

Converting Treebank Annotations to Language Neutral Syntax

Richard Campbell and Eric Ringger

Microsoft Research
One Microsoft Way
Redmond, Washington 98052 USA
{richcamp, ringger}@microsoft.com

Abstract

We describe the automatic conversion of English Penn Treebank (PTB) annotations into Language Neutral Syntax (LNS) (Campbell and Suzuki, 2002a,b). In this paper, we describe LNS and why it is useful, describe the conversion algorithm, present an evaluation of the conversion, and discuss some uses of the converted annotations and the potential for extending the coverage to other languages. The work described here is in the spirit of other automatic re-annotations of PTB trees (e.g. Frank, 2000 and Meyers, 2001), but differs in the nature of the output.

Introduction

In this paper we describe a method of converting annotated trees in the Penn Treebank (PTB) (Marcus et al., 1993) into Language Neutral Syntax (LNS) representations, a representation system for natural language sentences developed at Microsoft Research (Campbell and Suzuki, 2002a,b).

We begin by giving a brief description of LNS; for a detailed description, please see the references given above. We then describe the conversion method itself, along with an evaluation of the conversion. We then discuss possible uses of the new annotation. We close with a discussion of scalability and possible future work.

Language Neutral Syntax (LNS)

An LNS tree is a representation of a natural language sentence that is semantically motivated and abstract, yet sufficiently concrete to effectively and robustly mediate between languages and between applications. LNS is semantically motivated in that it represents the logical

arrangement of the parts of a sentence, normalizing such features as word order, function words and inflectional morphology to a language-neutral form. For example, word order in English might indicate logical scope or grammatical function; in LNS, logical scope is represented explicitly as hierarchical order, and grammatical functions are represented by labels on the arcs connecting nodes to their parents; word order per se is neither present nor represented. Similarly, information carried by inflectional morphology and function words is represented in LNS by separate nodes (e.g. tense, negation), or by binary features (e.g. definiteness, aspect). This results in a very different constituency from what occurs in the PTB representation, one which is less dependent on the particulars of English surface syntax and more reflective of logical relations, including scope and (deep) functional relations, which can be assumed to be more constant from one language to another.

As an example of an LNS tree, consider the (made-up) sentence *None of the largest American companies are being audited yet*; the LNS of which is given in Figure 1 (some details omitted).

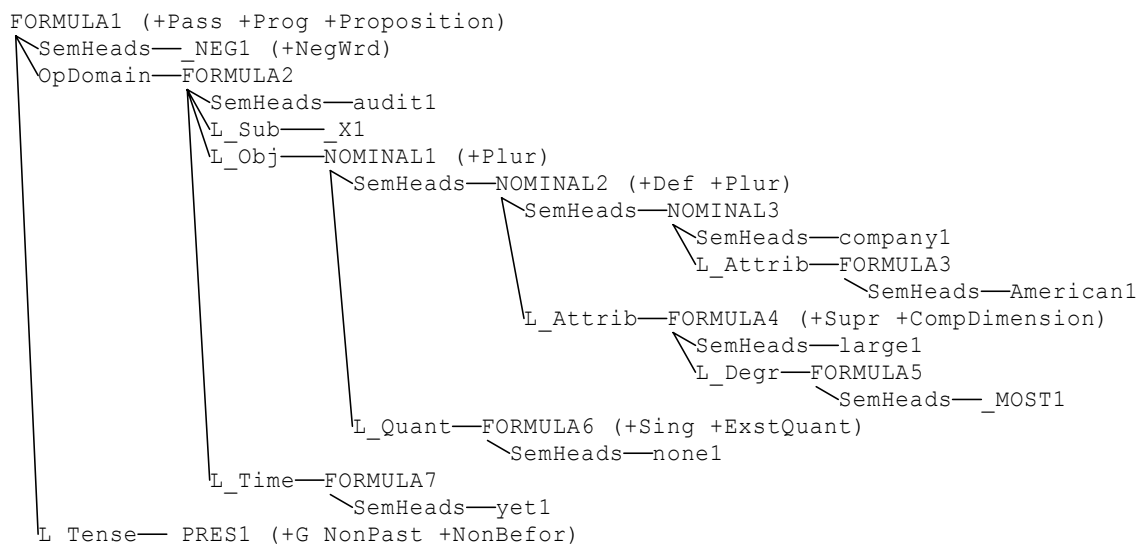


Figure 1: LNS for the sentence *None of the largest American companies are being audited yet*.

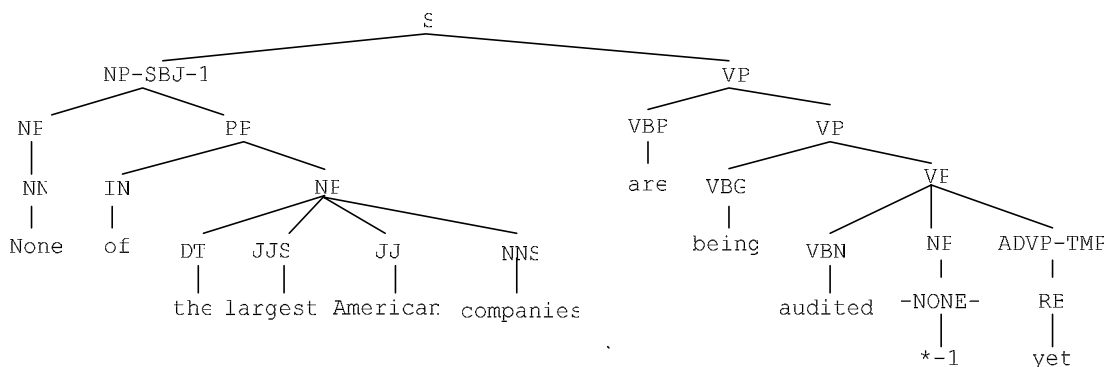


Figure 2: PTB-style tree for the sentence *None of the largest American companies are being audited yet.*

The LNS diagram is read as follows. The root is in the upper left; nodes are labeled either NOMINAL, FORMULA, or, in the case of terminal nodes, with a lemma (such as *company*) or abstract symbol (such as *NEG* or *MOST*); all node names include an integer index to distinguish it from other nodes of the same label. Arcs connecting nodes to their parent are labeled with a relation type, such as “L_Sub” (logical subject) or “SemHeads” (semantic head). A node may have zero or more binary features, indicated in parentheses to the right.

Comparing the LNS in Figure 1 to the PTB-style tree for the same sentence, shown in Figure 2 (some details omitted), we can use this example to illustrate several features of LNS that distinguish it from a PTB tree, as well as from other logical-form-type representations such as QLF (Alshawi et al., 1991) and F-Structure (Bresnan, 2001). In general, logical relations indicated explicitly in the LNS are indicated either indirectly or not at all in the PTB tree.

At a gross level, the PTB tree for this sentence has a relatively flat NP and a complex, branching VP, while the LNS tree has a complex branching NP and a flat predicate. The PTB annotation for this sentence has a complex VP, due to the presence of two auxiliary verbs, each of which heads its own VP. There is no corresponding complex predicate in the LNS for this sentence: instead, the voice and aspect information carried by those auxiliaries is represented by the features [+Pass +Prog] on FORMULA1, the node corresponding to the sentence; the auxiliaries themselves are not represented. Second, the subject NP has a relatively flat PTB structure, but its corresponding LNS node (NOMINAL1 in Figure 1) has a branching structure indicating the relative logical scope of the quantifier and the two adjectives. Third, features incorporated (either lexically or morphologically) into other words in the PTB tree are represented explicitly in LNS: the present tense incorporated into the auxiliary *are* is represented as a distinct Tense node, *_PRES1* (Campbell et al., 2002), and the negation inherent in the determiner *no* is represented as an abstract negative operator *_NEG*, taking scope over the rest of the sentence; FORMULA6, the node corresponding to *none* in the LNS, has the feature [+ExstQuant], indicating that it is to be interpreted as an existential quantifier.

Some aspects of the LNS derive more or less directly from the PTB annotation. For example, the fact that FORMULA7 is in the L_Time (logical time) relation with its parent straightforwardly reflects the ‘TMP’ functional

role tag on ADVP in the PTB tree. Other aspects derive from the PTB tree, but only indirectly. For example, in the PTB tree, one can tell that the sentence is passive because there is a VBN whose corresponding phrase is preceded by a form of the copula, and followed by an empty category NP whose head is * and is coindexed with the subject NP. In LNS, the subject NP, NOMINAL1, is directly represented as the logical object of the sentence (the logical subject being unspecified), and FORMULA1, representing the sentence itself, has the feature [+Pass].

But many aspects of the LNS structure are not represented in the PTB at all. For example, while the subject NP has some internal structure in the PTB, reflecting the fact that it is a partitive construction, there is no internal structure assigned to the string *the largest American companies* beyond word-level tags. The LNS, on the other hand, represents the fact that not only does *none* have scope over the rest of the NP, but that *largest* has logical scope over *American companies*. The internal structure of NOMINAL1 is not derived from the PTB annotation, but must be deduced in large part from general principles. The most difficult problems of conversion involve features not present in the PTB, but required in a fully specified LNS.

Conversion of Treebank structures to LNS

In this section we describe the conversion algorithm and the evaluation of the conversion.

The conversion algorithm

PTB annotations are deserialized into trees that are isomorphic to the original annotations. Next, heads are labeled using Eugene Charniak’s head-labeling rules (used for training the parser described in (Charniak,2000)). Additional features on each terminal are computed by dictionary lookup and morphological analysis, constrained by the PTB part-of-speech label. These features are propagated from head to parent.

Subsequent processing fleshes out the trees in various ways. For example, some preterminal nodes (such as JJS in Figure 2) have no phrasal node associated with them; these are fleshed out at this stage to ensure that every preterminal which corresponds to an LNS node has a phrasal projection. Also at this stage, complex NPs, including coordinate NPs and compounds, are assigned internal structure.

An LNS node is then created for each node in the PTB tree that is to have a corresponding LNS node. This

includes lexical preterminals and their phrasal projections, but excludes many function words such as articles and auxiliaries. The LNS nodes corresponding to preterminals are labeled with the lemma of the corresponding terminal node. A preliminary dependency structure is stitched together using these newly created nodes, and then the basic grammatical function relations, such as `L_Sub` and `L_Attrib`, are assigned, some based on evidence gleaned directly from the semantic role tags of the PTB, as mentioned above.

On a separate pass through the tree (now an intermediate LNS tree), abstract nodes, such as the negative operator `_NEG` illustrated in Figure 1, are created, and logical operators and modifiers are assigned scope, using functions originally designed to work within the NLPWin analysis system developed at Microsoft Research (Heidorn, 2000). Sentence-level logical operators, including sentential negation and modality operators, are typically assigned scope according to their linear order in the string. Some lexically specific exceptions, for example the fact that negation has scope over the modal in the sequence *can not*, are handled here as well.

A separate function, incorporating the language-independent algorithm described in Campbell (2002), assigns scope to modifiers within NP. This algorithm takes into account not only linear order, but also modifier type: quantifiers and quantifier-like adjectives are assigned wider scope than comparatives and superlatives, which in turn are assigned wider scope than plain adjectives. Post-modifiers, such as English relative clauses, are typically assigned wider scope than simple, premodifying adjectives. The reader is referred to Campbell (2002) for further details.

Evaluation

To evaluate the conversion process, we created a reference set of LNS trees by randomly selecting 108 sentences from section 11 of the PTB, and correcting the converter’s output using a tree manipulation tool. Each tree was then stored as a set of ordered triples, consisting of an LNS node, a relation name, and the LNS node that is the value of that relation. Each node `N` in the LNS tree is in turn identified as a pair consisting of `N`’s head word and an integer indicating the number of nodes in the path from the head of `N` to `N`, not counting the head itself.

As an example, the triple representing the relationship between `NOMINAL2` and `FORMULA4` in Figure 1 would be represented as follows:

company-2::L_Attrib::large-1

‘company-2’ indicates the node in the projection path of *company* that is two levels up the tree; i.e., `NOMINAL2`; similarly, ‘large-1’ indicates the first projection of *large*, i.e., `FORMULA4`. The representation above indicates that these two nodes are in the `L_Attrib` relation. An LNS tree is uniquely identified by the full set of LNS triples of this sort.

We then ran the sentences through the converter, storing the triples as above, and compared the result to the reference, counting only exact matches as correct. The results are given in Table 1.

Precision	Recall	F ₁
92.4	93.7	93.0

Table 1: Accuracy of conversion from PTB trees to LNS trees, expressed as percentage.

Precision (P) here is the percentage of LNS triples proposed by the converter that are in the reference set for that sentence. Recall (R) is the percentage of LNS triples in the reference set that are proposed for that sentence by the converter; and F₁ is balanced f-measure, i.e., $2PR/(P+R)$.

The figures in Table 1 are understandably high, given that the reference set was constructed by correcting the converter’s output; also, there are aspects of LNS (e.g. features of nodes, long-distance relations among nodes) that are not part of the basic tree, and hence not evaluated in this method. Nevertheless, the results indicate that the conversion is reasonably accurate.

Uses of the converted Treebank

Given its language-neutral character, LNS serves as a representation from which other application-specific semantic representations can be derived by language-neutral functions. For example, a dependency graph representing basic predicate-argument structure and other lexical dependencies is derived automatically from LNS by a simple language-independent function; an example is shown in Figure 3.

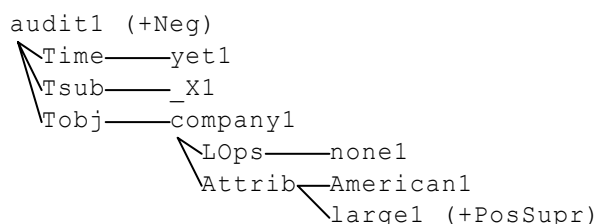


Figure 3: Predicate-argument structure of *None of the largest American companies are being audited yet.*

Predicate-argument structures, derived from the LNSs produced by NLPWin, are used as the transfer representation in the MSR-MT system (Richardson et al., 2001). Other applications make use of other representations derived from LNS. These include work on extraction of bilingual collocations (Wu and Zhou, 2003), multi-document summarization, automatic quiz generation, sentence classification, and document classification. Currently all these applications use NLPWin to produce the LNS representations from which the application-specific representation is derived; but given the ability to convert any PTB-style trees to LNS, the production of LNS-style annotation is no longer limited to the use of the NLPWin parser.

The ability to produce predicate-argument structures automatically and reliably from the Treebank enables a kind of virtual predicate-argument structure-bank, which could be an alternative or supplement to the existing, hand-annotated Propbank (Kingsbury and Palmer, 2002).

Being able to produce such representations reliably from the Treebank also allows one to evaluate the accuracy of systems that produce such representations.

For example, if an analyzer produces predicate argument structures similar to that shown in Figure 3, one could measure its accuracy by comparing to the structures produced from the Treebank itself.

Conclusion: Scalability and future work

Aside from features extracted from a dictionary (e.g. the fact that possessive *my* is based on the pronoun *I*, or that the determiner *no* is negative), the converter makes no mention of specific English words other than (a) determiners and auxiliaries realized in LNS as binary features or abstract (e.g. tense) nodes (e.g. *the*, *be*, *have*), (b) words that are not realized in LNS at all (e.g. pleonastic pronouns, auxiliary *do*). This fact, together with the language-neutral character of LNS, indicates that the converter can be adapted to other languages for which similar treebanks exist, such as Chinese (Xia et al., 2000), experiments we hope to undertake in the future.

The converter described here allows for the conversion of the entire PTB into LNS, which in turn will permit automatic conversion to semantic representations derived from LNS, such as predicate-argument structure.

References

- Alshawi, H., D. Carter, M. Rayner and B. Gambäck. 1991. Translation by Quasi Logical Form transfer. In *Proceedings of ACL 29*. 161-168.
- Bresnan, J. 2001. *Lexical-Functional Syntax*. Malden, MA and Oxford: Blackwell.
- Campbell, R. 2002. Computation of modifier scope in NP by a language-neutral method. In *Proceedings of COLING 2002*, Taipei.
- Campbell, R., T. Aikawa, Z. Jiang, C. Lozano, M. Melero and A. Wu. 2002. A language neutral representation of temporal information. In *LREC 2002 Workshop Proceedings: Annotation Standards for Temporal Information in Natural Language*. 13-21.
- Campbell, R. and H. Suzuki. 2002a. Language-neutral representation of syntactic structure. In *SCANALU 2002: Proceedings of the First International Workshop on Scalable Natural Language Understanding*, Heidelberg.
- Campbell, R. and H. Suzuki. 2002b. *Language Neutral Syntax: An overview*. Technical Report MSR-TR-2002-76. Microsoft Research, Redmond, WA.
- Charniak, E. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL 2000*.
- Frank, A. 2000. Automatic F-structure annotation of treebank trees. In *Proceedings of LFG00 Conference*.
- Heidorn, G. 2000. Intelligent writing assistance. In R. Dale, H. Moisl, and H. Somers, eds., *Handbook of Natural Language Processing*, Marcel Dekker.
- Kingsbury, P. and M. Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, 1974-1981. Las Palmas de Gran Canaria.
- Meyers, A., R. Grishman, M. Kosaka, & S. Zhao. 2001. Covering treebanks with GLARF. In *ACL/EACL Workshop on Sharing Tools for Research and Education*.
- Richardson S., W. Dolan, A. Menezes and J. Pinkham. 2001. Achieving commercial-quality translation with example-based methods. In *Proceedings of the VIIIth MT summit*. 293-298.
- Xia, F., M. Palmer, N. Xue, M.E. Okurowski, J. Kovarik, F.-D. Chiou, S. Huang, A. Kroch, and M. Marcus. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, Athens.
- Wu, Hua and Ming Zhou. 2003. Synonymous collocation extraction using translation information. In *Proceedings of ACL 41*.