Incremental Tree-Adjoining Grammars

Vera Demberg

Cluster of Excellence “Multimodal Computing and Interaction”
Universität des Saarlandes

Incremental Syntax and Semantics Seminar WS 2012/13

November 7th, 2012

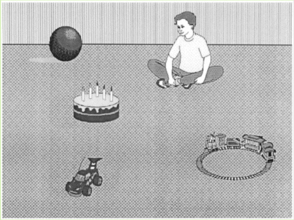
Table of Contents

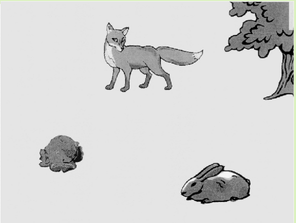
Table of Contents

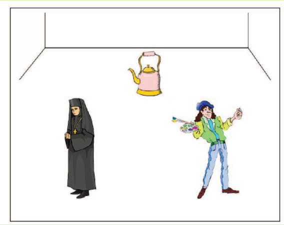
Reminder last week's materials

Evidence for incremental processing and even predictive processes

- Visual World and Eye-tracking
- Eye-tracking in Reading
- ERP / EEG





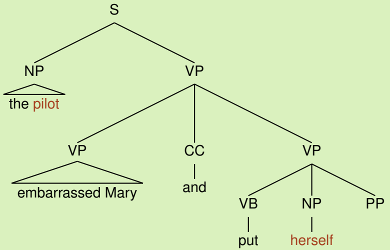


Reminder last week's materials

Evidence for incremental processing and even predictive processes

- Visual World and Eye-tracking
- Eye-tracking in Reading
- ERP / EEG

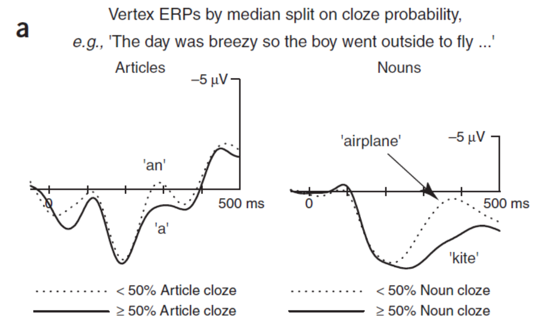
Either..or



Reminder last week's materials

Evidence for incremental processing and even predictive processes

- Visual World and Eye-tracking
- Eye-tracking in Reading
- ERP / EEG



Reminder of last weeks experimental findings

Results and Conclusions:

- Processing is generally **incremental** (syntax and semantics)
- Evidence for syntactic connectedness at specific points in the sentence (c-command relation)
- People **anticipate arguments**
- They do **not anticipate adjuncts**

Question for next few sessions:
How can we model incremental processing?

Reminder of last weeks experimental findings

Results and Conclusions:

- Processing is generally **incremental** (syntax and semantics)
- Evidence for syntactic connectedness at specific points in the sentence (c-command relation)
- People **anticipate arguments**
- They do **not anticipate adjuncts**

Question for next few sessions:
How can we model incremental processing?

Psycholinguistic Modelling

What is a cognitive model of language processing?

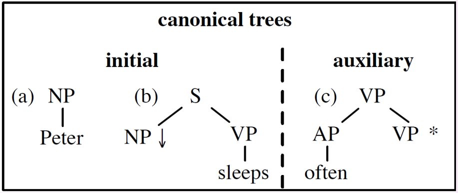
$\boxed{\text{Input: words}} \Rightarrow \boxed{\text{Output: per word difficulty}}$

- **Processing Model** (parser)
- **Linking Theory** (relates the parsing process to processing difficulty), e.g.
  - quantifying the ease with which particular structures are constructed or accessed
  - quantify how long it takes to disambiguate between alternatives
  - information theoretic measure, e.g. surprisal / entropy

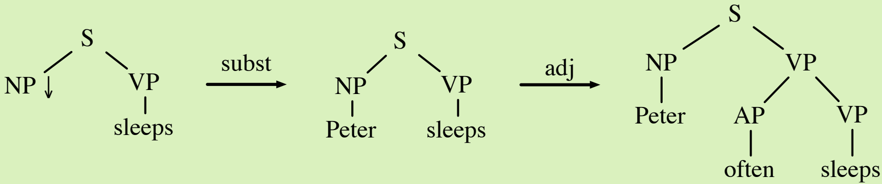
Table of Contents

Lexicalized Tree-Adjoining Grammar

Lexicon



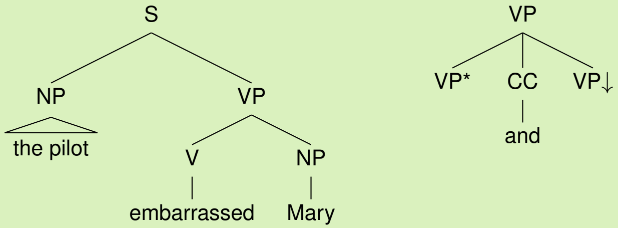
LTAG-Derivation for *Peter often sleeps.*



How suitable is TAG for explaining the experimental results we have seen?

TAG has been argued to be a suitable grammar formalism, consider:

Examples from experiments

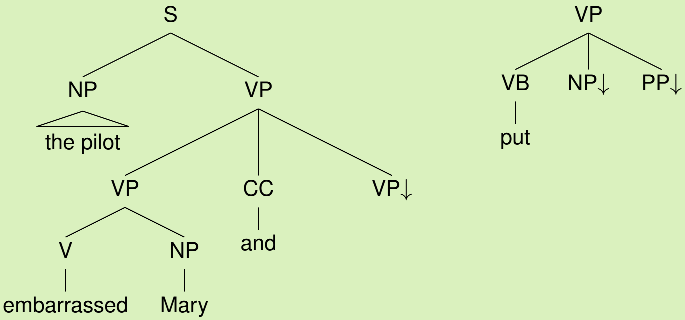


- TAG adjunction can model e.g. connectedness at *herself*.

How suitable is TAG for explaining the experimental results we have seen?

TAG has been argued to be a suitable grammar formalism, consider:

Examples from experiments

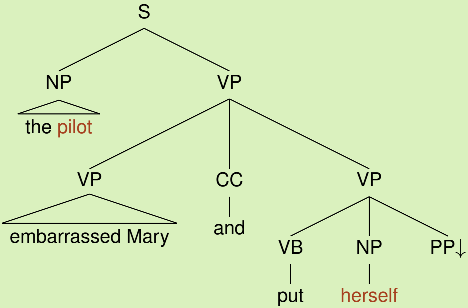


- TAG adjunction can model e.g. connectedness at *herself*.

How suitable is TAG for explaining the experimental results we have seen?

TAG has been argued to be a suitable grammar formalism, consider:

Examples from experiments

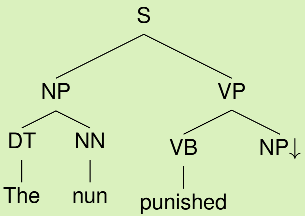


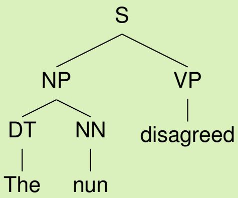
- TAG adjunction can model e.g. connectedness at *herself*.

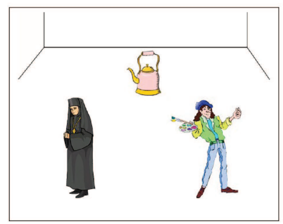
How suitable is TAG for explaining the experimental results we have seen?

TAG has been argued to be a suitable grammar formalism, consider:

Examples from experiments







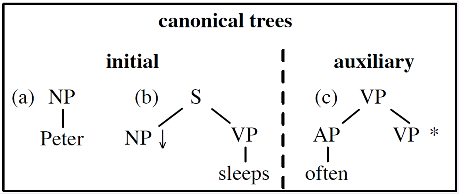
- TAG adjunction can model e.g. connectedness at *herself*.
- Subcategorization frames in elementary trees allow to account for simple prediction effects.

Table of Contents

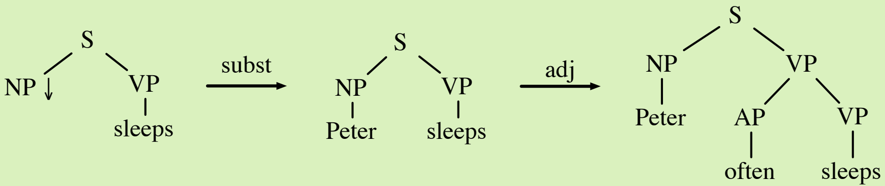
LTAG and strict incrementality

But LTAG is not incremental and does not allow for general connectedness.

Lexicon



LTAG-Derivation for *Peter often sleeps.*



Suggestions for an incremental version of TAG.

When is LTAG not incremental?

- if we have two or more dependents but no head
- if we have a grand-parent and a child, but not parent

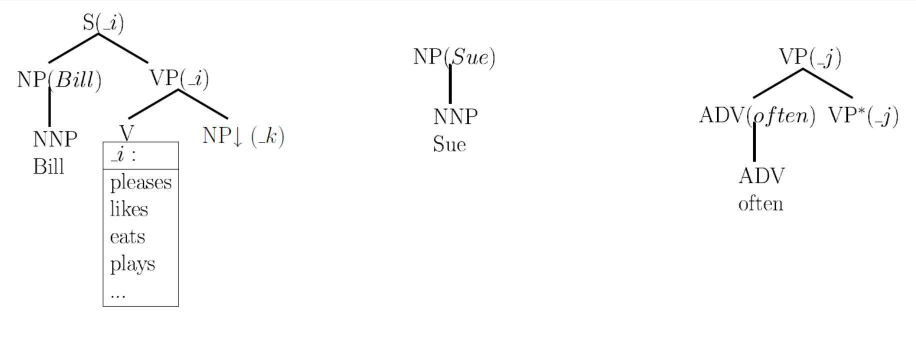
$\Rightarrow$ so need to insert missing structure

Versions of TAG that support full connectedness:

- Dynamic Version of TAG (DVTAG, Mazzei 2005)
- PsychoLinguistically motivated TAG (PLTAG, Demberg 2010)

The DVTAG Formalism (graphics taken from Mazzei 2005)

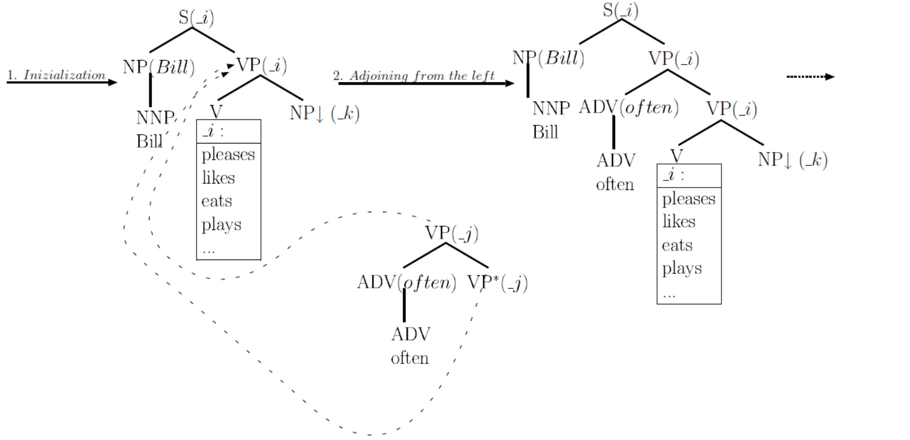
Lexicon



- Lexical items can be underspecified

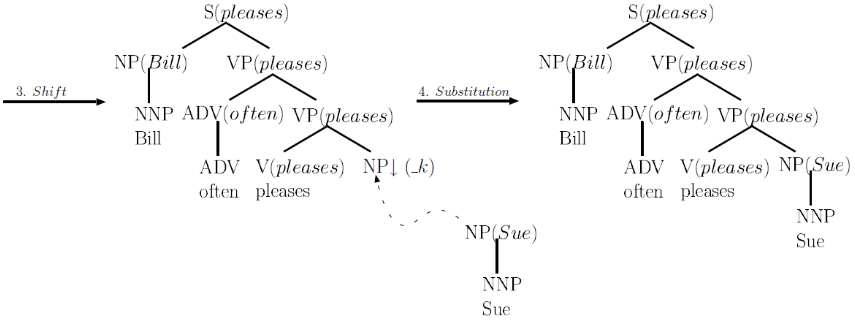
The DVTAG Formalism (graphics taken from Mazzei 2005)

LTAG-Derivation for *Bill often pleases Sue.*



The DVTAG Formalism (graphics taken from Mazzei 2005)

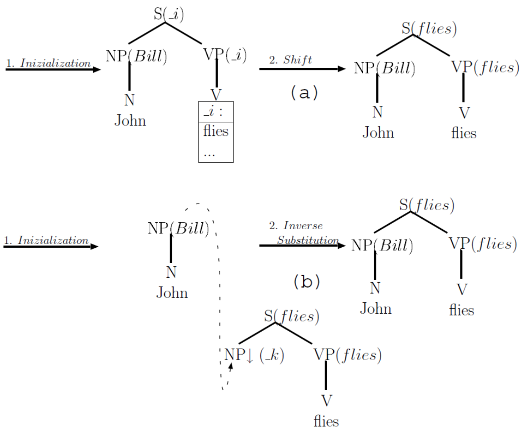
LTAG-Derivation for *Bill often pleases Sue.*



The DVTAG Formalism (graphics taken from Mazzei 2005)

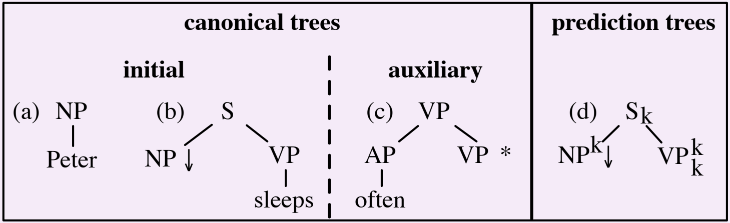
LTAG-Derivation for *Bill often pleases Sue.*

- DVTAG lexicon contains canonical trees plus additional trees which allow for connectedness
- Operations extended to *Inverse Substitution*, *Inverse Adjunction* and *Shift*
- Spurious ambiguity
- Ambiguity is in the lexicon
- DVTAG lexica are huge

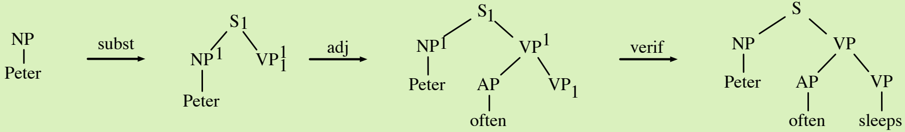


The PLTAG Formalism

Lexicon



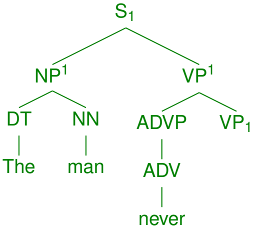
PLTAG-Derivation for *Peter often sleeps.*



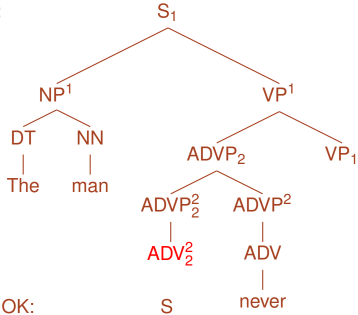
PLTAG – Valid Partial Derivations

Some examples of valid and invalid partial derivations.

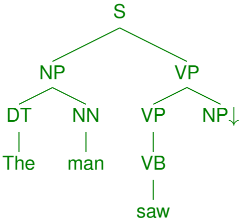
OK:



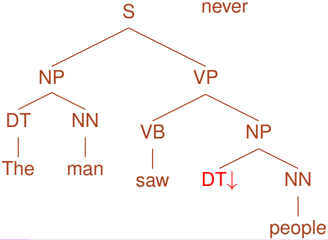
not OK:



OK:

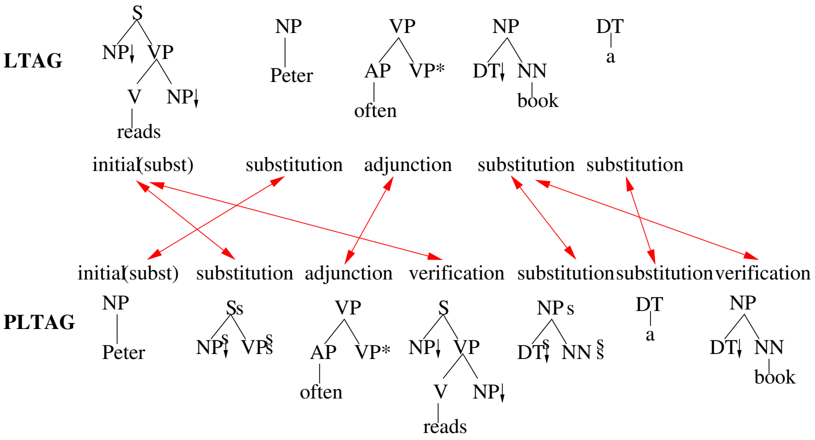


not OK:



The relationship between LTAG and PLTAG

Every LTAG derivation can be translated into an equivalent PLTAG derivation.



PLTAG and LTAG are equally expressive.

Comparison DVTAG and PLTAG

- Both strongly equivalent to LTAG
- Both introduce new operations
  (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than
  DVTAG lexicon (6m templates);
  (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for
  PLTAG only.

Comparison DVTAG and PLTAG

- Both strongly equivalent to LTAG
- Both introduce new operations (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates); (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Comparison DVTAG and PLTAG

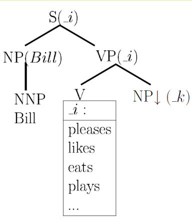
- Both strongly equivalent to LTAG
- Both introduce new operations
(inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates);
(both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Comparison DVTAG and PLTAG

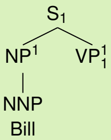
- Both strongly equivalent to LTAG
- Both introduce new operations (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates); (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Example

DVTAG:



PLTAG:



Comparison DVTAG and PLTAG

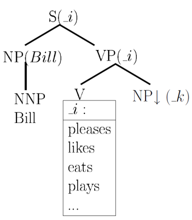
- Both strongly equivalent to LTAG
- Both introduce new operations (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates); (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Comparison DVTAG and PLTAG

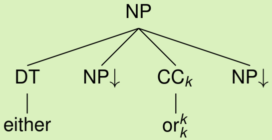
- Both strongly equivalent to LTAG
- Both introduce new operations (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates); (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Example

DVTAG:



PLTAG:

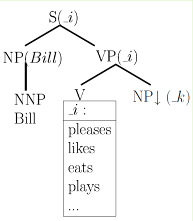


Comparison DVTAG and PLTAG

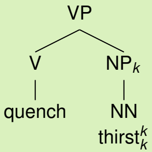
- Both strongly equivalent to LTAG
- Both introduce new operations (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates); (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Example

DVTAG:



PLTAG:



Comparison DVTAG and PLTAG

- Both strongly equivalent to LTAG
- Both introduce new operations
  (inversed subst & adj, shift / verification)
- Prediction explicitly marked in PLTAG
- Prediction granularities differ
- PLTAG lexicon (7k + 3k templates) smaller than DVTAG lexicon (6m templates);
  (both automatically generated from PTB)
- Both formalisms support multi-anchored trees
- Parsing algorithm and broad coverage parser for PLTAG only.

Discussion: Incremental TAG formalisms

- What is tree adjoining grammar?
  - Where's the grammar?
  - What are the standard TAG operations?
- Why is TAG suitable for modelling experimental results?
  - the pilot embarrassed Mary and put herself in an awkward situation
  - either..or
  - prediction of arguments but not adjuncts
- In how far does TAG not support incremental processing?
- What are the mechanisms in DVTAG and PLTAG to allow for incremental derivations?

Table of Contents

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

Step 1: Treebank Conversion

Converted Penn Treebank into TAG format [Xia et al. 2000]

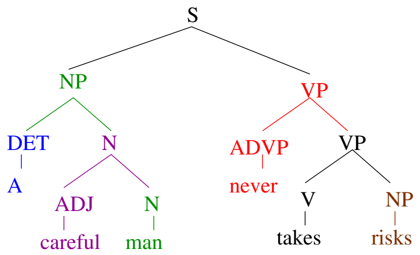
- **Head percolation table** for determining how to cut up the tree into elementary trees [Magerman, 1994]
- **PropBank** [Palmer et al., 2003] for discriminating arguments and modifiers
- **Noun phrase annotation** for the Penn Treebank [Vadas & Curran, 2007]
- Multi-anchored trees (e.g. either... or, pick... up)

The resulting structure is less flat, contains head information and argument / modifier distinction.

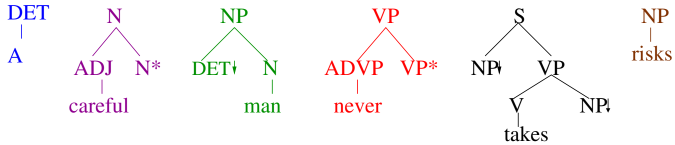
Step 2: Lexicon Induction

Creating the lexicon:

Sentence Tree:



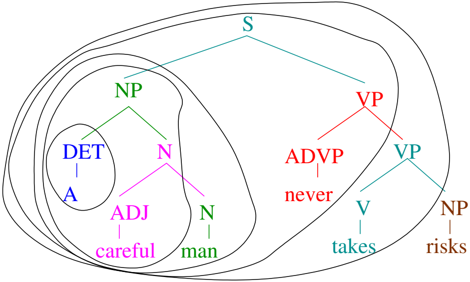
Canonical Lexicon Entries:



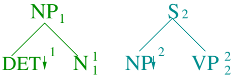
Step 2: Lexicon Induction

Creating the lexicon:

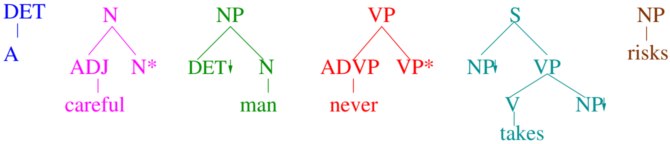
Sentence Tree:



Prediction Lexicon Entries:

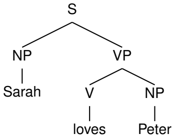


Canonical Lexicon Entries:



But how should we deal with the problem of indetermination?

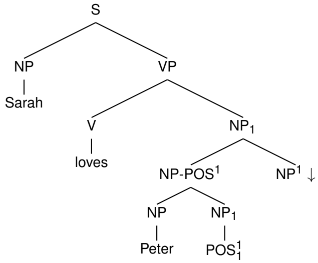
- Prediction trees are not lexicalized.
- So a parser could use infinitely many of them at any time.
- Our approach: only allow combination of prediction trees which have been observed in training.
- Left recursion?!
- People are not able to deal with infinite left recursion either.



Sarah loves Peter

But how should we deal with the problem of indetermination?

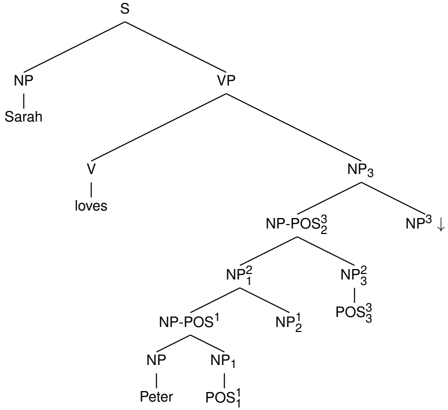
- Prediction trees are not lexicalized.
- So a parser could use infinitely many of them at any time.
- Our approach: only allow combination of prediction trees which have been observed in training.
- Left recursion?!
- People are not able to deal with infinite left recursion either.



Sarah loves Peter's books.

But how should we deal with the problem of indetermination?

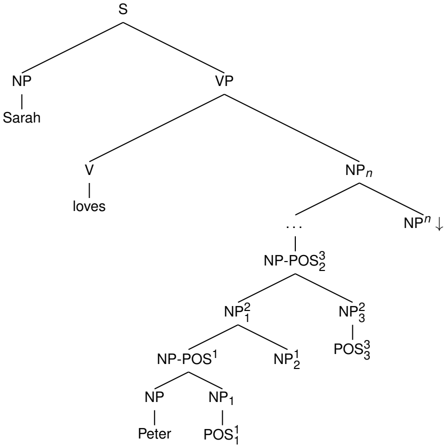
- Prediction trees are not lexicalized.
- So a parser could use infinitely many of them at any time.
- Our approach: only allow combination of prediction trees which have been observed in training.
- Left recursion?!
- People are not able to deal with infinite left recursion either.



Sarah loves Peter's father's books.

But how should we deal with the problem of indetermination?

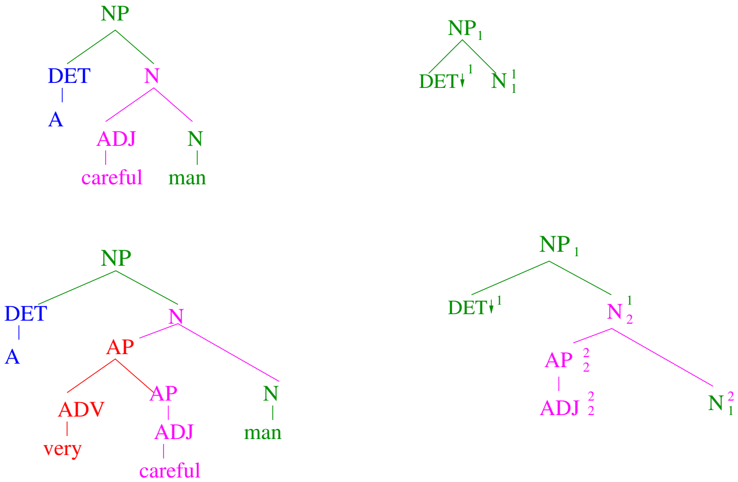
- Prediction trees are not lexicalized.
- So a parser could use infinitely many of them at any time.
- Our approach: only allow combination of prediction trees which have been observed in training.
- Left recursion?!
- People are not able to deal with infinite left recursion either.



Sarah loves Peter's father's neighbor's ... books.

Step 2: Lexicon Induction

Pre-combined prediction trees:



Step 3: The Parsing Algorithm

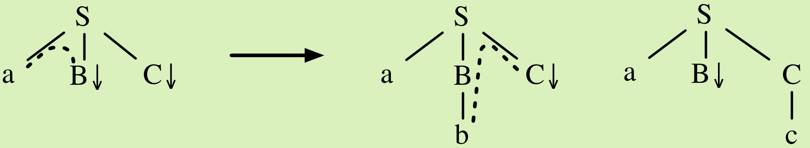
Requirements:

- 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: Substitution



(a) yes (b) no

Step 3: The Parsing Algorithm

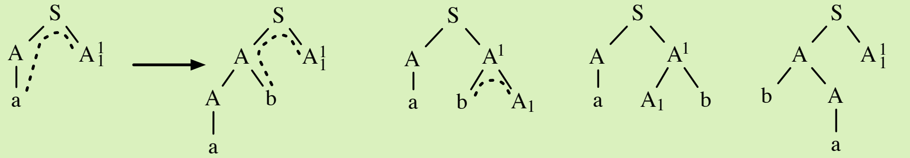
Requirements:

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

Helpful Concept: **Fringes**

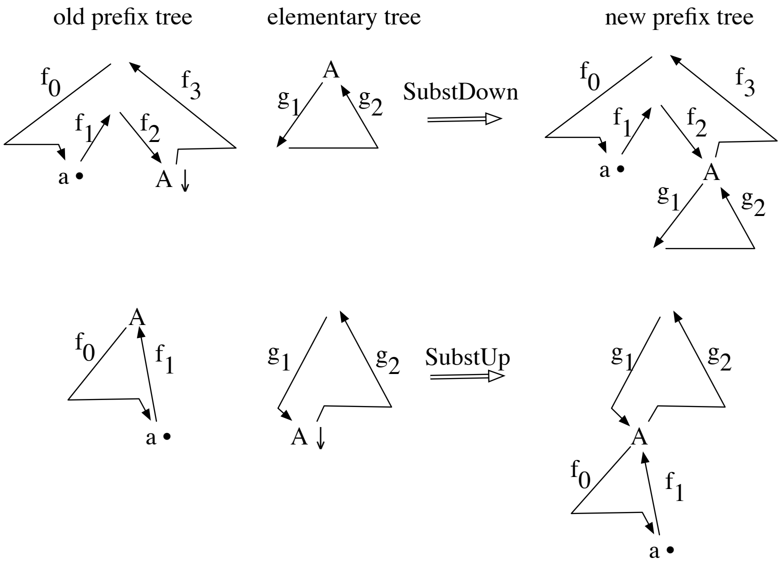
- tree can be described by its depth-first traversal
- only part of incremental tree is relevant at each step

Example: Adjunction

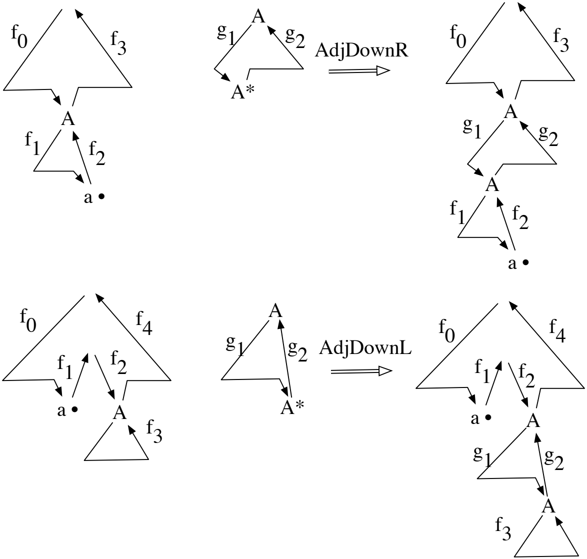


(a) yes (b) yes (c) no (d) no

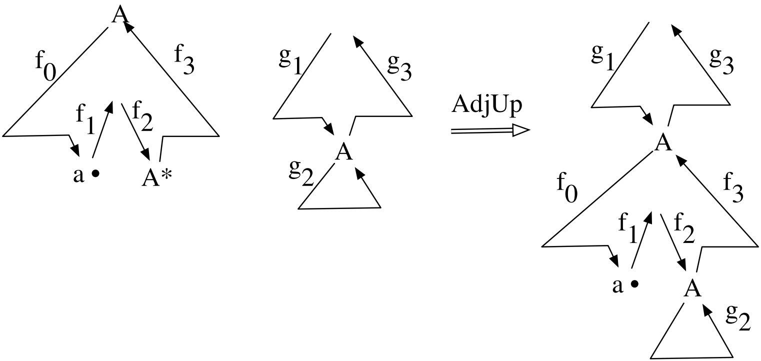
Parsing Rules



Parsing Rules



Parsing Rules

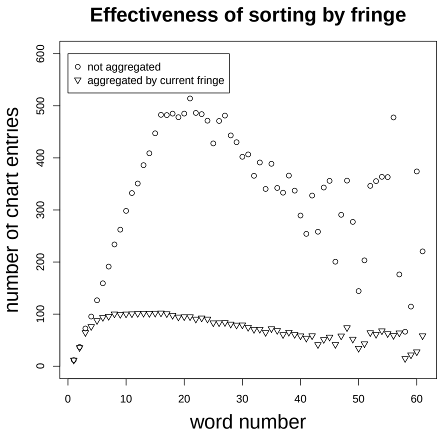


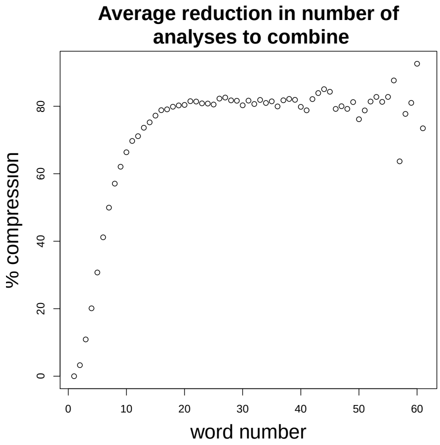
Step 3: The Parsing Algorithm (contd)

Efficiency – or how to get the parser to run for sentences with length > 3

- Trying out all prediction trees at each word is intractable.
  $>$ 1000 prediction trees vs. average ambiguity of canonical trees $\approx$ 50
- Introduce Supertagging for **prediction trees**.
  Parser only tries 20 most promising prediction trees.
- Beam Search
- Use of a chart (trees bundled by identical current fringes)

Effectiveness of Chart Parsing





(Statistics shown with beam search in use - without beam search, a much larger effect can be expected.)

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

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

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation

Probabilities are normalized with respect to other elementary trees ε that can attach at node η in prefix tree β with the same operation.

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

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

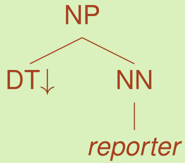
Verification: $\sum_{\varepsilon} P(\varepsilon|\pi_\beta) = 1$

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

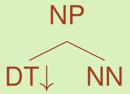
based on [Chiang, 2000]

Explanation

elementary tree ε:



is estimated as template τ_ε:



and lexeme λ_ε: *reporter*

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

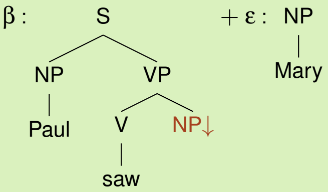
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation



integration node η_β

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

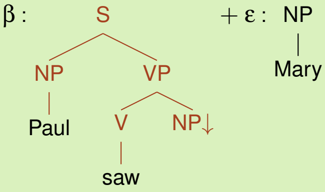
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation



integration tree template τ_η

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

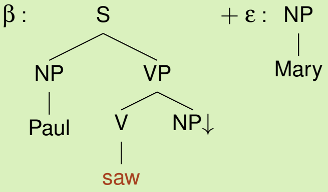
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

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

based on [Chiang, 2000]

Explanation



integration tree lexeme λ_η

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

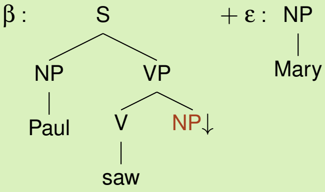
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

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

based on [Chiang, 2000]

Explanation



integration node category c_η

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

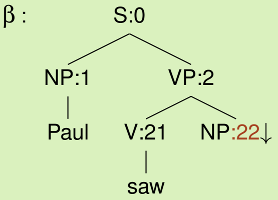
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation



integration node position n_η

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

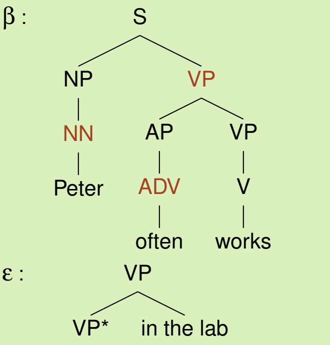
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation



attachment context b_η & level a_f

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

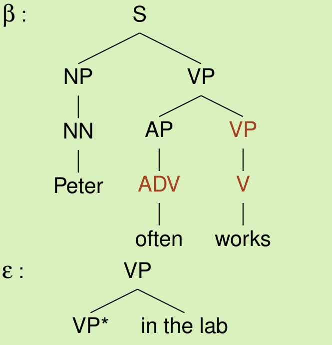
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation



attachment context b_η & level a_f

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Step 4: Probability Model

Generative model (enables us to easily estimate surprisal)

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

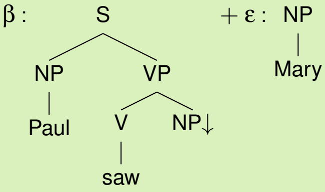
Adjunction: $\sum_{\varepsilon} P(\varepsilon|\eta_\beta) + P(NONE|\eta_\beta) = 1$

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

$$
\begin{aligned}
P(\varepsilon|\eta_\beta) &= P(\tau_\varepsilon|\eta_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_\eta) \\
P(\varepsilon|\pi_\beta) &= P(\tau_\varepsilon|\pi_\beta) \times P(\lambda_\varepsilon|\tau_\varepsilon, \lambda_{\pi_\eta}) \\
P(\eta_\beta) &= P(\tau_\eta, \lambda_\eta, c_\eta, n_\eta, b_\eta, a_f, tm)
\end{aligned}
$$

based on [Chiang, 2000]

Explanation



a trace mark *tm* which marks whether there is a trace at the beginning or end of the fringe

elementary trees	ε	prefix tree	β	prediction trees	π
tree structures	τ	integration point node	η	a tree's head leaf	λ

Parser Evaluation

Parser Performance:

Model	Prec	Recall	F-score	Cov
PLTAG parser	79.43	79.39	79.41	98.09
Pred tree oracle	81.15	81.13	81.14	96.18
No gold POS	77.57	77.24	77.41	98.09

Comparison to other TAG parsers:

Model	incr	con	pred	impl	F
Mazzei et al. (2007)	+	+	+	−	n/a
This work (gold POS)	+	+	+	+	79.4
Kato et al. (2004)	+	+	−	+	79.7
Shen and Joshi (2005)	(+)	−	−	+	(87.4)
Chiang (2000)	−	−	−	+	86.7

Parser Evaluation

Parser Performance:

Model	Prec	Recall	F-score	Cov
PLTAG parser	79.43	79.39	79.41	98.09
Pred tree oracle	81.15	81.13	81.14	96.18
No gold POS	77.57	77.24	77.41	98.09

Comparison to other TAG parsers:

Model	incr	con	pred	impl	F
Mazzei et al. (2007)	+	+	+	–	n/a
This work (gold POS)	+	+	+	+	79.4
Kato et al. (2004)	+	+	–	+	79.7
Shen and Joshi (2005)	(+)	–	–	+	(87.4)
Chiang (2000)	–	–	–	+	86.7

Discussion on Incremental TAG parsing

- What difficulty arises in parsing from the prediction trees not being lexicalized?
- What's the problem with left recursion?
- Which measures were taken to make parsing tractable?
- In how far is incremental parsing more difficult / error-prone than non-incremental parsing?

The Linking Theory

Before I get to the linking theory, I will briefly introduce two theories of processing difficulty, that our processing theory is inspired by:

- Surprisal
- Dependency Locality Theory

Surprisal [Hale 2001, 2003; Levy, 2008]

"Forward-looking measure"

Key idea: Processing difficulty at $w_i \propto$ amount of Surprisal at perceiving w_i

How to calculate surprisal:

- Calculate prefix probabilities:

$$pp_{w_n} = -log \sum_{T \in Trees} p(T|w_1 \dots w_n)$$

- Surprisal s of word w_n: $s_{w_n} = pp_{w_n} - pp_{w_{n-1}}$
- [Hale 2001; Roark 2001, 2009; Levy 2008]

Explains **Prediction Effects** (either..or processing) and **Anti-locality effects**.

Surprisal [Hale 2001, 2003; Levy, 2008]

"Forward-looking measure"

Key idea: Processing difficulty at w_i $\propto$ amount of Surprisal at perceiving w_i

How to calculate surprisal:

- Calculate prefix probabilities:
$$pp_{w_n} = -log \sum_{T \in Trees} p(T|w_1 \ldots w_n)$$
- Surprisal s of word w_n: $s_{w_n} = pp_{w_n} - pp_{w_{n-1}}$
- [Hale 2001; Roark 2001, 2009; Levy 2008]

Explains **Prediction Effects** (either..or processing) and **Anti-locality effects**.

DLT – Integration Cost

"Backward looking measure"

Key idea: predicts difficulty based on

- heads and their dependents: integration cost occurs at heads of phrases
- discourse referents: number of DR between head and its dependent

SRC

+2

	The	reporter	who	*t*	attacked	the	senator	admitted	the	error.
DR:	0	1	0		1	0	1	1	0	1
IC:	0	1	0		1	0	1	3	0	1

ORC

+2 +2

	The	reporter	who	the	senator	attacked	*t*	admitted	the	error.
DR:	0	1	0	0	1	1		1	0	1
IC:	0	1	0	0	1	3		3	0	1

Explains locality effects: SRC/ORC asymmetry, center embedding

Comparison of DLT and Surprisal

Comparison

Surprisal:	"forward-looking"	difficulty caused by violated expectations
DLT:	"backward-looking"	difficulty caused by long-distance integrations

Effects	Surprisal	DLT
Either-or Prediction	+	–
English Relative Clause	–	+
German Relative Clause	+	–
Facilitating Ambiguity	+	–
Storage Cost Effects	–	+
Center Embedding	NA	+

Conclusions:

- The two theories explain different parts of the data – can they be combined in a unified theory?

[Demberg & Keller, 2009, Cognition]

Step 5: The Linking Theory

Translates the PLTAG parser states into processing difficulty:

- Big changes in the probability distribution cause processing difficulty.
- Each of these predicted trees π has a time-stamp t.
- At verification time, the tree's nodes' decay d is calculated based on the time stamp (recently-accessed structures are easier to integrate).

Estimation of Processing Difficulty D_{w_i}

$$D_{w_i} = \overbrace{-\log \sum_{\beta_{w_1 \dots w_i}} P(\beta_{w_1 \dots w_i}) + \log \sum_{\beta_{w_1 \dots w_{i-1}}} P(\beta_{w_1 \dots w_{i-1}})}^{\text{Surprisal}}$$
$$\underbrace{-\log \sum_{\pi} P(\pi)^{(1-d^{t_\pi})}}_{\text{Verification Cost}}$$

Table of Contents

Psycholinguistic Evaluation

Linking Theory translates the PLTAG parsing process into processing difficulty estimates:

- Based on parallel, probabilistic parsing with a generative model.
- Big changes in the probability distribution cause processing difficulty (surprisal).
- Long-distance dependencies as evident from verification of nodes which were predicted a long time ago, cause processing difficulty.

Evaluation of Processing Theory based on PLTAG

- Broad-Coverage Evaluation on Dundee Corpus
- Psycholinguistic Case Studies

The Dundee Corpus

Dundee eye-tracking corpus [Kennedy et al. 2003]

- ca. 51.000 words of British newspaper articles (from The Independent)
- 10 subjects

Broad-Coverage Evaluation on Dundee Corpus

	Prediction Theory	
Predictor	Coef	Sig
Intercept	230.31	***
WordLength	3.93	***
WordFrequency	-8.92	***
PrevWordFreq	-4.53	***
PrevWordFixated	-30.96	***
LaunchDistance	-0.88	***
LandingPosition	70.34	***
WordNoInSentence	-0.14	***
BigramProb	-3.06	***
PredictTheory	0.38	**
WordLength:Freq	-1.44	***
WordLength:Posit	23.06	***

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$

Broad-Coverage Evaluation on Dundee Corpus

	Prediction Theory	
Predictor	Coef	Sig
Intercept	230.31	***
WordLength	3.93	***
WordFrequency	-8.92	***
PrevWordFreq	-4.53	***
PrevWordFixated	-30.96	***
LaunchDistance	-0.88	***
LandingPosition	70.34	***
WordNoInSentence	-0.14	***
BigramProb	-3.06	***
PredictTheory	0.38	**
StructSurprisal		
LexicalSurprisal		
WordLength:Freq	-1.44	***
WordLength:Posit	23.06	***

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$

Broad-Coverage Evaluation on Dundee Corpus

	Prediction Theory		Structural Surprisal		Lexical Surprisal	
Predictor	Coef	Sig	Coef	Sig	Coef	Sig
Intercept	230.31	***	230.44	***	230.43	***
WordLength	3.93	***	3.99	***	3.97	***
WordFrequency	-8.92	***	-8.88	***	-8.89	***
PrevWordFreq	-4.53	***	-4.50	***	-4.53	***
PrevWordFixated	-30.96	***	-30.95	***	-30.94	***
LaunchDistance	-0.88	***	-0.88	***	-0.88	***
LandingPosition	70.34	***	70.38	***	70.37	***
WordNoInSentence	-0.14	***	-0.14	***	-0.14	***
BigramProb	-3.06	***	-2.99	***	-3.03	***
PredictTheory	0.38	**				
StructSurprisal			0.21			
LexicalSurprisal					-0.02	
WordLength:Freq	-1.44	***	-1.38	***	-1.39	***
WordLength:Posit	23.06	***	22.98	***	23.00	***

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$

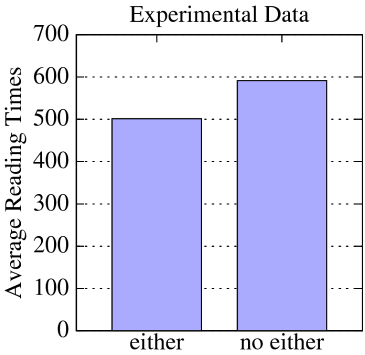
Either..or processing

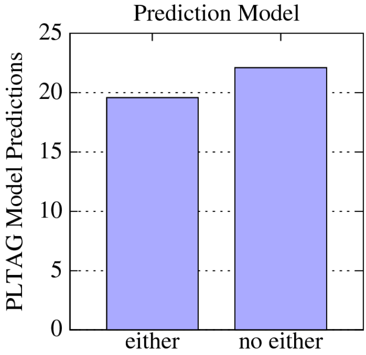
Evaluation on 48 sentences from experiment by [Staub and Clifton, 2006].

Example sentences:

Peter read either a book ***or an essay*** *in the school magazine.*

Peter read a book ***or an essay*** *in the school magazine.*





Summary PLTAG Syntax Modelling

PLTAG syntax:

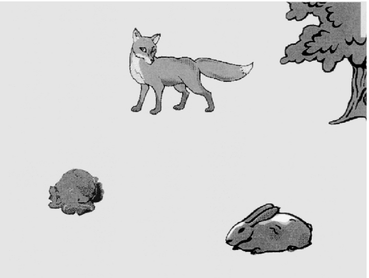
- Models incrementality and prediction at syntax level
- Successfully replicates patterns observed in naturalistic data
- as well as psycholinguistic experiments

Table of Contents

Limitations of syntax-only PLTAG model

But with syntax only, can't explain

- anticipatory eye movements in *Den Hasen frisst gleich der Fuchs.*



- Most garden path sentences, e.g. *The horse raced past the barn fell.*

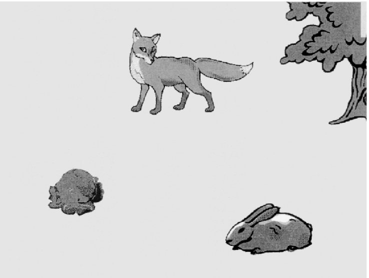
Example: Semantic plausibilty affects syntax

The doctor sent for the patient arrived.
The flowers sent for the patient arrived.

Limitations of syntax-only PLTAG model

But with syntax only, can't explain

- anticipatory eye movements in *Den Hasen frisst gleich der Fuchs.*



- Most garden path sentences, e.g. *The horse raced past the barn fell.*

Example: Semantic plausibilty affects syntax

The doctor <u>sent</u> for the patient arrived.
The flowers <u>sent</u> for the patient arrived.

Incremental Semantics

Desired properties of semantics for PLTAG:

- simultaneous construction of syntax and semantics
- reflect the incrementality of the syntax
- direct syntax-semantics interface
- semantics that are useful in combination with some form of compositional distributional semantics, so we can calculate e.g. semantic surprisal