Tree-based Translation

Ondřej Bojar, Adam Lopez

Revision: 169
2008-05-08 18:14:11 +0200 (Thu, 08 May 2008)

1 Dependency vs. Constituency Trees

Syntactic structure of sentences can be represented using **constituency trees** or **dependency trees**.

Constituency trees indicate recursive "bracketting" of the sentence–sequences of words are grouped together to form constituents:

(1) John (loves Mary)

Dependency trees indicate which words depend on which. Nivre (2005) gives a good review of dependency-based formalisms and dependency parsing.

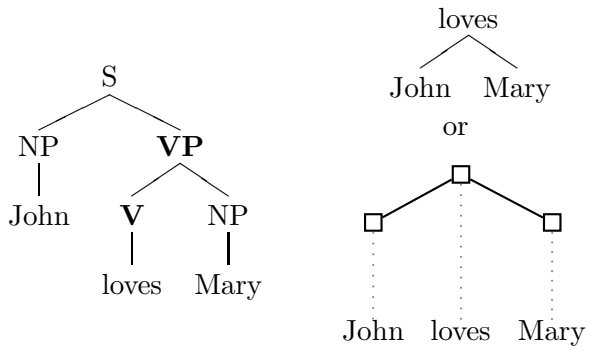


Figure 1: A constituency and a dependency tree. Non-terminals in bold mark heads. Following the trail of heads, we find the terminal node with the same label as the node in a dependency tree would have.

Figure 1 illustrates a constituency tree and a dependency tree. In constituency trees, each **non-terminal node** (labelled in capital letters) represents a constituent. There are no non-terminals in dependency trees. If we choose one of the sons in each constituent to be the **head** of the constituent, e.g. the VP to be the head of the S, we can convert the constituency tree

to a dependency tree by "lifting" the terminals up along paths marked with heads.

An **unordered dependency tree** is a connected rooted directed acyclic graph in graph-theoretic sense. An unordered dependency tree does not capture any linear order of words, just pure dependencies. We cannot speak about projectivity (see below) of unordered dependency trees.

An **ordered dependency tree** is an unordered dependency tree with a specified linear order of the nodes. We can thus draw the nodes in the tree from left to right (and the drawing actually means something).

A **constituency tree** can be defined e.g. as a term, using this recursive definition: 1) a terminal is a term, 2) if $t_1, \ldots, t_n$ are terms and N is a non-terminal, then $N(t_1, \ldots, t_n)$ is a term. In the graph-theoretic view, a constituency tree is a tree with linearly ordered sons of each non-terminal.

1.1 Crossing Brackets, Non-Projectivity

Here is a simple example of a sentence with "crossing brackets":

(2) Mary, John loves.

Constituency trees cannot represent structures where a constituent was "moved" outside of its father's span (unless we use empty constituents, sometimes called "traces", i.e. constituents spanning no words, optionally co-indexed with the "moved" words). Because there are no non-terminals in dependency trees to represent the derivation history, some of the "crossing brackets" structures just disappear, see Figure 2.[1]

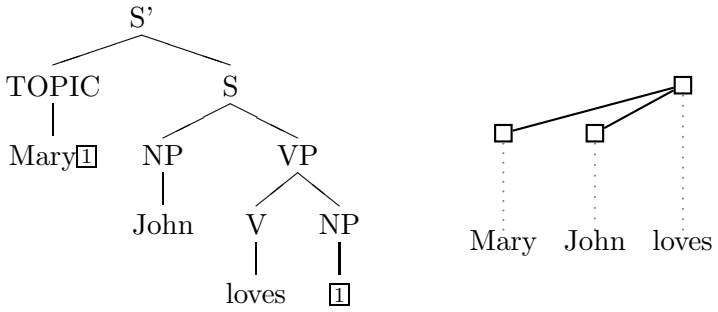


Figure 2: An example of a crossing-bracket yet projective structure.

There are however structures, such as the Dutch "cross-serial" dependencies where, even dependency trees become **non-projective**, i.e. there is a "gap" in the span of a subtree. Representing non-projectivity in dependency trees is easy and natural, see Figure 3.

[1] See the difference between a D-tree and a DR-tree as defined by Holan et al. (1998).

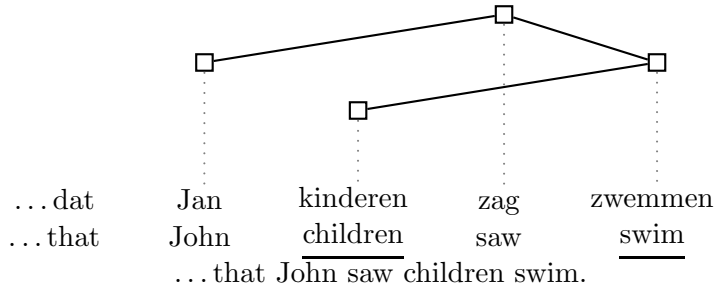


Figure 3: Dutch "cross-serial" dependencies, a non-projective tree with one gap caused by *saw* within the span of *swim*.

Non-projective structures can be relatively rare in English but amount to 23% of sentences in Czech, a Slavic language with relatively free word order (Debusmann and Kuhlmann, 2007).

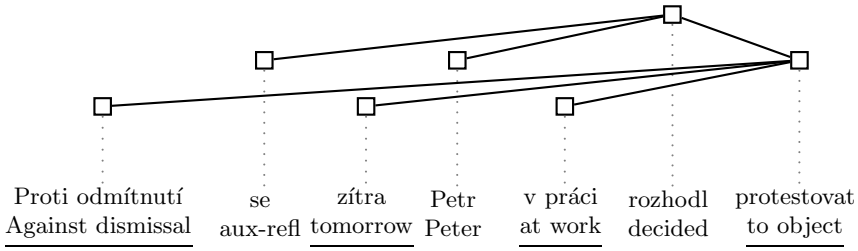
1.2 Gap Degree and Well-Nestedness

Holan et al. (1998) and Kuhlmann and Möhl (2007) define a measure of non-projectivity: **gap degree** is the number of gaps in a dependency structure. Gap-zero structures are projective structures.

Kuhlmann and Möhl (2007) define another constraint on dependency structures: in **well-nested** structures, disjoint subtrees must not interleave.

Debusmann and Kuhlmann (2007) evaluated that in the Prague Dependency Treebank (Hajič et al., 2006), 99.5% of structures are well-nested and up to gap-1, despite the fact that Czech grammar in principle allows unbounded pumping of gap-degree. The construction is based on two verbs and intermixed modifiers where the dependency relations are disambiguated based on syntactic criteria (e.g. obligatory reflexive particle *se* or subcategorization for a particular preposition or case) and semantic criteria (e.g. verb in past tense cannot accept time modifier refering to future):

(3)



Peter decided to object against the dismissal at work tomorrow.

The non-projective dependencies are *se* and *Peter* depending on the

main verb *decided* but appearing within the span of dependents of *to object*: *against dismissal*, *tomorrow*, *at work*. With the main verb itself, there are 3 gaps within the yield of *to object*.

2 Tree Grammars

Tree grammars are one type of finite formal means to define (infinite) sets of trees.

Tree-adjoining grammars (TAG, tag (), see also the review by Joshi et al. (1990)) start from a set of initial trees and use **tree substitution** and **tree adjunction** to derive a tree. The tree substitution operation attaches a treelet to a **frontier** (leaf non-terminal). The tree adjunction splits a tree in a non-terminal and stitches a treelet in between, see Figure 4. **Tree-substitution grammars** (TSG, Eisner (2003) or e.g. Bojar and Čmejrek (2007)) are like TAG but allow only tree substitution, no tree adjunction.

Figure 4: Tree substitution at frontier F and tree adjunction at internal node A.

Figure 5 illustrates how a sentence is analyzed using a constituency-based TSG and a dependency-based TSG. The difference between constituency- and dependency-based TSG is the type of underlying trees. Non-terminal nodes in a dependency-based TSG can appear as leaves of unfinished trees only and have to be substituted by a tree later in the derivation.

2.1 Constituency vs. Dependency Tree Adjunction

TAG defines the adjunction operation for constituency trees only. The same definition cannot be casted to dependency-based TSG (dep-TSG) because there are no internal non-terminals to adjoin at. However, we can still think of the "linguistic adjunction" in dep-TSG. This operation adds adjuncts to a node. In terms of TSG, a little tree gets attached to an internal node instead at a frontier. dep-TSG adjunction thus allows to add siblings to an already existing node.

The trouble starts if we consider ordered dependency trees. Where is the new dependent placed with respect of the existing dependents? And is the newly attached subtree attached projectively, or can older nodes in the tree introduce gaps into it? (And where the gaps are allowed to be?) E.g. Quirk

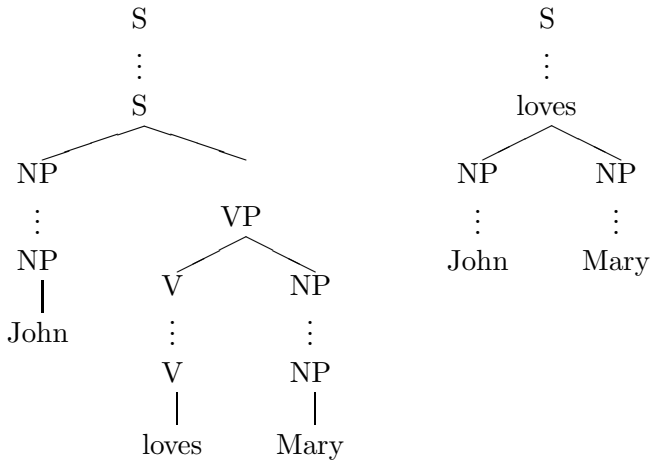


Figure 5: Derivation of a sentence using constituency-based and dependency-based tree substitutions. The substitution is indicated by ":".

et al. (2005) use a probabilistic model to interleave old dependents and newly adjoint dependents but do not seem allow non-projective attachments.

2.2 Remarks on Generative Capacity

This is by no means a complete survey.

Gaifman (1965) shows that *projective* dependency structures are weakly equivalent to CFG. We have already illustrated how marking of heads is used to convert a constituency tree to a dependency tree in Figure 1.

Joshi et al. (1990) describe various formalisms for so-called **mildly context sensitive** (MCS) grammars. The term MCS refers to various grammars beyond CFG but still polynomially parsable. TAG is one of them and was motivated by the need to represent Dutch cross-serial dependencies (Figure 3). Naturally, TAG needs traces in its constituency trees.

Kuhlmann and Möhl (2007) shows that lexicalized TAG (LTAG) is equivalent to well-nested dependency structures with at most one gap. kuhlmann-mohl:2007:ACLMain (also define an infinite hierarchy of mildly context-sensitive dependency structures (i.e. parsable in polynomial time) of ever growing weak generative power.

Plátek (2001) defines a special type of formal automata to define a hierarchy of languages beyond CFG. Jurdziński et al. (2008) shows that already the class of languages accepted by a quite restricted from of the automaton contains NP-complete languages and is thus not much useful for efficient parsing.

3 Synchronous Grammars

Synchronous grammars are vaguely speaking pairs of formalisms attached to each other. So instead of deriving a single tree, we obtain a pair of structures in a synchronous derivation.

Chiang and Knight (2006) gives a nice and brief overview of synchronous context-free grammars (SCFG), synchronous tree-substitution grammars (STSG) and synchronous tree-adjoining grammars (STAG), including parsing strategies. All his examples are based on constituency trees.

3.1 Translation Direction

When designing an MT system, one should consider the properties of the source and target languages.

For instance, when translating from Czech to English, source-side non-projectivities have to be accounted for. Alternatively, a non-projective dependency parser such as (McDonald et al., 2005) can be used and the resulting dependency tree can be tranfered to the target language using e.g. STSG.

When translating from English to Czech, significant portion of non-projective structures can be disregarded because there exists a grammatically correct reordering that reduces the gap degree. For instance, the sentence in Example 3 could be translated from the English gloss as *Petr se rozhodl proti odmítnutí zítra v práci protestovat.*, rendering no gap at all. However, the position of the reflexive particle *se* is fairly rigid (the "second" position in the sentence) and constraints on topic-focus articulation often lead to a gap-1 structure. Forcing projective word order by e.g. CFG as Galley et al. (2006) do on the target side would lead to mildly disfluent output.

4 References

Ondřej Bojar and Martin Čmejrek. 2007. Mathematical Model of Tree Transformations. Project Euromatrix - Deliverable 3.2, ÚFAL, Charles University, December.

David Chiang and Kevin Knight. 2006. An Introduction to Synchronous Grammars. Part of a tutorial given at ACL 2006, `http://www.isi.edu/čhiang/papers/synchtut.pdf`, June.

Ralph Debusmann and Marco Kuhlmann. 2007. Dependency grammar: Classification and exploration. Project report (CHORUS, SFB 378).

Jason Eisner. 2003. Learning Non-Isomorphic Tree Mappings for Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Companion Volume*, pages 205–208, Sapporo, July.

Haim Gaifman. 1965. Dependency Systems and Phrase-Structure Systems. *Information and Control*, 8(3):304–337.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 961–968. Association for Computational Linguistics.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. 2006. Prague Dependency Treebank 2.0. LDC2006T01, ISBN: 1-58563-370-4.

T. Holan, V. Kuboň, K. Oliva, and M. Plátek. 1998. Two Useful Measures of Word Order Complexity. In A. Polguere and S. Kahane, editors, *Proceedings of the Coling '98 Workshop: Processing of Dependency-Based Grammars*, Montreal. University of Montreal.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163.

Aravind K. Joshi, K. Vijay Shanker, and David Weir. 1990. The Convergence of Mildly Context-Sensitive Grammar Formalisms. Technical Report MS-CIS-90-01, University of Pennsylvania Department of Computer and Information Science.

Tomasz Jurdziński, Friedrich Otto, František Mráz, and Martin Plátek. 2008. On the complexity of 2-monotone restarting automata. *Theor. Comp. Sys.*, 42(4):488–518.

Marco Kuhlmann and Mathias Möhl. 2007. Mildly context-sensitive dependency languages. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 160–167, Prague, Czech Republic, June. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT/EMNLP 2005*, October.

Joakim Nivre. 2005. Dependency Grammar and Dependency Parsing. Technical Report MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Martin Plátek. 2001. Two-way restarting automata and j-monotonicity. In *SOFSEM '01: Proceedings of the 28th Conference on Current Trends in Theory and Practice of Informatics Piestany*, pages 316–325, London, UK. Springer-Verlag.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 271–279. Association for Computational Linguistics.