Comparing and Integrating Tree Adjoining Grammars

Fei Xia, Martha Palmer

Department of Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104, USA
{fxia,mpalmer}@linc.cis.upenn.edu

Abstract

Grammars are core elements of many NLP applications. Grammars can be developed in two ways: built by hand or extracted from corpora. In this paper, we compare a hand-crafted grammar with a Treebank grammar. We contend that recognizing substructures of the grammars' basic units is necessary not only because it allows grammars to be compared at a higher level, but also because it provides the building blocks for consistent and efficient integration of the grammars.

1. Introduction

A Lexicalized Tree Adjoining Grammar (LTAG) is a core element of many NLP applications. It often has hundreds of elementary trees (*etrees*), which can either be built by hand (hand-crafted grammars), or extracted from annotated corpora (Treebank grammars). Hand-crafted grammars have rich representations (such as feature structures), and tend to be more precise, but they take a long time to build and their coverage on naturally-occurring data is hard to determine. In addition, they lack statistical information which is crucial for statistical parsers. Treebank grammars, on the other hand, require little human effort (Xia, 1999; Chen & Vijay-Shanker, 2000) to build, once the Treebank has been created. They have rich statistical information and will cover at least the corpora from which the grammars are extracted. However, Treebank grammars are noise-prone because of annotation errors in the corpora and they also lack features and semantic information which are rarely represented in the corpora. It would be ideal if we could combine the strengths of both types of grammar. As a first step towards addressing this issue, in this paper we compare a hand-crafted grammar with a Treebank grammar and propose a way of integrating them to produce new grammars.

2. Two grammars

The two LTAGs that we compare are the XTAG English grammar (XTAG-Group, 1995) and a grammar extracted from Penn English Treebank. The **XTAG** grammar has 1004 tree templates.[1] The **Treebank** grammar that we use in this paper is extracted from the Penn English Treebank II (Marcus *et al.*, 1994) using the extraction algorithm described in (Xia, 1999). The extracted grammar has 3072 templates.

For lack of space, we will not describe the extraction algorithm, other than pointing out that by design all the *etrees* extracted from the Treebank fall into one of three types according to the relations between the anchor of the *etree* and other nodes in the tree, as shown in Figure 1. Figure 2 shows a bracketed sentence from the Penn Treebank. From that sentence, five *etrees* are extracted by the algorithm, as shown in Figure 3.

[1] If we remove the anchor(s) from *etrees*, we get *tree* templates. Each template indicates where the anchor(s) of that *etree* will be instantiated.

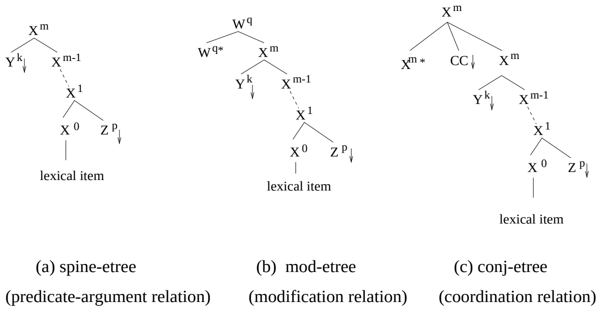


Figure 1: Forms of extracted *etrees*

```
(SBAR (WHNP-1 (WP who) )
       (S (NP-SBJ (-NONE- *T*-1) )
          (VP (VBD worried)
              (PP-CLR (IN about)
                      (NP (DT the)
                          (NN flood) ))))))
```

Figure 2: An example from the Treebank

3. Comparing two grammars

To compare the grammars, we need to find out how many trees in one grammar *match* trees in the other grammar. We define two types of matching : t-match and c-match. From now on, we use XTAG and Ext-G to stand for the XTAG grammar and the extracted grammar respectively.

3.1. t-match

We call two trees *t-match* (*t* for *tree*) if they are identical barring the type of information present only in one grammar, such as feature structures and subscripts[2] in XTAG and frequency information in Ext-G. In Figure 4, XTAG tree 4(a) and 4(b) t-match Ext-G tree 4(c).

XTAG also differs from Ext-G in that XTAG includes multi-anchor trees to handle idioms (Figure 5(a)), light verbs (Figure 5(b)) and so on. In each of these cases,

[2]The subscripts on the nodes mark the same semantic arguments in related subcategorization frames.

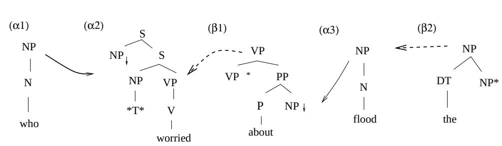


Figure 3: The extracted *Etrees*

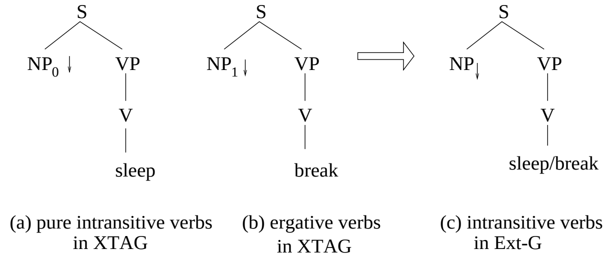


Figure 4: An example of *t-match*

the multi-anchors form the predicate. These trees are the same as the spine-etree in Figure 1(a) except that some nodes of the XTAG trees (e.g. NP_1 in Figure 5(a) and its counterpart Z_p in Figure 1) are expanded. By having multi-anchors, each tree can be associated with semantic representations directly (as shown in in Figure 5), which is an advantage of LTAG formalism. Ext-G does not have multi-anchor trees because semantics is not marked in the Treebank and consequentially the extraction algorithm can not distinguish idiomatic meanings from literal meanings. Two trees are called *t-match without expansions* if they *t-match* after the expanded part is removed from the XTAG trees. Figure 5 is such an example.

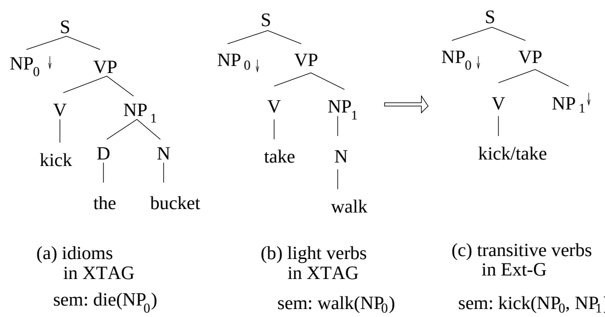
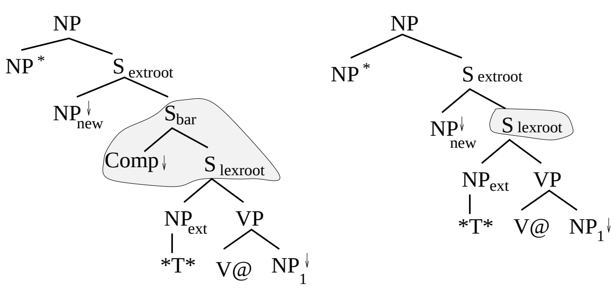


Figure 5: *t-match w/o expansion*

3.2. c-match

t-match requires two trees to have exactly the same structure, therefore, it does not tolerate minor differences between the trees. For instance, in XTAG, relative pronouns such as *which* and the complementizer *that* occupy distinct positions in the *etree* for relative clauses, whereas the Penn Treebank treats both as pronouns and therefore they occupy the same position in Ext-G, as shown in Figure 6. Because the circled subtrees will occur in every tree for relative clauses and wh-movement, all these trees will not *t-match* their counterparts in the other grammar. Nevertheless, the two trees share the same subcategorization frame *(NP V NP)*, the same subcategorization chain[3] $S \rightarrow VP \rightarrow V$ and the same modification pair (NP, S). To capture this kind of similarity, we decompose a mod-etree into a tuple of (subcat frame, subcat chain, modification pair). Similarly, a spine-etree is decomposed into a (subcat frame, subcat chain) pair, and a conj-etree into (subcat frame, subcat chain, coordination sequence). Two *etrees* are said to *c-match* (*c* for *component*) if they are decomposed into the same tuples. According to this definition, the two trees in Figure 6 *c-match*.



(a) in XTAG (b) in Ext-G

Figure 6: Relative clause trees

3.3. Comparison results

So far, we have defined several types of matching. Table 1 lists the numbers of tree templates[4] in one grammar that match some tree templates in the other grammar.[5] The last row lists the frequencies of the matched Ext-G templates. For instance, the fourth column says 496 templates in XTAG *match* 189 templates in Ext-G, and these 189 templates account for 57.1% of the template tokens in the Penn Treebank. If we decompose templates into components as mentioned in Section 3.2, the components that are shared by both grammars will cover 82.9% of all the component occurrences, as shown in Table 2. Templates in Ext-G are missing from the XTAG grammar for one or more of the following reasons:

T1: incorrect templates in Ext-G These templates result from Treebank annotation errors. Our extraction algorithm has a filter that detects implausible templates in Ext-G by decomposing a template into parts and checking each part against several small hand-crafted tables. The filter marks 2299 templates in Ext-G as implausible and they account for 5.2% of the template tokens in the Treebank.

T2: conj-etrees in XTAG Most conj-etrees in XTAG are generated on-the-fly while parsing (Sarkar & Joshi, 1996), and are not part of the 1004 templates. Therefore, many of the *conj-etrees* in Ext-G, which account for 2.8% of the template tokens in the Treebank, do not match any templates in XTAG.

T3: different analyses XTAG and Ext-G often choose different analyses for the same phenomenon. For example, the two grammars treat reduced relative clauses differently. [6]

T4: missing constructions in XTAG Some constructions such as the unlike coordination phrase (UCP) in the Treebank are not covered in XTAG.[7]

[3]A *subcategorization chain* is a subsequence of the spine in a spine-etree where each node on the chain is a parent of some argument(s) in the subcategorization frame. The nodes on a subcategorization chain roughly correspond to various lexical projections in GB-theory.

[4]We compare tree templates, not trees, in the two grammars because we are focusing on general syntactic structure.

[5]If a template in one grammar matches several templates in the other grammar and the match types are different, we label it with the strongest match type.

[6]Also, in XTAG, adjectives and nouns directly modify nouns, whereas in Ext-G, they modify noun phrases. These two pairs – (N, NP) and (A, NP) – account for 26.6% of the modification pairs in the Treebank, explaining XTAG's lack of coverage (53.1%) of the modification pair occurrences in the Treebank.

[7]The difference between matched templates (58.0%) and matched components (82.9%) imply that some combinations of components are missing from XTAG, The problem is very common for hand-crafted grammars because the the redundancy among trees in the grammar makes it very hard

	t-match	t-match w/o expansion	c-match	subtotal	conj-etree templates	no-match	total
XTAG	73	107	316	496(49.4%)	39	469	1004
Ext-G	59	5	125	189(6.15%)	411	2472	3072
frequency	53.9%	0.5%	2.7%	57.1%	2.8%	40.1%	100%

Table 1: Numbers of templates that match and their frequencies

	subcat chains	subcat frames	modification pairs	coordination pairs	total
in XTAG	44	115	72	25	256
in Ext-G	471	507	309	53	1340
matched types	35	45	31	10	121
matched tokens	977,218	954,776	357,563	22,937	2,312,494
frequency	93.7%	91.6%	53.1%	77.7%	82.9%

Table 2: Numbers of components in the two grammars

3.4. Integrating the two grammars

Simply taking the union of the two template sets will only yield a more noisy and inconsistent grammar. Our method has several steps: First, starting from Table 2, use the plausibility filter to automatically rule out all of the implausible components in XTAG and Ext-G, then integrate the remaining plausible components into a new set, one for each type of component (such as subcat frames, subcat chains, etc.). Next, generate a new grammar from the component sets using various grammar development tools such as Metarules(Becker, 1994) or LexOrg(Xia *et al.*, 1998). The new grammar will be of high quality and have good coverage of the Treebank.

4. Conclusion

In this paper, we compare the XTAG grammar with the Penn Treebank grammar and propose a way of integrating them in order to derive a new grammar which has the strength of both. We believe that recognizing components of elementary trees in the two grammars is necessary because it not only allows the grammars to be compared at a more fine-grained level, but also provides the building blocks for integrating the grammars in a consistent and efficient way.

to maintain the grammar by hand. Various tools to semi-automatically generate templates (Becker, 1994; Candito, 1996; Xia *et al.*, 1998) could alleviate the problem.

References

BECKER T. (1994). Patterns in metarules. In *Proceedings of the 3rd International Workshop on TAG and Related Frameworks(TAG+3)*, Paris, France.

CANDITO M.-H. (1996). A principle-based hierarchical representation of ltags. In *Proceedings of COLING-96*, Copenhagen, Denmark.

CHEN J. & VIJAY-SHANKER K. (2000). Automated extraction of tags from the penn treebank. In *6th International Workshop on Parsing Technologies (IWPT 2000), Italy.*

MARCUS M., KIM G., MARCINKIEWICZ M. A. *et al.* (1994). The Penn Treebank: annotating predicate argument structure. In *Proc of ARPA speech and Natural language workshop.*

SARKAR A. & JOSHI A. (1996). Coordination in Tree Adjoining Grammars: Formalization and Implementation. In *Proceedings of the 18th COLING*, Copenhagen, Denmark.

XIA F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Proc. of NLPRS-99*, Beijing, China.

XIA F., PALMER M., VIJAY-SHANKER K. & ROSENZWEIG J. (1998). Consistent Grammer Development Using Partial-tree Descriptions for Lexicalized Tree-Adjoining Grammar. In *Proc. of tag+4.*

XTAG-GROUP T. (1995). *A Lexicalized Tree Adjoining Grammar for English.* Technical Report IRCS 95-03, University of Pennsylvania.