BASIC DEPENDENCY PARSING IN NATURAL LANGUAGE INFERENCE

Aleshinloye Abass Yusuf
Department of Computer Science
Nile University of Nigeria, Abuja.
yusuf.abass@nileuniversity.edu.ng

Nnanna Agwu Nwojo
Department of Computer Science
Nile University of Nigeria, Abuja
nagwu@nileuniversity.edu.ng

Moussa Mahamat Boukar
Department of Computer Science
Nile University of Nigeria, Abuja
musa.muhammed@nileuniversity.edu.ng

Abstract*— Parsing is the process of analyzing a sentence for it structure, content and meaning, this process uncover the structure, articulate the constituents and the relation between the constituents of the input sentence. This paper described the importance of parsing strategy in achieving entailment in natural language inference. Parsing is the basic task in processing natural language and it is also the basis for all natural language applications such as machine learning, question answering and information retrieval. We have used the parsing strategy in natural language inference to achieve entailment through an approach called normalization approach where entailment is achieved by removing or replacing some nodes as well as relations in a tree. This process requires a detailed understanding of the dependency structure, in order to generate a tree that does not contain nodes and relations that are irrelevant to the inference procedure. In order to achieve this, the dependency trees are transformed by applying some rewrite rules to the dependency tree.

Keywords—parsing; entailment; unification; dependency

I. INTRODUCTION

Parsing is simply the term used in describing the processes that are required in breaking down a given sentence into its constituent words in order to find the part of speech of each word or grammatical components of each word in the sentence. Parsing can also be defined as the decomposition of the input (sentence) into some more precisely processed components. To analyze a sentence, the components and meaning of the sentence is very important. For example, let consider a sentence like "John likes Mary". The Natural language processing applies the concept of parsing to natural language sentence. In natural language, parsing is termed as a method used in analyzing an input sentence based on it grammatical constituents, syntactic relations among the sentence constituent and part of speech.
Parsing is a procedure used in determining how a string of terminals (sentence) is generated from its constituents; it involves breaking down the sentence into tokens. Every corresponding word in the sentence is term token. For example, "John", "likes", "Mary" are all tokens for the above sentence. Every natural language is made of up grammatical rules that determine how sentence are formed. Parsing helps in discovering how sequences of rules are applied for the purpose of sentence generation in that particular language. Parsing natural language sentence can be seen as making a sequence of disambiguation decision: determining the various parts of speech of the tokens (words), choosing between possible constituents structures and selecting labels for the constituents [6].

Part of speech can be defined as the categorization of words in any given sentence based on the syntactic behavior. Every natural language has its own part of speech; in this work we are concerned with the part of speech in English language. The English language has eight part of speech which are noun, pronoun, verb, adverb, adjective, preposition, conjunction and article. Considering the example John likes Mary. The part of speech for each word in the sentence are noun for "John" and "Mary" and verb, proverb, conjunction, noun, adjective and preposition for "like" or "likes". Determining the correct part of speech for a word in a sentence having multiple parts of speech requires making a disambiguous decision. Ambiguity simply means having more than one meaning for a word in a sentence. Parsing technique is used to determine the exact parse for a word or a sentence. Parsing a word or sentence through a parser helps to generate a parse tree, which represent the graphical order in which the grammar productions are applied during the parsing process.

II. PROBLEM DEFINITION

When sentences are fed into the parser, in most cases it gives incorrect output. It is quite difficult to use those dependency trees in the inference system, and there is little or nothing to be done regarding the output from the parser. The research scope includes investigating sentences (rules) where the parser gives the right representation.
In this paper, we present an improvement on the standard tree-matching approach of textual entailment by introducing inference engine for every standard sentence which is more efficient in doing inferencing than the textual entailment approach and the logical approach.

978-1-5386-2501-9/17/$31.00 ©2017 IEEE

III. PROPOSED SYSTEM

Our proposed system is a compromise between the translation to logical approach and textual entailment approach. The compromise is what we call the normalization approach to NLI. In normalisation approach, an entailment is achieved by normalising the dependency parse tree in order to obtain a simpler version of the parsed tree. Figure 1 below shows features, such as text tagging, dependency tree parsing and dependency tree extraction from a known dependency parser. These features are the commonality that exists between the translation to the logical form approach and the textual entailment approach. Based on this commonality, the normalised inference form is developed, for which there exists an inference engine. The normalisation approach is the basis on which this paper is built, where the inference engine is a simple backward-chaining PROLOG-like engine. This approach basically uses a dependency tree structure that is generated from a dependency parser, rather than using first order logic (FOL) as it was in the translation to a logical form.
The aim of Normalization approach to NLI as shown in figure 1 is to set the background knowledge to a set of facts and rules, and it is necessary to prove a goal in the inference engine based on the known facts and set of rules. The way to achieve this is to follow what is obtainable in the system architecture of the Normalization approach.

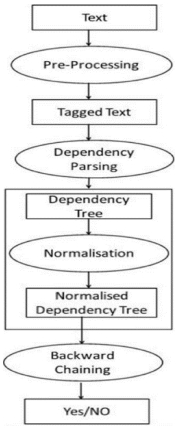

Figure 1: System Architecture for Normalization approach.

The three major tasks required in the Normalization approach are:
i. Parsing the input text (facts, rules and goal) into dependency trees.
ii. Normalizing the dependency tree, so that they are in a suitable format for the inference engine.
iii. Doing proof via the inference engine.

A. First Stage: Dependency Parsing

The task of a dependency parser is to take an input text and impose on the text an appropriate set of dependency links; that is, to tokenize each input text; on each token, the tagger assigns part of speech (POS) tags [3]. The parser generates a tree based on the relationships that exist between tokens or words in the input text. The following are some of the basic characteristics expected of a dependency tree generated from the Stanford parser:

B. Second Stage: Normalizing Dependency Trees

The parser is likely to generate a dependency tree and often time the trees generated are not quite suitable to represent a set of rules in the inference engine. In order to obtain a set of rules that a PROLOG-like inference engine can handle, an approach is required to transform the dependency tree. It is necessary to normalize the single dependency tree produced by the parser. The Normalization approach employs the use of hand coded rules, which is an indication that it is rich in background knowledge.
The normalization of a dependency tree is the process of removing or replacing some nodes as well as relations in the tree. This process requires a detailed understanding of the dependency structure, in order to generate a tree that does not contain nodes and relations that are irrelevant to the inference procedure. In order to achieve this, the dependency trees are transformed by applying some rewrite rules to the dependency tree.

C. Third Stage: Inference Framework

Generally, most inference representations are built by applying some kind of transformation to the trees representing the set of rules. Such transformations are viewed as the main inference rules, which capture semantic information, lexical relations, syntactic variations etc. Like the PROLOG system, the inference framework is composed of rules. The propositions include facts (the antecedent), goals (consequent) and the intermediate premises inferred during the proof (set of rules). The task of the inference rule is to define the way propositions are derived from previously established ones [1].

IV. THE NORMALIZATION APPROACH

In the normalization approach input sentence(s) which are in the form of a natural language are translated into a restricted subset of the same natural language.
In this case, the first stage is to generate a dependency tree, obtained from a dependency parser. In most cases such a tree is not always ideal for use with the inference engine in the third stage. A normalization procedure is applied to such tree so as to adapt to one that an inference engine can handle. The type of normalization that will be required is dependent on the structure of the tree generated from the dependency parser. For example

i. *X buys Y if X has money and X wants Y.*

Figure 2 shows the dependency tree representation of the parser for each of the sentence above:

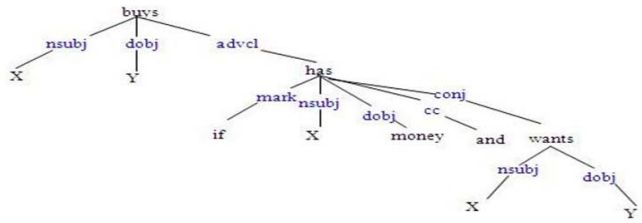

Figure 3: The dependency trees for examples (i), where the root of the tree is the context word

Figure 2 shows the dependency trees of (i), where the root of the dependency tree conveys significant information between the rests of the words. The dependency tree in Figure 2 (i) shows the kind of dependency structures

required for the natural language inference based on the research goal; it is not perfect, but there are mechanisms in the implementation that address such challenges.
The dependency tree in Figure 3(i) is amongst the few instances where the parser got it right. The adverbial clause modified the verb phrase. The root of the sentence is *buys*, and every other word is linked to it, and in the case of the coordinating conjunction, a good representation was given between *conj (has*, *wants*), as shown in Figure 3(i) above.
In the context of the present research, normalization is all about modifying the set of rules that the inference engine will be able to handle. Figure 4 shows the tree that represents the set of rules in the inference engine that needs to be transformed.

```
[('buys',[('nsubj',('X',[])),
          ('dobj',('Y',[])),
          ('advcl',('has',[('mark',('if',[])),
                           ('nsubj',('X',[])),
                           ('dobj',('money',[])),
                           ('cc',('and',[])),
                           ('conj',('wants',[('nsubj',('X',[])), ('dobj',('Y',[]))]))]))])]
```

Figure 4: Showing the set of rules that needs to be transformed
The set of rules *r* is made up of two parts, the consequent part and the antecedent part.
Considering Figure 4 above, the normalization process starts with the removal of the adverbial clause alongside its *marker*, which is the word that introduces the conditional part of the antecedent in *r*. Figure 5 shows the structure of *r* after applying the normalization rule:

```
[('buys', [('nsubj', ('X', [])),
           ('dobj', ('Y', []))]),   ⇐

 ('has', [ ('nsubj', ('X', [])),
           ('dobj', ('money', [])),
           ('cc', ('and', [])),
           ('conj', ('wants', [('nsubj', ('X', [])),
                               ('dobj', ('Y', []))]))])]
```

Figure 5: Application of normalization rule on adverbial component
It is somewhat obvious that the tree has been divided into its constituent part of antecedent and consequent. The transform tree, which is also a rule, is one of the inputs in the inference engine on which the inference engine attempts to prove a given goal. The consequent part of *r* is made up of relations and nodes that have the same pattern as the goal, as shown in Figure 6(a):

```
goal=('buys',[('nsubj', ('Mary', [])),
              ('dobj', ('car', []))])
```

(a)

```
facts=[('has',[('nsubj', ('Mary', [])),('dobj', ('money', []))]),
       ('wants',[('nsubj', ('Mary', [])),('dobj', ('car', []))])]
```

(b)

Figure 6: The goal and set of facts that needs to be proved.

The normalization approach is based on consequent inference; that is, inference from goal to known facts. It is a very good method of controlling a search, and is a simple mechanism for enforcing a certain form of relevance in action towards deciding the current goal and ensuring the use of rules that are present in the inference, which mention the current goal in conclusion. It is quite important to note that anything deducible via consequent inference is also deducible by antecedent inference, where the consequent acts as a way of controlling the antecedent. The inference engine exhibits backward chaining (consequent-driven, goal-driven and hypothesis-driven), and supports the goal state by checking the known facts in the set of rules, and if these facts do not support the goal, then the preconditions needed for the goal are set to sub-goals. The backward chaining involves the use of rules. Each rule is associated with a pattern by which the inference engine accesses it. The inference engine maintains a data structure that describes important aspects of the situation, and it is represented in a PROLOG-style which uses predicate names (*head*) to represent a relation, and constants or variables to represent the values. The data structure is of the form *relation* [*value1*, *value2*]. The attribute–value pairs are referred to as conditions to be tested by the set of rules. Rules are matched against a goal, and those parts (patterns) of the rule that match the goal patterns are unified via variable binding from the rules.

V. BACKWARD CHAINING STRATEGY

Among the core functions of the inference engine is determining how a set of rules is applied in order to infer a goal. The inference engine role, as captioned above, is to use a set of predetermined rules to define different strategies that are required in the inference process, the inference strategy of interest is the backward chaining inference strategy.
The backward chaining strategy is a goal-driven strategy, which works backwardly from goals, chaining through sets of rules in order to find known facts that support a proof. The backward chaining strategy is essentially a selection process which is primarily applicable when there is a small number of goals and a large number of facts. If a set of facts is derivable with the application of the backward chaining strategy in the inference engine, such facts can be used to prove a corresponding initial goal.

VI. UNIFICATION PROCEDURE

In order to apply the inference rule over dependency trees, an inference engine should have the capacity to determine when two trees match. The procedure that is used to determine the substitutions needed to make two patterns match is called a unification algorithm. The description of the algorithm is based on the one used by the PROLOG language. In order to use unification and the inference rule in a system such as the Normalisation approach to NLI, the inputs must be expressed in a suitable format. The following are the assumptions that are associated with the unification algorithm in PROLOG:

i. All variables must be *universally* quantified. Whenever a variable appears in a dependency tree, the assumption is that such a variable is universally

quantified, which enables substitutions to be made with ease.

ii. Variables that are *existentially* quantified are eliminated and replaced with constants that maintain the dependency tree in the right format.

Figure 7 shows an instance in the implementation of the Normalisation approach where the terms in the goal found a match in the head of the rules, and the variables in the head of the rules were unified with the terms in the goal via binding.

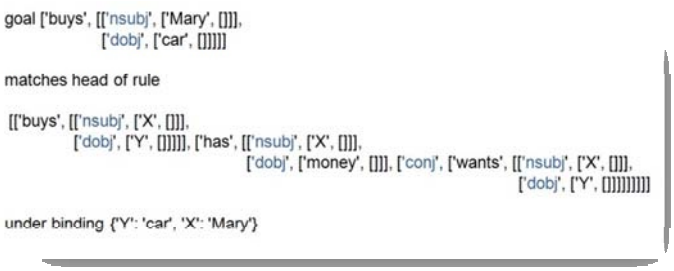

Figure 7: Showing the matching of the goal with the head of the rules

VII. RELATED WORKS

1. **The Shallow Inference**: Most effective NLI systems have relied on simple surface representations and estimated measures of lexical-syntactic similarity so as to determine whether the meaning of *H* is subsumed by the meaning of *T* [5]. The systems that are based on the lexical or semantic overlap, pattern-based relation extraction, as well as approximate matching of the predicate argument structure are classified as being shallow approaches [4]. Among the challenges of the shallow approaches are:
    a) The inability of taking semantic representations into consideration.
    b) They lack the ability to take into account complex background knowledge.
    c) They are not always sound.
2. **Textual Entailment and Syntactic Graph Distance**: Graphs are one of the most powerful data structures, and can be used in a wide range of applications. They are most often used to represent known models, which are stored in databases as unknown objects and are yet to be recognised [2]. Because it is possible to represent hypothesis *H* and text *T* as syntactic graphs, by implication, this means that the textual entailment recognition problem can be viewed as a graph in terms of its similar measure of estimation. However, [7] outlined the properties of textual entailment as:
    a) It is not symmetric.
    b) Similarities that exist between nodes cannot be reduced to label level.
    c) Similarity should be estimated on the basis of linguistically motivated graph transformation.

VIII. CONCLUSION

Parsing helps in analyzing the input text (words) in a sentence by associating additional components to the sentence. In Natural language Inference, parsing has been used to give an insight into the sentence structure, the sentence structure (tree) gives background knowledge into the sentence and with this; it is possible to know if an entailment can be achieved. Parsing helps in defining the functional relationships between the words (tokens) in a sentence. The tokens are represented in a tree format which the inference engine act upon using backward chaining strategy in order to achieve entailment by considering the antecedent and consequence components of the tree.

References

[1]. Bar-Haim, R., Dagan, I., Greental, I., & Shnarch, E. (2007). *Semantic inference at the lexical-syntactic level.* Paper presented at the Proceedings of the National Conference on Artificial Intelligence.

[2]. Bunke, H., & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern recognition letters, vol. 19*, page 255-259.

[3]. Covington, M. A. (2001). *A fundamental algorithm for dependency parsing.* Paper presented at the Proceedings of the 39th annual ACM southeast conference.

[4]. Dagan, I., O. Glickman, et al. (2006). "The pascal recognition textual entailment challenge." Machine Learning Challenge. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognition Textual Entailment: 177-190.

[5]. MacCartney, B. (2009). *Natural language inference.* (Ph.d Dissertation), Stanford University.

[6]. Magerman, D. M. (1995). *Statistical decision-tree models for parsing.* Paper presented at the Proceedings of the 33rd annual meeting on Association for Computational Linguistics.

[7]. Pazienza, M. T., Pennacchiotti, M., & Zanzotto, F. M. (2005). *Textual entailment as syntactic graph distance: a rule based and a SVM based approach.* Paper presented at the Proceedings of the recognizing textual entaiment challenge workshop.