A Psycholinguistically Motivated version of Tree Adjoining Grammar (TAG)

Sayyed Auwn Muhammad
Introduction

• Language comprehension is incremental
  ▫ Comprehenders build an interpretation of a sentence on word–by–word basis

• Human Sentence Processing Properties
  ▫ Incrementality
  ▫ Connectedness
  ▫ Prediction
Incrementality

- Perception of a word in a sentence leads to integration into a already perceived structure of the sentence
  - Left to right processing
  - word-by-word basis
- Strict Incrementality
  - Fully connectedness [Sturt and Lombardo 2005]
  - Connected under the same syntactic root node
Connectedness

- At any point of incremental sentence processing
  - All words are attached to a single syntactic structure
  - Parser are not allowed to build unconnected tree fragments
Prediction

- To achieve fully connectivity
- Make prediction of upcoming
  - Words
  - Structures
- Prediction about Structure
  - Previous structure
  - Lexicon entries
Notion of Prediction

• Either ... or construction [Staub and Clifton 2006]
  • Word *either* triggers prediction of *or* and the second conjunct
  • Syntactic parallelism indicates that the second conjunct of a coordinate structure is processed faster if its internal structure is identical to that of the first conjunct.

“Mary is looking for either a maid or a cook”
Notion of Prediction

- Support to perdition
- Linking Parsing with Processing difficulty
- Subject Relative Clause (SRC) occurs more often than Object Relative Clause (ORC) [King and Just, 1991; Gibson 1998]

  - \((\text{SRC})\) The reporter that attacked the senator admitted the error.
  - \((\text{ORC})\) The reporter that the senator attacked admitted the error.

- Higher Processing difficulty occurs when the more probable structure has to be discarded.
Grammar Formalisms

- **Context Free Grammar (CFG)**
  - Production rules, derivation trees
  - PCFG (Probabilistic CFG)
  - Context Sensitive Grammar (CSG)
- **Dependency Grammar (DG)**
  - Relation between a word (a head) and its dependents
  - Lack phrasal node
- **Combinatory Categorial Grammar (CCG)**
- **Tree Adjoining Grammar (TAG)**
  - More expressive power
  - Richer structural description to sentence [David Chiang 2004]
  - Respect fully connectedness and the prediction task
Tree Adjoining Grammar

• Tree-adjoining grammar (TAG) is defined by [Joshi et al., 1975]

• Rules in a TAG are trees

• Two types of basic trees in TAG:
  ▫ initial trees (α)
  ▫ auxiliary trees (β)
Tree Adjoining Grammar Formalism

- Tree types
  - Initial trees
  - Auxiliary trees
- Operations
  - Substitution
  - Adjunction
Substitution operation

- Substitution node
- Substitution Symbol ↓
- Derived tree
Adjunction operation

- Same root node as foot node
- Recursive in nature
- Two operations
Problem with LTAG

- Does not allow derivation in strictly incremental fashion
- Consider the example of “Peter often reads a book”
- The head *reads* which provide the intervening structure has not been encountered yet.
Problem with LTAG
Proposed PLTAG

- **Lexicon:**
  - Canonical LTAG Lexicon
  - Prediction Lexicon

- **Operations:**
  - Substitution
  - Adjunction
  - Verification
Verification

- Prediction nodes need to be verified
- Verification is an operation that removes prediction indices
Connection Path Concept

- The minimal amount of structure needed at each word for sentence incremental processing.
Definition of a PLTAG Derivation:

- A PLTAG derivation starts with the tree of the first input word, and then applies **Substitution and adjunction operations** to canonical trees or prediction trees. Every prediction tree has to be validated using the **verification operation** later.
- In a partial PLTAG derivation for words $w_1..w_i$, all leaves to the left of $w_i$ must be **fully lexicalized**. A PLTAG derivation is **complete** when every leaf node is labeled with a terminal symbol, none of the nodes in the tree is marked as predictive, and the root symbol of the derived tree is S.
PLTAG

OK:

not OK:

OK:

not OK:

19
PLTAG

- PLTAG derivations are always strictly incremental.
PLTAG

- PLTAG derivations are always strictly incremental.
Relationship between LTAG and PLTAG

- Every LTAG derivation can be translated into an equivalent PLTAG derivation.
Steps in Constructing the Parser

1. Conversion of the Penn Treebank into PLTAG format
2. Lexicon Induction
3. The Incremental Parsing Algorithm
4. The Probability Model
5. Parser Evaluation
Conversion of the Penn Treebank into PLTAG format

- Induce the lexicon (both canonical trees and prediction trees) needed for PLTAG from Penn Treebank
Lexicon Induction: creating 1

Sentence Tree:

```
S
 / \
NP   VP
 /   /
DET  ADVP
|    |   VP
A   never V   NP
     man   risks

ADJ careful
N man
```

Canonical Lexicon Entries:

```
S
 / \
NP   VP
 /   /
ADVP never VP*
\    /
   V   NP
   takes risks

ADJ careful
N man
```

DET A
N N*
ADJ N
DET N
ADVP
VP
V
NP

Lexicon Induction: creating 2
The Parsing Algorithm

• **Requirements:**
  ▫ Produce incremental and fully connected structures at every point in time
  ▫ Only produce valid PLTAG trees

• **Helpful Concept: Fringes**
  ▫ tree can be described by its depth-first traversal
  ▫ only part of incremental tree is relevant at each step

---

**Example**

```
S
  /   |
NP   VP
  /   |
Peter sleeps
```

(S, NP, Peter, Peter, NP, VP, sleeps, sleeps, VP, S)
Verification: Fringes

![Diagram showing the process of verification involving substitution, adjectival phrase attachment, and verb phrase attachment.]
The Parsing Algorithm

• **Requirements:**
  ▫ Produce incremental and fully connected structures at every point in time
  ▫ Only produce valid PLTAG trees

• **Helpful Concept: Fringes**
  ▫ tree can be described by its depth-first traversal
  ▫ only part of incremental tree is relevant at each step

---

**Example**

```
S
   / \
NP  VP  
   / \   |
Peter sleeps
```

(S⁺, NP⁺, Peter⁺, Peter⁻, NP⁻, VP⁺, sleeps⁺, sleeps⁻, VP⁻, S⁻)
The Parsing Algorithm

• **Requirements:**
  ▫ Produce incremental and fully connected structures at every point in time
  ▫ Only produce valid PLTAG trees

• **Helpful Concept: Fringes**
  ▫ tree can be described by its depth-first traversal
  ▫ only part of incremental tree is relevant at each step

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Peter</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>NP↓</td>
</tr>
<tr>
<td>VP</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>sleeps</td>
</tr>
<tr>
<td>(NP⁺, Peter⁺, Peter⁻, NP⁻)</td>
</tr>
<tr>
<td>(S⁺, NP↓⁺, NP↓⁻, VP⁺, sleep⁺, sleep⁻, VP⁻, S⁻)</td>
</tr>
</tbody>
</table>
The Parsing Algorithm

• Requirements:
  ▫ Produce incremental and fully connected structures at every point in time
  ▫ Only produce valid PLTAG trees

• Helpful Concept: Fringes
  ▫ tree can be described by its depth-first traversal
  ▫ only part of incremental tree is relevant at each step

Example

- S
  - NP
    - Peter
  - VP
    - sleeps

- (NP⁺, Peter⁺, Peter⁻, NP⁻)
- (S⁺, NP↓⁺, NP↓⁻, VP⁺, sleep⁺, sleep⁻, VP⁻, S⁻)
- (S⁺, NP⁺, Peter⁺, Peter⁻, NP⁻, VP⁺, sleeps⁺, sleeps⁻, VP⁻, S⁻)
**The Parsing Algorithm**

- **Requirements:**
  - Produce incremental and fully connected structures at every point in time
  - Only produce valid PLTAG trees

- **Helpful Concept: Fringes**
  - Tree can be described by its depth-first traversal
  - Only part of incremental tree is relevant at each step

### Example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>(NP⁺, Peter⁺, ⋅ Peter⁻, NP⁻)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>S</td>
<td>(S⁺, NP⁺, Peter⁺, Peter⁻, NP⁻, VP⁺, sleeps⁺, sleeps⁻, VP⁻, S⁻)</td>
</tr>
<tr>
<td>Peter</td>
<td>NP⁻</td>
<td>(NP⁺, Peter⁺, ⋅ sleeps⁻, VP⁻, S⁻)</td>
</tr>
<tr>
<td></td>
<td>VP</td>
<td></td>
</tr>
</tbody>
</table>
Probability Model

Substitution: \( \sum_{\varepsilon} P(\varepsilon|\eta_\beta) = 1 \)

Adjunction: \( \sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(\text{NONE}|\eta_\beta) = 1 \)

Verification: \( \sum_{\varepsilon} P(\varepsilon|\pi_\beta) = 1 \)

\[
\begin{align*}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_\eta, tnm)
\end{align*}
\]

based on [Chiang, 2000]

Explanation

Probabilities are normalized with respect to other elementary trees \( \varepsilon \) that can attach at node \( \eta \) in prefix tree \( \beta \) with the same operation.

<table>
<thead>
<tr>
<th>elementary trees</th>
<th>( \varepsilon )</th>
<th>prefix tree</th>
<th>( \beta )</th>
<th>prediction trees</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree structures</td>
<td>( \tau )</td>
<td>integration point node</td>
<td>( \eta )</td>
<td>a tree's head leaf</td>
<td>( \lambda )</td>
</tr>
</tbody>
</table>
Probability Model

Substitution:  \[ \sum_{\varepsilon} P(\varepsilon | \eta_{\beta}) = 1 \]

Adjunction:  \[ \sum_{\varepsilon} P(\varepsilon | \eta_{\beta}) + P(\text{NONE} | \eta_{\beta}) = 1 \]

Verification:  \[ \sum_{\varepsilon} P(\varepsilon | \eta_{\beta}) = 1 \]

\[
P(\varepsilon | \eta_{\beta}) = P(\tau_{\varepsilon} | \eta_{\beta}) \times P(\lambda_{\varepsilon} | \tau_{\varepsilon}, \lambda_{\eta})
\]
\[
P(\varepsilon | \pi_{\beta}) = P(\tau_{\varepsilon} | \pi_{\beta}) \times P(\lambda_{\varepsilon} | \tau_{\varepsilon}, \lambda_{\pi_{\eta}})
\]
\[
P(\eta_{\beta}) = P(\tau_{\eta}, \lambda_{\eta}, c_{\eta}, n_{\eta}, b_{\eta}, a_{f}, tm)
\]

based on [Chiang, 2000]

---

**Explanation**

**elementary tree** \( \varepsilon \):

```
NP
  DT↓  NN
    reporter
```

is estimated as template \( \tau_{\varepsilon} \):

```
NP
  DT↓  NN
```

and lexeme \( \lambda_{\varepsilon} \): *reporter*

---

<table>
<thead>
<tr>
<th>elementary trees</th>
<th>( \varepsilon )</th>
<th>prefix tree</th>
<th>( \beta )</th>
<th>prediction trees</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree structures</td>
<td>( \tau )</td>
<td>integration point node</td>
<td>( \eta )</td>
<td>a tree’s head leaf</td>
<td>( \lambda )</td>
</tr>
</tbody>
</table>
Probability Model

Substitution: \[ \sum_{\varepsilon} P(\varepsilon|\eta_\beta) = 1 \]

Adjunction: \[ \sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(\text{NONE}|\eta_\beta) = 1 \]

Verification: \[ \sum_{\varepsilon} P(\varepsilon|\pi_\beta) = 1 \]

\[
\begin{align*}
P(\varepsilon|\eta_\beta) &= P(\tau_c|\eta_\beta) \times P(\lambda_c|\tau_c, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\pi_\eta) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{align*}
\]

based on [Chiang, 2000]

<table>
<thead>
<tr>
<th>elementary trees</th>
<th>( \varepsilon )</th>
<th>prefix tree</th>
<th>( \beta )</th>
<th>prediction trees</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree structures</td>
<td>( \tau )</td>
<td>integration point node</td>
<td>( \eta )</td>
<td>a tree's head leaf</td>
<td>( \lambda )</td>
</tr>
</tbody>
</table>
Probability Model

Substitution: \( \sum_{\varepsilon} P(\varepsilon | \eta_\beta) = 1 \)

Adjunction: \( \sum_{\varepsilon} P(\varepsilon | \eta_\beta) + P(\text{NONE} | \eta_\beta) = 1 \)

Verification: \( \sum_{\varepsilon} P(\varepsilon | \pi_\beta) = 1 \)

\[
\begin{align*}
P(\varepsilon | \eta_\beta) &= P(\tau_c | \eta_\beta) \times P(\lambda_c | \tau_c, \lambda_\eta) \\
P(\varepsilon | \pi_\beta) &= P(\tau_\varepsilon | \pi_\beta) \times P(\lambda_\varepsilon | \tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{align*}
\]

based on [Chiang, 2000]
Probability Model

Substitution: \( \sum_{\varepsilon} P(\varepsilon | \eta_\beta) = 1 \)

Adjunction: \( \sum_{\varepsilon} P(\varepsilon | \eta_\beta) + P(\text{NONE} | \eta_\beta) = 1 \)

Verification: \( \sum_{\varepsilon} P(\varepsilon | \pi_\beta) = 1 \)

\[
\begin{align*}
P(\varepsilon | \eta_\beta) &= P(\tau_\varepsilon | \eta_\beta) \times P(\lambda_\varepsilon | \tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon | \pi_\beta) &= P(\tau_\varepsilon | \pi_\beta) \times P(\lambda_\varepsilon | \tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_\eta, tm)
\end{align*}
\]

based on [Chiang, 2000]

<table>
<thead>
<tr>
<th>elementary trees</th>
<th>(\varepsilon)</th>
<th>prefix tree</th>
<th>(\beta)</th>
<th>prediction trees</th>
<th>(\pi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree structures</td>
<td>(\tau)</td>
<td>integration point node</td>
<td>(\eta)</td>
<td>a tree's head leaf</td>
<td>(\lambda)</td>
</tr>
</tbody>
</table>
Probability Model

Substitution: \[ \sum_{\varepsilon} P(\varepsilon|\eta_{\beta}) = 1 \]

Adjunction: \[ \sum_{\varepsilon} P(\varepsilon|\eta_{\beta}) + P(\text{NONE}|\eta_{\beta}) = 1 \]

Verification: \[ \sum_{\varepsilon} P(\varepsilon|\pi_{\beta}) = 1 \]

\[
P(\varepsilon|\eta_{\beta}) = P(\tau_{\varepsilon}|\eta_{\beta}) \times P(\lambda_{c}|\tau_{\varepsilon}, \lambda_{\eta})
\]

\[
P(\varepsilon|\pi_{\beta}) = P(\tau_{\varepsilon}|\pi_{\beta}) \times P(\lambda_{\varepsilon}|\tau_{\varepsilon}, \lambda_{\pi_{\eta}})
\]

\[
P(\eta_{\beta}) = P(\tau_{\eta}, \lambda_{\eta}, c_{\eta}, n_{\eta}, b_{\eta}, a_{f}, t_{m})
\]

based on [Chiang, 2000]

<table>
<thead>
<tr>
<th>elementary trees</th>
<th>( \varepsilon )</th>
<th>prefix tree</th>
<th>( \beta )</th>
<th>prediction trees</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree structures</td>
<td>( \tau )</td>
<td>integration point node</td>
<td>( \eta )</td>
<td>a tree's head leaf</td>
<td>( \lambda )</td>
</tr>
</tbody>
</table>
Probability Model

Substitution: \[ \sum_{\varepsilon} P(\varepsilon | \eta_\beta) = 1 \]

Adjunction: \[ \sum_{\varepsilon} P(\varepsilon | \eta_\beta) + P(\text{NONE} | \eta_\beta) = 1 \]

Verification: \[ \sum_{\varepsilon} P(\varepsilon | \pi_\beta) = 1 \]

\[
P(\varepsilon | \eta_\beta) = P(\tau_c | \eta_\beta) \times P(\lambda_c | \tau_c, \lambda_\eta) \\
P(\varepsilon | \pi_\beta) = P(\tau_\varepsilon | \pi_\beta) \times P(\lambda_\varepsilon | \tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) = P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\]

based on [Chiang, 2000]
Probability Model

Substitution: \( \sum_{\varepsilon} P(\varepsilon|\eta_\beta) = 1 \)

Adjunction: \( \sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(\text{NONE}|\eta_\beta) = 1 \)

Verification: \( \sum_{\varepsilon} P(\varepsilon|\pi_\beta) = 1 \)

\[
P(\varepsilon|\eta_\beta) = P(\tau_c|\eta_\beta) \times P(\lambda_c|\tau_c, \lambda_n)
\]

\[
P(\varepsilon|\pi_\beta) = P(\tau_e|\pi_\beta) \times P(\lambda_e|\tau_e, \lambda_{\pi_n})
\]

\[
P(\eta_\beta) = P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, t_m)
\]

Based on [Chiang, 2000]
Probability Model

Substitution: \[ \sum_{\epsilon} P(\epsilon | \eta_{\beta}) = 1 \]

Adjunction: \[ \sum_{\epsilon} P(\epsilon | \eta_{\beta}) + P(\text{NONE} | \eta_{\beta}) = 1 \]

Verification: \[ \sum_{\epsilon} P(\epsilon | \pi_{\beta}) = 1 \]

\[ P(\epsilon | \eta_{\beta}) = P(\tau_{c} | \eta_{\beta}) \times P(\lambda_{c} | \tau_{c}, \lambda_{\eta}) \]
\[ P(\epsilon | \pi_{\beta}) = P(\tau_{e} | \pi_{\beta}) \times P(\lambda_{e} | \tau_{e}, \lambda_{\pi_{\eta}}) \]
\[ P(\eta_{\beta}) = P(\tau_{\eta}, \lambda_{\eta}, c_{\eta}, n_{\eta}, b_{\eta}, a_{f}, t m) \]

based on [Chiang, 2000]

<table>
<thead>
<tr>
<th>elementary trees</th>
<th>( \varepsilon )</th>
<th>prefix tree</th>
<th>( \beta )</th>
<th>prediction trees</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree structures</td>
<td>( \tau )</td>
<td>integration point node</td>
<td>( \eta )</td>
<td>a tree's head leaf</td>
<td>( \lambda )</td>
</tr>
</tbody>
</table>

\[ \beta : \]
\[ S \]
\[ NP \rightarrow \]
\[ Paul \]
\[ V \rightarrow \]
\[ saw \]
\[ NP \downarrow \]
\[ + \varepsilon : \]
\[ NP \rightarrow \]
\[ Mary \]

a trace mark \( tm \) which marks whether there is a trace at the beginning or end of the fringe
Parser Evaluation

- Parser Performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec</th>
<th>Recall</th>
<th>F-score</th>
<th>Cov</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLTAG parser</td>
<td>79.43</td>
<td>79.39</td>
<td>79.41</td>
<td>98.09</td>
</tr>
<tr>
<td>Pred tree oracle</td>
<td>81.15</td>
<td>81.13</td>
<td>81.14</td>
<td>96.18</td>
</tr>
<tr>
<td>No gold POS</td>
<td>77.57</td>
<td>77.24</td>
<td>77.41</td>
<td>98.09</td>
</tr>
</tbody>
</table>
Parser Evaluation

- Comparison with other TAG Parser

<table>
<thead>
<tr>
<th>Model</th>
<th>incr</th>
<th>con</th>
<th>pred</th>
<th>impl</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazzei et al. (2007)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>n/a</td>
</tr>
<tr>
<td>This work (gold POS)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>79.4</td>
</tr>
<tr>
<td>Kato et al. (2004)</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>79.7</td>
</tr>
<tr>
<td>Shen and Joshi (2005)</td>
<td>(+)</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>(87.4)</td>
</tr>
<tr>
<td>Chiang (2000)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>86.7</td>
</tr>
</tbody>
</table>
References:


3) Vera Demberg, “Incremental, Predictive Parsing with Psycholinguistically motivated Tree Adjoining Grammar”
The End

• Thanks for your attention