D

Dependency Parsing: Recent Advances

Ruket Çakıcı
ICCS School of Informatics,University of Edinburgh, UK

INTRODUCTION

Annotated data have recently become more important, and thus more abundant, in computational linguistics . They are used as training material for machine learning systems for a wide variety of applications from Parsing to Machine Translation (Quirk et al., 2005). Dependency representation is preferred for many languages because linguistic and semantic information is easier to retrieve from the more direct dependency representation. Dependencies are relations that are defined on words or smaller units where the sentences are divided into its elements called heads and their arguments, e.g. verbs and objects. Dependency parsing aims to predict these dependency relations between lexical units to retrieve information, mostly in the form of semantic interpretation or syntactic structure.

Parsing is usually considered as the first step of Natural Language Processing (NLP). To train statistical parsers, a sample of data annotated with necessary information is required. There are different views on how informative or functional representation of natural language sentences should be. There are different constraints on the design process such as: 1) how intuitive (natural) it is, 2) how easy to extract information from it is, and 3) how appropriately and unambiguously it represents the phenomena that occur in natural languages.

In this article, a review of statistical dependency parsing for different languages will be made and current challenges of designing dependency treebanks and dependency parsing will be discussed.

DEPENDENCY GRAMMAR

The concept of dependency grammar is usually attributed to Tesnière (1959) and Hays (1964). The dependency theory has since developed, especially with the works of Gross (1964), Gaiffman (1965), Robinson (1970), Mel'čuk (1988), Starosta (1988), Hudson (1984, 1990), Sgall et al. (1986), Barbero et al. (1998), Duchier (2001), Menzel and Schröder (1998), Kruijff (2001).

Dependencies are defined as links between lexical entities (words or morphemes) that connect heads and their dependants. Dependencies may have labels, such as *subject, object*, and *determiner* or they can be unlabelled. A dependency tree is often defined as a directed, acyclic graph of links that are defined between words in a sentence. Dependencies are usually represented as trees where the root of the tree is a distinct node. Sometimes dependency links cross. Dependency graphs of this type are non-projective. Projectivity means that in surface structure a head and its dependants can only be separated by other dependants of the same head (and dependants of these dependants). Non-projective dependency trees cannot be translated to phrase structure trees unless treated specially. We can see in Table 1 that the notion of non-projectivity is very common across languages although distribution of it is usually rare in any given language. The fact that it is rare does not make it less important because it is this kind of phenomena that makes natural languages more interesting and that makes all the difference in the generative capacity of a grammar that is suggested to explain natural languages.

An example dependency tree is in Figure 1. The corresponding phrase structure tree is shown in Figure 2. The ROOT of this tree is "*hit*".

Given the basic concept of dependency, different theories of dependency grammar exist. Among many well known are: Functional Generative Description (Sgall et al., 1969, 1986), (Petkevič, 1987, 1995), Dependency Unification Grammar (DUG) Hellwig (1986, 2003), Meaning Text Theory (Gladkij and Mel'čuk, 1975), (Mel'čuk, 1988) and Lexicase (Starosta, 1988), Topological Dependency Grammar (Gerdes and Kahane, 2001). Kruijff (2001) also suggests a type of logic for dependency grammar, "Dependency Grammar Logic" which aims transparent semantic interpretation during parsing.

There are many open issues regarding the representation of dependency structure. Hays (1964)

Figure1. Dependency Tree for the sentence "The red car hit the big motorcycle"

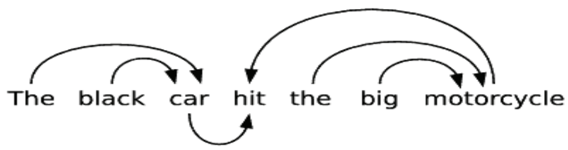
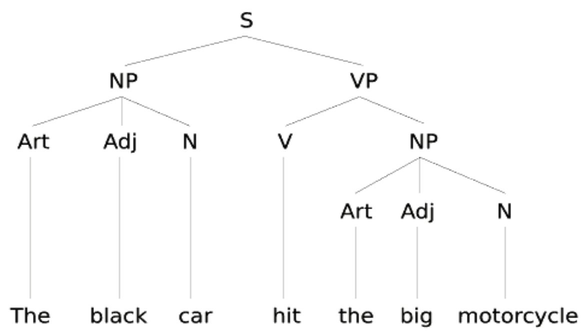


Figure 2. Phrase Structure Tree for the sentence in Figure 1



and Gaifman (1965) take dependency grammars as special cases of phrase structure grammars whereas Barbero et al. (1998), Menzel and Schröder (1998), Eisner (2000), Samuelsson (2000), Duchier (2001), Gerdes and Kahane (2001), Kruijff (2001) think they are completely different.

Generative capacity of dependency grammars has long been discussed (Gross, 1964), (Hays, 1964), (Gaifman, 1965), (Robinson, 1970). Dependency grammars were proved to be context-free (Gaiffman, 1965). When natural languages were proved to be not context-free, but in a class called "Mildly Context-Sensitive" (Joshi, 1985) they were abandoned until 90s, when Vijayashanker and Weir (1994) showed that Head Grammars -an extension of CFGs- (Pollard, 1984) are mildly context-sensitive like Tree Adjoining Grammar (TAG), (Joshi et al., 1975) and Combinatory Categorial Grammar (CCG), (Steedman,2000). Recently, Kuhlmann and Möhl (2007) defined "regular dependency languages" and showed that applying different combinations of gap-degree and well-nestedness restrictions on non-projectivity in these languages gave a class of mildly context-sensitive grammars.

DEPENDENCY TREEBANKS

Why Dependency Trees?

Many new corpora have been designed and created in the past few years. Dependency representation is preferred when these corpora are designed. This can be argued by the following properties of dependency trees:

1. They are easier to annotate than some other representation types like phrase structure trees (PST). There are fewer tags and labels (only as many as words in a sentence) and no internal nodes to name the phrases as in PSTs.
2. Some information such as predicate-argument structure can be extracted trivially from them which is not the case for PSTs.
3. Another interesting result is that some dependency parsers run much faster than PST parsers. Computational complexity of a standard PST parser is $O(n^5)$ whereas a non-projective DT parser runs in $O(n^2)$.

Related Content