Nondeterministic LTAG Derivation Tree Extraction

Libin Shen

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
`libin@linc.cis.upenn.edu`

Abstract

In this paper we introduce a naive algorithm for nondeterminisctic LTAG derivation tree extraction from the Penn Treebank and the Proposition Bank. This algorithm is used in the EM models of LTAG Treebank Induction reported in (Shen and Joshi, 2004). Given the trees in the Penn Treebank with PropBank tags, this algorithm generates shared structures that allow efficient dynamic programming in the EM models.

1 Introduction

In recent years, the statistical approach has been successfully used in natural language processing (NLP). No matter which statistical model people use, a generative model or statistical machine learning, large corpora are always needed to train the models. For example, after the introduction of the Penn Treebank (PTB) (Marcus et al., 1994), a serial of improvements has been achieved on natural language parsing and shallow parsing tasks. In the field of Lexicalized Tree Adjoining Grammar (LTAG), the statistical approach has also been successfully employed in many LTAG-based NLP tasks, such as LTAG parsing (Chiang, 2000; Shen et al., 2003) and Supertagging (Joshi and Srinivas, 1994).

However, the lack of very large corpora based on LTAG prevents the statistical approach from being widely used in the field of LTAG. As we know, very large corpora are crucial to statistical NLP. In previous works, people managed to induce LTAG style grammars and LTAG based corpora from the PTB, and use them in their applications.

Joshi and Srinivas (1994) first implemented a supertag corpus by extracting it from the PTB, using heuristic rules. Due to various limitations of this system, extracted supertags of the words in a sentence cannot always be successfully put together. Xia (2001) and Chen (2001) described deterministic systems that extract LTAG-style grammars from PTB. In their systems, the *head percolation table* (Magerman, 1995) and the PTB functional tags were used to solve ambiguities in extraction. Chiang (2000) reported a similar method to extract an LTAG like treebank from PTB, and used it in a statistical parser. Shen et al. (2003) employed a similar technique to induce an LTAG treebank, to be used in a parse reranking system.

In these LTAG grammar and treebank induction systems, deterministic rules were used to solve the ambiguities in the elementary tree extraction process. However, it is clear that deterministic rules are not enough to solve ambiguity in extraction, especially in the case of the argument-adjunct distinction (Paola and Leybold, 2001).

2 A Statistical Model

In (Shen and Joshi, 2004), we have proposed a statistical model for LTAG Treebank induction using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The EM Algorithm is a general iterative method of search to find the maximum-likelihood estimate of the parameters of the hidden data from the observed data.

If we take the PTB as the observed data, then the LTAG derivation trees for the PTB trees can be treated as the hidden data. Then our goal is the find out the hidden structures, or LTAG derivation trees, with maximum-likelihood. Similar idea was previously employed in (Chiang and Bikel, 2002) in statistical parsing.

In (Shen and Joshi, 2004), several EM models were proposed for LTAG Treebank induction. In these models, linguistic knowledge is used to overcome EM's weakness that EM cannot guarantee to find a global optimum. By using linguistic knowledge, we can not only start the EM iteration from the point close to the global optimum in the first iteration, but also limit the search space of hidden derivation trees in the following rounds.

However in that paper, we did not give the details on how to employ the linguistic knowledge to constrain the search space. In this paper, we will introduce a novel

algorithm that searches the space the derivation trees, respecting the linguistic constraints and maintaining the ambiguities in elementary tree extraction, for each given PTB tree. In this section, we first analyze the ambiguities existing in LTAG elementary tree extraction from PTB, and then we illustrate the linguistic information to be used in nondeterminisctic derivation extraction.

2.1 Ambiguities

Our analysis is mainly based on the work of (Xia, 2001) and (Chen, 2001). There are two kinds of ambiguities in LTAG elementary tree extraction, which are *head competition* and *argument-adjunct distinction.*

Given a deduction rule of Context Free Grammar (CFG) in the PTB, we need to find out which item on the right hand side of the rule is the *lexical head* of the item on the left hand side. For example, for rule VP → V NP, V is the head of VP. In (Xia, 2001) and (Chen, 2001), the so called head percolation table (Magerman, 1995) is used to determine the head item of each CFG rule in the PTB. Following the lexical heads in a tree, we can get the spines of elementary trees.

Argument-adjunct distinction is mainly to distinguish the arguments and adjuncts of a predicate. In (Xia, 2001) and (Chen, 2001), argument-adjunct distinction is solved with respect to the constituent tags and function tags in the PTB.

In their systems, these two kinds of ambiguities are solved deterministically as we described above. However, there exists a lot ambiguities that can not been solved easily. For example, in the sentence ... *was named a nonexecutive director of this British industrial conglomerate* extracted from the PTB, the NP dominating *director* has a function tag PRD, which means predicate. In this case there exists a head competition between *director* and *named*. As far as argument-adjunct distinction is concerned, the ambiguities are ubiquitous, which can only be solved with a lexicon, i.e. the hand-crafted XTAG English Grammar (XTAG-Group, 2001), with statistical methods.

2.2 Linguistics Information

2.2.1 Penn TreeBank

The input of our algorithm is a full bracketed PTB tree, as shown in Figure 1. The structure information is the main source of derivation tree extraction. We can simply regard a PTB tree as a derived tree in LTAG.

Besides the structure information, The Penn Treebank provides more information useful in LTAG derivation tree extraction, such as special information for the predicate-argument structures. Although the PTB does not try to distinguish *arguments* and *adjuncts*, and treats them as arguments in general, it assigns functional tags for these arguments which help to to distinguish arguments and
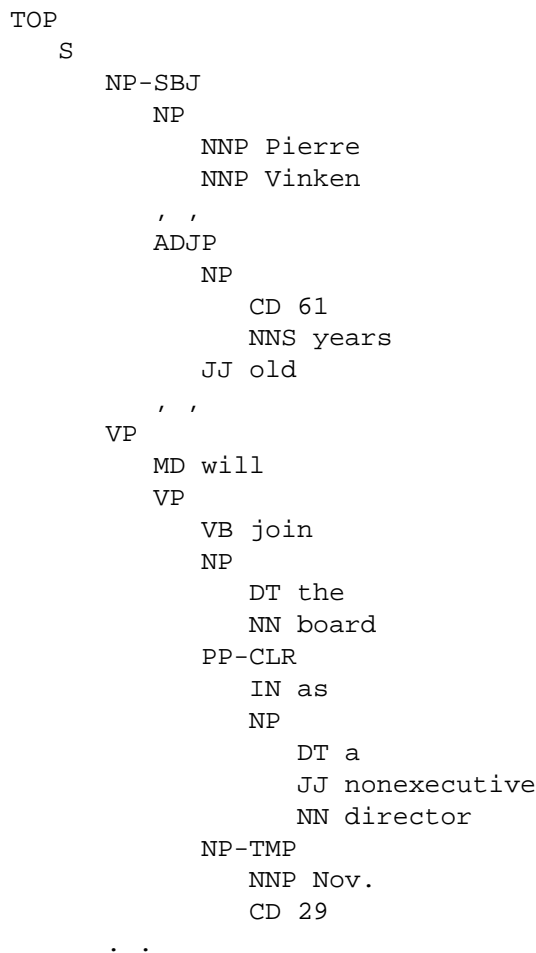
```
TOP
    S
        NP-SBJ
            NP
                NNP Pierre
                NNP Vinken
            , ,
            ADJP
                NP
                    CD 61
                    NNS years
                JJ old
            , ,
        VP
            MD will
            VP
                VB join
                NP
                    DT the
                    NN board
                PP-CLR
                    IN as
                    NP
                        DT a
                        JJ nonexecutive
                        NN director
                NP-TMP
                    NNP Nov.
                    CD 29
        . .
```

Figure 1: PTB tree

adjuncts. For example, in Figure 1, functional tag *SBJ* means subject, *TMP* means temporal phrase, and *CLR* means closely related. This information was used in previous LTAG extraction systems and will also be used in our system too.

2.2.2 PropBank

We will also use the Penn Proposition Bank (Prop-Bank) (Kingsbury and Palmer, 2002) in our LTAG derivation tree extraction algorithm. The PropBank provides more information on the predicate-argument structures for the PTB data.

In the PropBank, each predicate is assigned with a tag of *major sense* defined on usage of the predicate. Each argument of this predicate is assigned with an argument ID with respect to the *major sense* of this predicate, as shown in Figure 2. We will use the argument tags in our extraction algorithm too.

What is worth mentioning is that, like the PTB, the PropBank does not distinguish arguments and adjuncts, and both are called arguments in the PropBank. Using

```
TOP
    S
        [join:0_0_0:ARG0]NP-SBJ
            NP
                NNP Pierre
                NNP Vinken
            , ,
            ADJP
                NP
                    CD 61
                    NNS years
                JJ old
            , ,
        VP
            [join:1_0_0:ARGM]MD will
            VP
                [join:2_0_0:rel]VB join
                [join:3_0_0:ARG1]NP
                    DT the
                    NN board
                [join:4_0_0:ARGM]PP-CLR
                    IN as
                    NP
                        DT a
                        JJ nonexecutive
                        NN director
                [join:5_0_0:ARGM]NP-TMP
                    NNP Nov.
                    CD 29
        . .
```
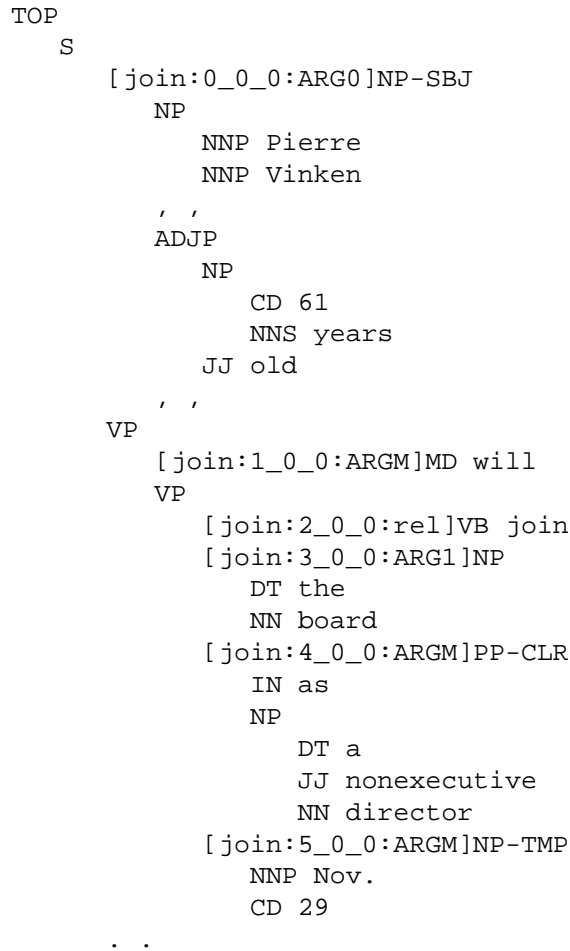
Figure 2: PTB tree with PropBank tags

the PropBank tags, we can easily get all the arguments
and adjuncts of a given predicate, but how to distinguish
them is still a problem. This is the reason why we in-
troduce the nondeterminisctic method in derivation tree
extraction. The EM algorithms in (Shen and Joshi, 2004)
are supposed to find out the global optimum over various
selections.

3 Nondeterministic Derivation Tree Extraction

In this paper we will propose an algorithm to main-
tain the ambiguities in elementary tree extraction. For a
given PTB tree, the derivation tree candidates serve as the
search space used in the EM algorithm. A naive way is
to represent all the candidate derivation trees one by one.
However, a more efficient way to do this is to use shared
structures. Therefore, we need an efficient way to repre-
sent all the LTAG derivation trees that meet the linguistic
constraints, given a PTB tree. Then the EM algorithm can
re-estimate expectation over all derivation tree candidates

with less computational complexity by inside-outside al-
gorithm as described in (Shen and Joshi, 2004).

3.1 Idea

We first give the idea of the extraction algorithm. The ex-
traction consists of two phases, the bottom-up head com-
petition phase and the top-down argument-adjunct dis-
tinction phase.

In the head-competition phase, we visit all the nodes
throughout a PTB tree from bottom and look for the head
candidates for every internal node. For each internal
node, there will be several head candidates with respect
to different triggering rules. For example, in the exam-
ple given above, ... *was named a nonexecutive director of
this British industrial conglomerate*, either the VP node
anchored on *named* or the NP node anchored on *direc-
tor* can be the head of the main S node with respect to
different analyses.

If any head candidate is selected as the head for the
parent node, the rest children nodes serve as

- a leaf node to which an initial tree substitutes,

- a leaf node to which an auxiliary tree adjoins, or

- an internal node of the elementary tree for the head
  node.

No matter what it will be, these elementary trees are
beneath the elementary tree for the head node, in any
derivation tree for this PTB tree. Thus, we can use shared
structure to represent all these situations; each non-head
child nodes can be used in three way as described above.

To sum up, for each head candidate, we can use an *in-
dex* structure to represent all the possible sub-structures
beneath this elementary tree in the derivation tree, and
this *index* structure will be further used for one or several
times by the upper nodes. So the goal of the head com-
petition phase is to search for all the head candidates for
each internal node and to generate an *index* structure the
represent all the sub-structures beneath this point.

In the argument-adjunct distinction phase, we visit *in-
dex* structures in a top-down style, starting from the *in-
dex* structures of the root node. Keeping track of the
head child, we first get the spine of the elementary for
the main predicate. Then we use linguistic information to
get all the possible operations with which other sub-trees
are attached to the sister nodes on different levels along
the spine. The possible attachment methods, i.e. sub-
stitution, adjunction or internal node, are recorded in the
index structures. Then we do argument-adjunct distinc-
tion recursively on all the subtrees rooted on these sister
nodes.

Now we explain what an *index* structure is. Each node
in a PTB tree is associated with a set of *index* structures.

An *index* structure stores the following information: its lexical head, a set of attachment points, and a set of the references of index structures of attached subtrees.

After the head competition phase and argument-adjunct distinction phase, we get a shared forest composed of *index* structures, with which we can get all the possible derivation trees of a PTB tree as well as the corresponding elementary trees.

3.2 Algorithm

The following algorithm is used to find all candidate derivation trees and store them in shared structures. The input is a PTB tree with PropBank tags, and the result of the algorithm is a shared forest that represents all candidate derivation trees for this input PTB tree.

1. head competition (node n)

   1.1 if (head-comp-done) return;

   1.2 for each child of n, call head competition;

   1.3 look for the candidate heads using linguistic constraints;

   1.4 for each candidate, generate an *index* structure and associate it with n;

   1.5 head-comp-done = true;

2. argument-adjunct distinction (node n)

   2.1 if (arg-adj-done) return;

   2.2 for each *index* structure of n, for each attachment candidate

      - refine attachment types;
      - call argument-adjunct distinction on the attached subtree;

   2.3 arg-adj-done = true;

3.3 Example

Figure 3 shows the results after head competition. For this case, there is no ambiguity on head competition, so there is only one output. Otherwise shared structures are used to represent all the results.

It is shown in Figure 3 that the result of the head competition does not distinguish arguments and adjuncts. The ambiguities are maintained in the single output in this case. Specifically, the PP subtree anchored on *as* can be either an argument or an adjunct. So do the NP subtrees for the subject, the object and the temporal phrase.

After the argument-adjunct distinction phase, ambiguities on the subject, the object and the temporal phrase are solved with respect to the templates defined on context, in a way similar to (Xia, 2001; Chen, 2001). However, the argument-adjunct ambiguity on the PP node is still maintained in the final output of the algorithm.

```
TOP
   [join:2_0_0:rel]S
      [join:0_0_0:ARG0]NP-SBJ
         %NP
            NNP Pierre
            %NNP Vinken
         , ,
         ADJP
            NP
               CD 61
               %NNS years
            %JJ old
         , ,
      %[join:2_0_0:rel]VP
         [join:1_0_0:ARGM]MD will
         %[join:2_0_0:rel]VP
            %[join:2_0_0:rel]VB join
            [join:3_0_0:ARG1]NP
               DT the
               %NN board
            [join:4_0_0:ARGM]PP-CLR
               %IN as
               NP
                  DT a
                  JJ nonexecutive
                  %NN director
            [join:5_0_0:ARGM]NP-TMP
               NNP Nov.
               %CD 29
      . .
```

Figure 3: PTB tree with head annotated. % stands for *head*

3.4 Discussion

In the previous sections, we did not cover the following two special situations in derivation tree extraction.

- Auxiliary predicate

- Coordination

In order to handle auxiliary predicate structure, we introduce a stack structure to maintain a head chain as described in (Chen, 2001). Since we have used the PropBank tags, we can easily recognize the predicate-argument relation between an argument associated with a sister node and a head in the stack.

Coordination is another important case in derivation extraction. In our current implementation, we use the first item in a coordination phrase as the head, and treat the rest items as adjuncts. For example, in the phrase *red and blue*, *red* is the head, *blue* attaches to *and*, and *and* adjoins to *red*. We are still working on the treatment of

202

coordination. One solution is to follow the approach described in (Sarkar and Joshi, 1996).

Theoretically, the number of the *index* structures grows exponentially with respect to the height of a tree. However, our experiments show that it does not grow that fast, thanks to the linguistic constraints used in head competition.

By using rich PropBank features, we can almost solve the ambiguities in head competition. The only thing that we need to take care of is the different analyses of head word in PropBank and LTAG. In some cases, PropBank and LTAG select different words as head. However, this problem can be solved with templates. For example, in the following example, *thought* is the head for the whole sentence in PropBank, while *come* is the head in LTAG analysis.

- *John thought Mary didn't come yet.*

4 Experiments

The nondeterminisctic LTAG derivation extraction algorithm was used in the EM models reported in (Shen and Joshi, 2004). With the algorithm given in that paper, about 12,000 elementary trees were extracted from the Penn Treebank. The experiments in (Shen and Joshi, 2004) showed that the number of the elementary trees was reduced to about 10,000 with several rounds of EM training.

It also noted in (Shen and Joshi, 2004) that some simple EM models reported in that paper prefer elementary trees of lower frequency, which is undesirable for grammar extraction. In our future research, we will incorporate the hand crafted XTAG English Grammar (XTAG-Group, 2001) in the EM models. Some XTAG Grammar based EM models were proposed in (Shen and Joshi, 2004) as future work.

5 Conclusions

In this paper we have introduced a naive algorithm for nondeterminisctic LTAG derivation tree extraction from the Penn Treebank and the PropBank, which is an extension of the deterministic methods in (Xia, 2001) and (Chen, 2001). The algorithm will be used in the EM models of LTAG Treebank Induction. The shared structures generated by this nondeterminisctic algorithm allow efficient expectation computation via dynamic programming in the EM algorithm.

References

J. Chen. 2001. *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

D. Chiang and D. M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan.

D. Chiang. 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proceedings of ACL-2000*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38.

A. Joshi and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *COLING'94*.

P. Kingsbury and M. Palmer. 2002. From treebank to propbank. In *Proceedings of the 3rd LREC*.

D. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd ACL*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

M. Paola and M. Leybold. 2001. Automatic distinction of arguments and modifiers: the case of prepositional phrases. In *Proceedings of the Fifth Workshop on Computational Natural Language Learning (CoNLL-01)*, Toulouse, France.

A. Sarkar and A. K. Joshi. 1996. Coordination in tree adjoining grammars: Formalization and implementation. In *Proceedings of COLING 1996*.

L. Shen and A. K. Joshi. 2004. Extracting Deeper Information from Richer Resource: EM Models for LTAG Treebank Induction. In *Proceedings of IJCNLP 2004*.

L. Shen, A. Sarkar, and A. K. Joshi. 2003. Using ltag based features in parse reranking. In *Proceedings of EMNLP 2003*.

F. Xia. 2001. *Automatic Grammar Generation From Two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.

XTAG-Group. 2001. A lexicalized tree adjoining grammar for english. Technical Report 01-03, IRCS, Univ. of Pennsylvania.