The Simple Truth about Dependency and Phrase Structure Representations

An Opinion Piece

Owen Rambow
CCLS, Columbia University
New York, NY, USA
rambow@ccls.columbia.edu

Abstract

There are many misconceptions about dependency representations and phrase structure representations for syntax. They are partly due to terminological confusion, partly due to a lack of meta-scientific clarity about the roles of representations and linguistic theories. This opinion piece argues for a simple but clear view of syntactic representation.

1 Introduction

To the machine learning community, treebanks are just collections of data, like pixels with captions, structural and behavioral facts about genes, or observations about wild boar populations. In contrast, to us computational linguists, treebanks are not naturally occurring data at all: they are the result of a very complex annotation process. While the text that is annotated (usually) is naturally occurring, the annotation itself is already the result of a scientific activity. This opinion piece argues that the level of discourse about treebanks often found in our community does not reflect this fact (presumably due to the influence of the brute machine learning perspective). We, as a community of computational linguists, need to be very precise when talking about treebanks and syntactic representations in general.

So let's start with three very important concepts which we must always distinguish. The **representation type**: what type of mathematical object is used to represent syntactic facts? In this opinion piece, I only consider dependency trees (DTs) and phrase structure trees (PSTs) (Section 2). The represented **syntactic content**: the morphological and syntactic facts of the analyzed sentence (Section 3). The **syntactic theory**: it explains how syntactic content is represented in the chosen representation type (Section 4).

A crucial confusing factor is the fact that the terms *dependency* and *phrase structure* both have both a mathematical and a linguistic meaning. The mathematical meaning refers **representation types**. The linguistic meaning refers to **syntactic content**. I discuss this issue in Section 3. I discuss the issue of converting between DTs and PSTs in Section 5, as an example of how my proposed conceptualization of syntactic representation throws light on a computational problem.

This opinion piece will be a success if after reading it, the reader concludes that actually he or she knew this all along. In fact, this opinion piece does not advocate for a controversial position; its mission is to make its readers be more precise when talking about syntactic representations. This opinion piece is intentionally polemical for rhetorical reasons.

2 DTs and PSTs as Representation Types

Assume we have two disjoint symbol sets: a set of terminal symbols which contains the words of the language we are describing; and a set of nonterminal symbols. A **Dependency Tree** (DT) is a tree in which all nodes are labeled with words (elements of the set of terminal symbols) or empty strings. A **Phrase Structure Tree** (PST) is a tree in which all and only the leaf nodes are labeled with words or empty strings, and internal nodes are labeled with nonterminal symbols. There is nothing more to the

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 337–340,
Los Angeles, California, June 2010.

definitions. Trees of both types can have many other properties which are not part of the two definitions, and which do not follow from the definitions. I mention some such properties.

Unordered trees. DTs and PSTs can be ordered or unordered. For example, the Prague Theory (Sgall et al., 1986) uses unordered DTs at the deeper level of representation and ordered DTs at a more surfacy level. GPSG (Gazdar et al., 1985) uses unordered trees (or at any rate context-free rules whose right-hand side is ordered by a separate component of the grammar), as does current Chomskyan theory (the PST at spell-out may be unordered).

Empty categories. Empty categories can be empty pronouns, or traces, which are co-indexed with a word elsewhere in the tree. Empty pronouns are widely used in both DT- and PST-based representations. While most DT-based approaches do not use traces, Lombardo and Lesmo (1998) do; and while traces are commonly found in PST-based approaches, there are many that do not use them, such as the c-structure of LFG.

Discontinuous Constituents or **Non-Projectivity**. Both types of trees can be used with or without discontinuous constituents; PSTs are more likely to use traces to avoid discontinuous constituents, but linguistic proposals for PSTs with discontinuous constituents have been made (work by McCawley, or (Becker et al., 1991)).

Labeled Arcs. In DTs, arcs often have labels; arcs in PSTs usually do not, but we can of course label PST arcs as well, as is done in the German TIGER corpus.I note that in both DTs and PSTs we can represent the arc label as a feature on the daughter node, or as a separate node.

3 Syntactic Content

While there is lots of disagreement about the proper **representation type** for syntax, there is actually a broad consensus among theoretical and descriptive syntacticians of all persuasions about the range of syntactic phenomena that *exist*. What exactly is this content, then? It is not a theory-neutral representation of syntax (Section 4). Rather, it is the *empirical matter* which linguistic theory attempts to represent or explain. We cannot represent it without a theory, but we can refer to it without a theory, using names such as *control constructions* or *transitive verb*. In the same manner, we use the word *light* and physicists will agree on what the phenomenon is, but we cannot represent light within a theory without choosing a representation as either particles or wave.

Note that in linguistics, the terms *dependency* and *phrase structure* refer to **syntactic content**, i.e., syntactic facts we can represent. **Syntactic dependency** is direct relation between words. Usually, this relation is labeled (or typed), and is identical to (or subsumes) the notion of **grammatical function**, which covers relations such as SUBJECT, OBJECT, TEMPORAL-ADJUNCT and so forth. **Syntactic phrase structure**, also known as **syntactic constituency structure** is recursive representation using sets of one or more linguistic units (words and empty strings), such that at each level, each set (constituent) acts as a unit syntactically. Linguistic phrase structure is most conveniently expressed in a phrase structure tree, while linguistic dependency is most conveniently expressed in a dependency tree. However, we can express the same content in either type of tree! For example, the English Penn Treebank (PTB) encodes the predicate-argument structure of English using structural conventions and special nonterminal labels ("dashtags"), such as NP-SBJ. And a dependency tree represents constituency: each node can be interpreted both as a preterminal node (X^0) and as a node heading a constituent containing all terminals included in the subtree it heads (the XP). Of course, what is more complex to encode in a DT are intermediate projections, such as VP. I leave a fuller discussion aside for lack of space, but I claim that the syntactic content which is expressed in intermediate projections can also be expressed in a DT, through the use of features and arc labels.

4 Syntactic Theory

The choice of representation type does not determine the representation for a given sentence. This is obvious, but it needs to be repeated; I have heard "What is the DT for this sentence?" one too many times. There are many possible DTs and PSTs, proposed by serious syntacticians, for even simple sen-

tences, even when the syntacticians agree on what the syntactic content (a transitive verb with SVO order, for example) of the analysis should be! What is going on?

In order to make sense of this, we need a third player in addition to the **representation type** and the **content**. This is the **syntactic theory**. A linguistic theory chooses a representation type and then defines a coherent mapping for a well-defined set of content to the chosen representation type. Here, "coherent representation" means that the different choices made for conceptually independent content are also representationally independent, so that we can compose representational choices. Note that a theory can decide to omit some content; for example, we can have a theory which does not distinguish raising from control (the English PTB does not).

There are different types of syntactic theories. A **descriptive theory** is an account of the syntax of one language. Examples of descriptive grammars include works such as Quirk for English, or the annotation manuals of monolingual treebanks, such as (Marcus et al., 1994; Maamouri et al., 2003). The annotation manual serves two purposes: it tells the annotators how to represent a syntactic phenomenon, and it tells the users of the treebank (us!) how to interpret the annotation. A treebank without manual is meaningless. And an arborescent structure does not mean the same thing in all treebanks (for example, a "flat NP" indicates an unannotated constituent in the English ATB but a fully annotated construction in the Arabic Treebank is).

An **explanatory theory** is a theory which attempts to account for the syntax of all languages, for example by reducing their diversity to a set of principles and finite-valued parameters. Linguistic theories (and explanatory theories in particular) often take the form of a one-to-many mapping from a simple representation of syntactic dependency (predicate-argument structure) to a structural representation that determines surface word order. The linguistic theory itself is formulated as a (computational) device that relates the deeper level to the more surfacy level. LFG has a very pure expression of this approach, with the deeper level expressed using a DT (actually, dependency directed acyclic graphs, but the distinction is not relevant here), and the surfacy level expressed using a PST. But the Chomskyan approaches fit the same paradigm, as do many other theories of syntax.

Therefore, there is no theory-neutral representation of a sentence or a set of sentences, because *every* representation needs a theory for us to extract its meaning! Often what is meant by "theory-neutral tree" is a tree which is interpreted using some notion of consensus theory, perhaps a stripped-down representation which omits much content for which there is no consensus on how to represent it.

5 Converting Between DTs and PSTs

Converting a set of DS annotations to PS or *vice versa* means that we want to obtain a representation which expresses exactly the same **content**. This is frequently done these days as interest in dependency parsing grows but many languages only have PS treebanks. However, this process is often not understood.

To start, I observe that uninterpreted structures (i.e., structures without a syntactic theory, or trees from a treebank without a manual) cannot be converted from or into, as we do not know what they mean and we cannot know if we are preserving the same content or not.

Now, my central claim about the possibility of automatically converting between PSTs and DTs is the following. If we have an interpretation for the source representation and the goal representation (as we must in order for this task to be meaningful), then we can convert any facts that are represented in the source structure, and we cannot convert any facts that are not represented in the source structure. It is that simple. If we are converting from a source which contains less information than the target, then we cannot succeed. For example, if we are converting from a PS treebank that does not distinguish particles from prepositions to a DS treebank that does, then we will fail. General claims about the possibility of conversion ("it is easier to convert PS to DS than DS to PS") are therefore *meaningless*. It only matters **what** is represented, not **how** it is represented.

There is, however, no guarantee that there is a *simple* algorithm for conversion, such as a parametrized

head percolation algorithm passed down from researcher to researcher like a sorcerer's incantation. In general, if the two representations are independently devised and both are linguistically motivated, then we have no reason to believe that the conversion can be done using a specific simple approach, or using conversion rules which have some fixed property (say, the depth of the trees in the rules templates). In the general case, the only way to write an automatic converter between two representations is to study the two annotation manuals and to create a case-by-case converter, covering all linguistic phenomena represented in the target representation.

Machine learning-based conversion (for example, (Xia and Palmer, 2001)) is an interesting exercise, but it does not give us any general insights into dependency or phrase structure. Suppose the source contains all the information that the target should contain. Then if machine learning-based conversion fails or does not perform completely correctly, the exercise merely shows that the machine learning is not adequate. Now suppose that the source does not contain all the information that the target should contain. Then no fancy machine learning can ever provide a completely correct conversion. Also, note that unlike, for example, parsers which are based on machine learning and which learn about a natural phenomenon (language use), machine learning of conversion merely learns an artificial phenomenon: the relation between the two syntactic theories in question, which are created by researchers. (Of course, in practice, machine learning of automatic conversion between DT to PSTs is useful.)

6 Conclusion

I have argued that when talking about dependency and phrase structure representations, one should always distinguish the type of representation (dependency or phrase structure) from the content of the representation, and one needs to understand (and make explicit if it is implicit) the linguistic theory that relates content to representation. Machine learning researchers have the luxury of treating syntactic representations as mere fodder for their mills; we as computational linguists do not, since this is our area of expertise.

Acknowledgments

I would like to thank my colleagues on the Hindi-Urdu treebank project (Bhatt et al., 2009) (NSF grant CNS-0751089) for spirited discussions about the issues discussed here. I would like to thank Sylvain Kahane, Yoav Goldberg, and Joakim Nivre for comments that have helped me improve this paper. The expressed opinions have been influenced by far too many people to thank individually here.

References

Tilman Becker, Aravind Joshi, and Owen Rambow. 1991. Long distance scrambling and tree adjoining grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, pages 21–26. ACL.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189, Suntec, Singapore.

Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Mass.

Vincenzo Lombardo and Leonardo Lesmo. 1998. Formal aspects and parsing issue of dependency theory. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 787–793, Montréal, Canada.

Mohamed Maamouri, Ann Bies, Hubert Jin, and Tim Buckwalter. 2003. Arabic treebank: Part 1 v 2.0. Distributed by the Linguistic Data Consortium. LDC Catalog No.: LDC2003T06.

Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The Penn Arabic Treebank: Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press, New York.

P. Sgall, E. Hajičová, and J. Panevová. 1986. *The meaning of the sentence and its semantic and pragmatic aspects*. Reidel, Dordrecht.

Fei Xia and Martha Palmer. 2001. Converting dependency structure to phrase structures. In *hlt2001*, pages 61–65.