Tree-Adjoining Grammar (TAG)

Linguistics 614

With thanks to Detmar Meurers & Laura Kallmeyer

Spring 2015

Introduction to TAG

TAG (Joshi et al. 1975, Joshi & Schabes 1997) extends CFG in the following sense:

- In a CFG, each derivation step amounts to substituting a new tree of height 1 for a leaf.
- In a TAG, we allow (finite) trees that are arbitrarily large

Phrase Structure Trees

S \rightarrow NP VP

NP \rightarrow John

ADV \rightarrow sometimes

VP \rightarrow V

 sometimes

V \rightarrow laughs

String rewriting derivation

- S
- NP VP (rule #1)
- John VP (rule #4)
- John ADV VP (rule #2)
- John sometimes VP (rule #5)
- John sometimes V (rule #3)
- John sometimes laughs (rule #6)

Overview

- Introduction
  - Introduction to TAG
  - Feature-Based TAG

- TAG for Natural Languages
  - Properties of Elementary Trees
  - Constituency and Dependency

- Conclusion

Introduction to TAG (3)

Substitution (Tree Substitution Grammars (TSGs))

Elementary structures are trees

- Arrow indicates where substitution takes place

Tree 1:

S
NP ↓ VP

John

sometimes

V

laughs

Tree 2:

S
NP ↓ VP

NP ↓ VP

John

laughs
Introduction to TAG (5)

Q: with TSGs, how would we obtain John heartily laughs?

- Besides substitution at leaves, we also can replace internal nodes with new trees (adjunction).
  - In an adjunction, the new tree is an auxiliary tree and has a special leaf, the foot node.
  - The trees that are added in substitution operations are called initial trees.

Auxiliary tree modifies an XP only if root & foot nodes are both XP

- Using adjunction gives Tree Adjoining Grammar (TAG)

Introduction to TAG (6)

Using adjunction gives Tree Adjoining Grammar (TAG)

Introduction to TAG (7)

A Tree Adjoining Grammar (TAG) is a quadruple $G = (N, T, I, A)$ such that

- $T$ and $N$ are disjoint alphabets, the terminals and nonterminals,
- $I$ is a finite set of initial trees, and
- $A$ is a finite set of auxiliary trees.

The trees in $I \cup A$ are called elementary trees.

- nodes in elementary trees are labeled with symbols from $T \cup N \cup \{\varepsilon\}$
- all internal nodes have labels from $N$
- $G$ is lexicalized iff each elementary tree has at least one leaf with a terminal label.

Introduction to TAG (8)

TAG as defined above are more powerful than CFG but they cannot generate the copy language ($\{ww \mid w \in \{a, b\}^*\}$).

In order to increase the expressive power, adjunction constraints are introduced that specify for each node whether adjunction is mandatory and which trees can be adjoined.

Three types of constraints are distinguished:

- A node is said to carry a obligatory adjunction (OA) constraint if adjunction is obligatory at that node.
- A node is said to carry a null adjunction (NA) constraint if adjunction is not obligatory and the set of adjoinable trees is empty.
- A node is said to carry a selective adjunction (SA) constraint if adjunction is not obligatory and the set of adjoinable trees is not empty.

Example: TAG for the copy language
Introduction to TAG (7)

Example

(2) John seems to sleep

\[
\begin{array}{c}
\text{NP} \rightarrow S \\
\text{NP} \rightarrow \text{VP} \rightarrow \text{OA} \rightarrow \text{V} \\
\text{NP} \rightarrow \text{V} \rightarrow \text{to} \rightarrow \text{V} \\
\text{NP} \rightarrow \text{sleep} \\
\end{array}
\]

Feature-Based TAG (1)

Feature-structure based TAG (FTAG) (Vijay-Shanker & Joshi, 1988): each node has a top and a bottom feature structure (except substitution nodes that have only a top). Nodes in the same elementary tree can share features (extended domain of locality).

Intuition:
- The top feature structure tells us something about what the node represents within the surrounding structure, and
- the bottom feature structure tells us something about what the tree below the node represents.

In the final derived tree, both must be the same.

FTAG (2)

Example:

\[
\begin{array}{c}
\text{cat} \rightarrow \text{S} \\
\text{cat} \rightarrow \text{VP} \\
\text{cat} \rightarrow \text{V} \\
\end{array}
\]

FTAG (3)

Example:

\[
\begin{array}{c}
\text{cat} \rightarrow \text{NP} \\
\text{cat} \rightarrow \text{VP} \\
\text{cat} \rightarrow \text{V} \\
\end{array}
\]
FTAG (4)

Unification during derivation:
- Substitution: the top of the root of the new initial tree unifies with the top of the substitution node.
- Adjunction: the top of the root of the new auxiliary tree unifies with the top of the adjunction site, and the bottom of the foot of the new tree unifies with the bottom of the adjunction site.
- In the final derived tree, top and bottom unify for all nodes.

FTAG (5)

Example:

In FTAG, there are no explicit adjunction constraints. Instead, adjunction constraints are expressed via feature unification requirements.
- Important: LTAG feature structures are restricted; there is only a finite set of possible feature structures.
- Therefore, the following can be shown:
  
  For each FTAG there exists a weakly equivalent TAG with adjunction constraints and vice versa. The two TAGs generate even the same sets of trees, only with different node labels.

FTAG (6)

Example:

In FTAG, there are no explicit adjunction constraints.
- Instead, adjunction constraints are expressed via feature unification requirements.
- Important: LTAG feature structures are restricted; there is only a finite set of possible feature structures.
- Therefore, the following can be shown:
  
  For each FTAG there exists a weakly equivalent TAG with adjunction constraints and vice versa. The two TAGs generate even the same sets of trees, only with different node labels.

FTAG (7)

In FTAG, there are no explicit adjunction constraints.
- Instead, adjunction constraints are expressed via feature unification requirements.
- Important: LTAG feature structures are restricted; there is only a finite set of possible feature structures.
- Therefore, the following can be shown:
  
  For each FTAG there exists a weakly equivalent TAG with adjunction constraints and vice versa. The two TAGs generate even the same sets of trees, only with different node labels.

LTAG game: http://www.ltaggame.com
Elementary trees (3)

- Elementary trees are extended projections of lexical items.
  - Recursion is factored away ⇒ finite set of elementary trees.
  - The elementary tree of a lexical predicate contains slots for all arguments of the predicate, for nothing more.
- Besides lexical predicates, there are functional elements (complementizers, determiners, auxiliaries, negation) whose treatment in LTAG is less clear. They can be
  - either in separate elementary trees (XTAG, 2001)
  - or in the elementary tree of the lexical item they are associated with (Frank, 2002).

Elementary trees (4)

Example

(4) John gives a book to Mary

Elementary trees (5)

Example:

(5) John expected Mary to make a comment

expected selects for a subject NP and an infinitival sentence:

The sentential object is realised as a foot node in order to allow extractions:

(6) whom does John expect to come?

Elementary trees (6)

Example with modifiers:

(7) the good student participated in every course during the semester

Elementary trees (8)
Elementary trees (9)

- Constraints on larger structures (constraints on “unbounded dependencies”) need not be stipulated but follow from the possibilities of adjunction in the elementary trees.
- **Fundamental LTAG hypothesis**: Every syntactic dependency is expressed locally within a single elementary tree.
- **Non-local dependency corollary**: Non-local dependencies always reduce to local ones once recursive structure is factored away.

What do the elementary trees look like for the following sentence?

(8) which book did Harvey say Cecile had read

Introduction
TAG for Natural Languages
Conclusion
Properties of Elementary Trees
Constituency and Dependency (4)

Raising to Object

(13) John expects [ Bill to win ]

\[
\begin{align*}
N & \downarrow \quad V \\
\text{expects} & \quad \text{to win} \\
\text{John} & \quad \text{Bill}
\end{align*}
\]

Constituency and Dependency (5)

Object-control Equi

(14) John persuades Bill [ PRO to leave ]

\[
\begin{align*}
N & \downarrow \quad V \\
\text{persuades} & \quad \text{to leave} \\
\text{John} & \quad \text{Bill}
\end{align*}
\]

Constituency and Dependency (6)

Subject raising

(15) John seems to like Bill

\[
\begin{align*}
V & \downarrow \quad V \\
\text{seems} & \quad \text{to like} \\
\text{John} & \quad \text{Bill}
\end{align*}
\]

Constituency and Dependency (7)

Long distance phenomena

(16) which book did Harvey say Cecile had read

\[
\begin{align*}
N & \downarrow \quad N \\
\text{had read} & \quad \text{to like} \\
\text{which book} & \quad \text{Cecile} \\
\text{Harvey}
\end{align*}
\]

Constituency and Dependency (8)

The derivation tree is not always the semantic dependency structure:

(17) roasted red pepper

\[
\begin{align*}
N & \downarrow \quad N \\
\text{roasted} & \quad \text{red} \\
\text{pepper}
\end{align*}
\]

⇒ proposal of alternative derivation with multiple adjunctions of modifier trees at the same node.

Constituency and Dependency (9)

On the other hand, multiple adjunctions are not always desired:

(18) John seems to be likely to win the race

\[
\begin{align*}
V & \downarrow \quad V \\
\text{seems} & \quad \text{to win} \\
\text{John} & \quad \text{the race}
\end{align*}
\]

This is the correct dependency structure.
Constituency and Dependency (10)

Another problematic case:

(19) John claims Bill is likely to win

to

1 ǫ 2

Bill claims is likely

1

John

Conclusion

- TAG extend CFGs by introducing *adjunction*, in addition to *substitution*. TAG are only slightly more powerful than CFG.
- Elementary trees of lexical predicates encapsulate subcategorization frames: For each subcategorized argument, there is a non-terminal leaf (either a substitution node or a foot node).
- Recursion is factored away: only slots for subcategorized arguments are provided. Modifiers are added by adjunction.
- Extended domain of locality: Syntactic dependencies are defined locally, within single elementary trees. Unbounded dependencies arise from adjunction between an argument and its lexical head.

References