generated.

5.1) Shallow Semantic Parsing

Superficial semantic parsing is labelling words of a phrase with semantic positions with regard to a focus on term. Labelling is the procedure of labelling (or tagging) each term in a phrase with its appropriate aspect of conversation. We choose whether each term is a noun, action-word, adjective, or whatever. Superficial parsing (also chunking, "light parsing") is a research of a phrase which recognizes the elements (noun categories, verbs, action-word categories, etc.), but does not specify their inner framework, nor their aspect in the primary phrase.

A typical semantic research results in a sensible way of a phrase. Logical type is used to catch semantic significance and illustrate this significance separate of a particular perspective (Lu et al., 2008). The objective of semantic research is to view the real explanations of the opinions released written text and identify that connection in various segments. For a complete semantic research of industry particular released written text, we have to assess the composing in respect of particular industry such as the UML classification style. Sector particular released written text research specifications information from the system industry to be organized with the opinions English.

In trivial semantic parsing the semantic or thematic positions are usually allocated to syntactic framework in a NL term. This process is also known as Semantic Aspect Labeling.

The real objective of semantic part labelling is determining connection of affiliates (semantic arguments) with the main action-word (semantic predicate) in a situation. SRL is a most common way of such as term semantics of NL released published written text. Semantic labelling on a substring (semantic predicate or a semantic argument) in a restriction (NL sentence) "S" can be used. A series of actions was conducted for labelling semantic positions to particular semantic predicates

5.2) Deep Semantic Parsing

Generally computational semantics aim at studying the whole explanations of the natural terminology term, rather than working on released written text segments only. For computational semantics, we need to assess the highly effective semantics of the opinions released published written text. The highly effective semantic research contains growth of a fine-grained semantic reflection from the opinions released published written text. Various aspects are engaged in highly effective semantics research. In natural 'languages', quantifications are usually indicated with noun conditions (NPs). However, in First-Order Considering (FOL), the aspects are quantified at the beginning of the sensible overall look. Usually, the natural terminology quantifiers are much more strange and different. This vagueness makes demonstration of NL to FOL complicated.

5.3) Goal Detection in Natural Language Software Constraints

In the last phase, we detect goals from the natural language software constraints. For goal detection, the subject of the each sentence is identified and the subject of the sentences if identified with the help of the dependencies

4.Experiments And Results

This section discusses the experimentation with the demo system constructed. This chapter also presents the results of the experiments. In this section results are calculated using above said case study and 3 more solved elsewhere, in which we calculated results. Using these values we can calculate the value of recall, precision and F measure.

$$\text{Recall} = \frac{\text{matched NL Constraint}}{\text{Total NL Constraint}}$$

$$\text{Precision} = \frac{\text{relevant NL Constraint}}{\text{retrieved NL Constraint}}$$

$$F = \frac{2(P)(R)}{P + R}$$

Case Studies	Pre%	Rec%	F-Measure
1	88.9	82.8	85.7
2	86.2	73.5	79.3
3	81.8	72.6	76.9
4	80.0	72.7	86.7
Average	84.2	75.4	82.1

Table 5.1. Values of Precision, Recall and F measure

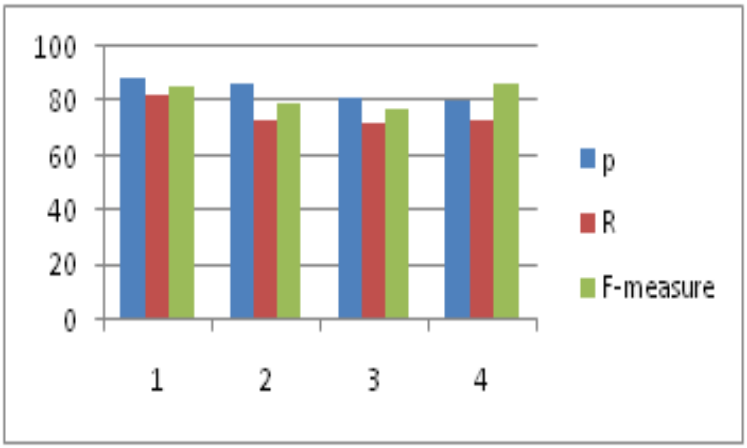

Figure 5. Chart shows the Precision, Recall and F measure values of all case studies

Following are charts of above said case studies. Figure 6 Showing case studies of Precision, Recall and F measureCase studies relating to business rules are solved in this chapter. All phases that are defined in figure 4.1 in chapter 4 are solved in these case studies one by one. Tables and graphs which are presented in this chapter show the effectiveness of our system. According to our outcomes in desk 5.1 display that Remember (84.2%) and Perfection (75.4%) outcomes implementing on the used research for business concept by using our systems are very appropriate. According to desk 5.1 measured F-Measure 82.1 is quite optimistic.

Conclusion And Future Work

The main purpose of this study was to improve the procedure of goals detection from NL based software by solving the ambiguity problem of Natural Languages (NL).To deal with this task we have present a natural language based Stanford parser to parse Natural constraints and detect goals. Stanford parser is able to evaluate the piece of written text such as English which includes syntactic, lexical understanding and semantic explanation in its first phase and after getting propositions, our tool detects goals from these constraints its last phase.

Tests and assessment are conducted efficiently on different case studies. Hence, the outcomes of our evaluation clearly display the effective efficiency of program in term of use correctness, capability and time. The result of recommended device can be used for computerized OO (Object Oriented) research and style from NL centered software written text. Furthermore, recommended program provides a higher precision as in comparison to other available NL centered resources. Formally, a lot of research has been done to the automated of software requirements (SR) using natural language processing techniques, relatively little effort has been done on the techniques depending on requirement specification of businessrules. Therefore, many phases are still needs to be investigated while using Natural Languages (NL) for recognition of dependencies

of business constraints. As, the long run work is to attract the OO information from business rules (BR) of software requirements such as instances, classes and their particular functions, functions, aggregations, organizations and summary. Removal of such information can be beneficial in automated acting of NL.This analysis is all about the powerful recognition of goals from natural terminology constraints and then generates UML designs and their particular rule by reading and analyzing the given situation in British terminology provided by the user. The designed system can find out the classes and things and their functions using a synthetic intelligence technique such as natural terminology managing. Previously, a large amount of analysis has been controlled to the automated of program requirements using Natural Language managing methods, relatively little execute has been done on the methods based on control reflection of need requirements. Therefore, many expect are still needs to be investigated AS, the long run execute is to attract out the object-oriented information from program requirements such as classes, circumstances and their particular functions, functions, organizations, aggregations, and summary. Computerized removal of such information can be helpful in automated conceptual acting of natural terminology program need requirements.

References

[1] Venkatesan, M. D. (2005). BPEL Validation with Object Constraint Language (OCL). Studienarbeit. TU Hamburg-Harburg.

[2] Meziane, F., Athanasakis, N., Ananiadou, S. (2008). Generating Natural Language specifications from UML class diagrams. *Requirements Engineering*, 13 (1) 1-18.

[3] OMG, O. (2007). Unified Modeling Language (OMG UML). Superstructure.

[4] Perez-Gonzalez, H. G.,&Kalita, J. K. (2002). GOOAL: A graphic object oriented analysis laboratory. *In:* Companion of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (p. 38-39). ACM.

[5] Babar, M. A., & Gorton, I. (2007). A tool for managing software architecture knowledge. *In:* Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (p. 11). IEEE Computer Society.

[6] Liu, X. (2009). Process oriented analysis for software automation. In Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on (Vol. 1, p. 1131-1133). IEEE.

[7] Santiago Jr, V. (2010). Natural language requirements: automating model-based testing and analysis of defects. InstitutoNacional de PesquisasEspaciais, São José dos Campos.

[8] Bajwa, I. S., Bordbar, B., Lee, M. G. (2010). OCL constraints generation from natural language specification. *In:* Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International (p. 204-213). IEEE.

[9] Ilieva, M., Boley, H. (2008). Representing Textual Requirements as Graphical Natural Language for UML Diagram Generation. *In:* SEKE (Vol. 8, p. 478-483).

[10] Zapata, C., Gelbukh, A., & ARANGO, F. (2007). A Novel CASE Tool based on Pre-Conceptual Schemas for Automatically Obtaining UML Diagrams. *Avances en sistemas einformatica*, 4 (2) 117-124.

[11] Bajwa, I. S., Samad, A., Mumtaz, S. (2009). Object oriented software modeling using NLP based knowledge extraction. *European Journal of Scientific Research*, 35(01), 22-33.