CMPT-882: Statistical Learning of Natural Language

Lecture #11

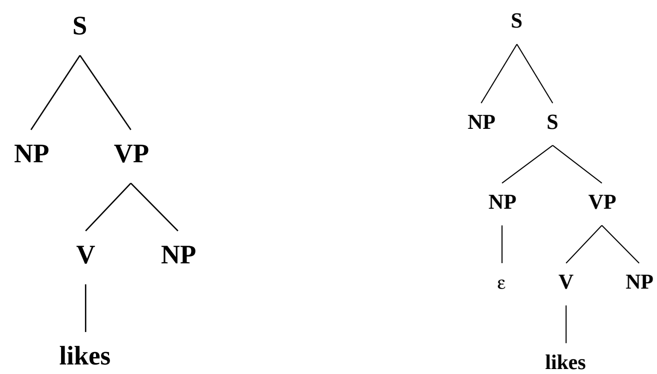
Anoop Sarkar
anoop@cs.sfu.ca
http://www.sfu.ca/~anoop

- *Supertagging: an approach to almost parsing*. Srinivas Bangalore and Aravind K. Joshi. (1999)

- slides mostly taken from material prepared by B. Srinivas

Descriptions of Primitives

- Simple: likes/V
- Complex:

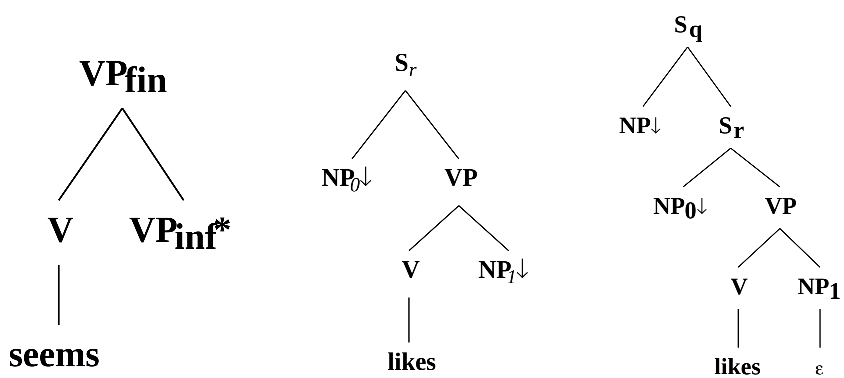


- Complexity of Descriptions
  - Complex constraints operate locally
  - Implications for statistical computations

Extended Domain of Locality (EDL)

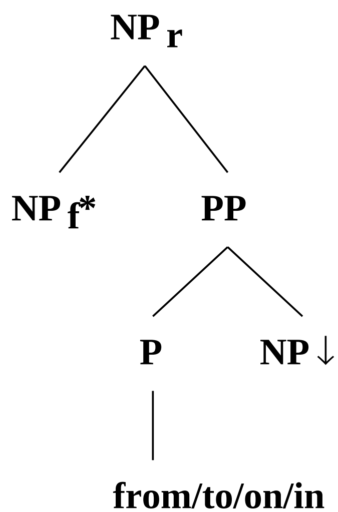
1. Every elementary structure must contain all and only the arguments of the anchor.

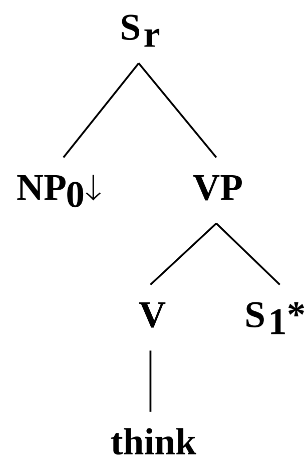
2. There is one elementary structure for each syntactic environment a lexical item may appear in.



Factoring of Recursion

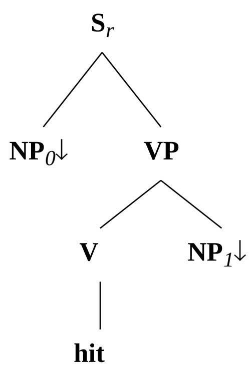
- Recursion is factored away from the domain for the statement of dependencies.

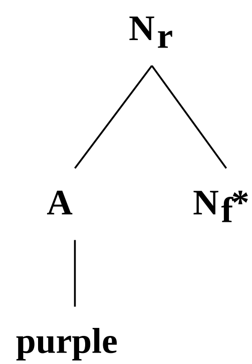




Lexicalized Tree-Adjoining Grammars

- Primary objects of LTAGs are Elementary Trees.
- Lexicalized, Extended Domain of Locality, Factoring of Recursion.
- Elementary Trees are of two types
  - Initial Trees and Auxiliary Trees

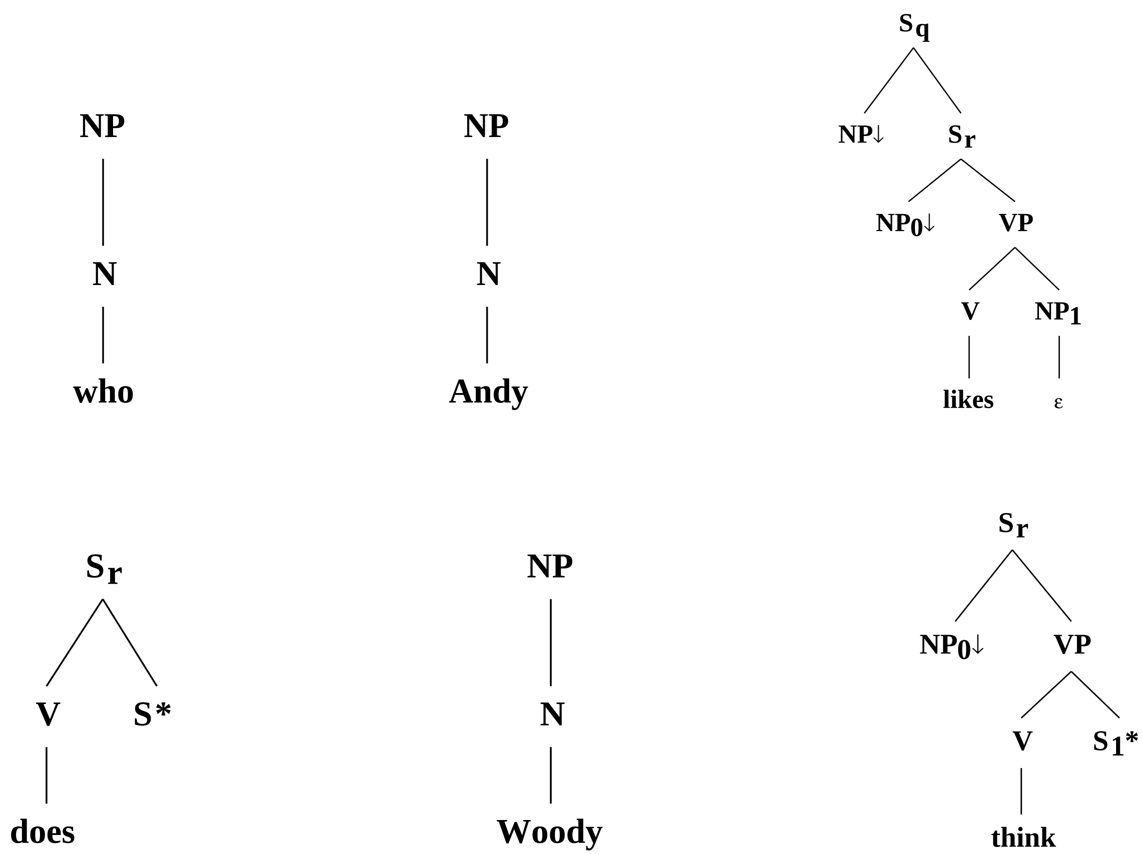




- Substitution and Adjunction are two combining operations.

Example

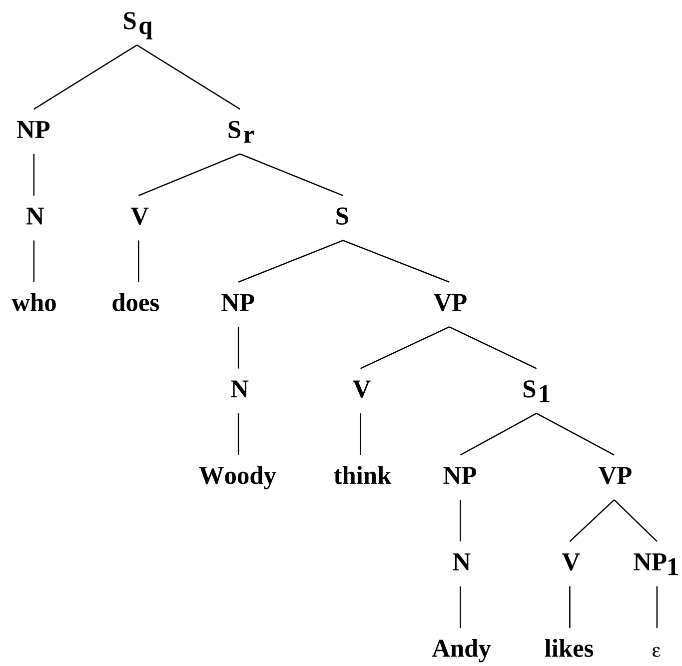
who does Woody think Andy likes



Example

who does Woody think Andy likes

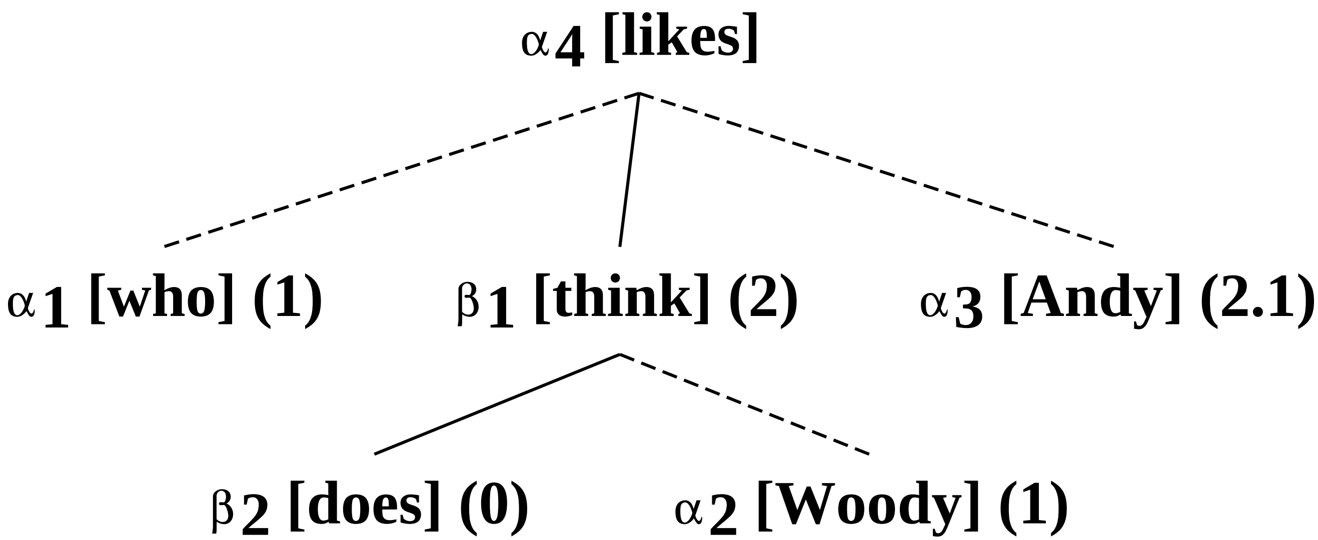
- Derived Tree



Example

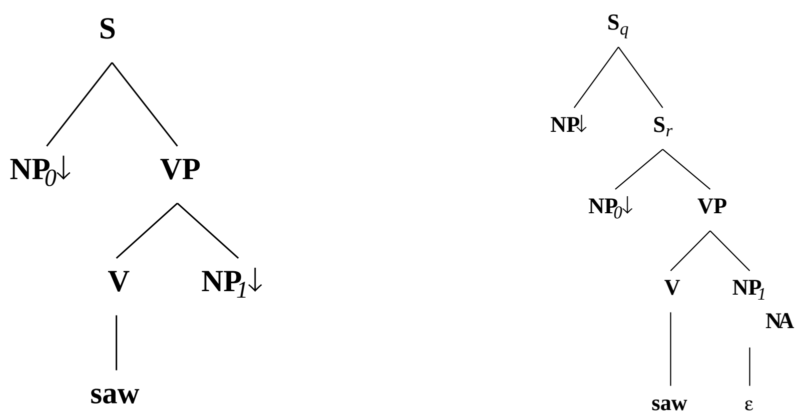
who does Woody think Andy likes

- Derivation Tree

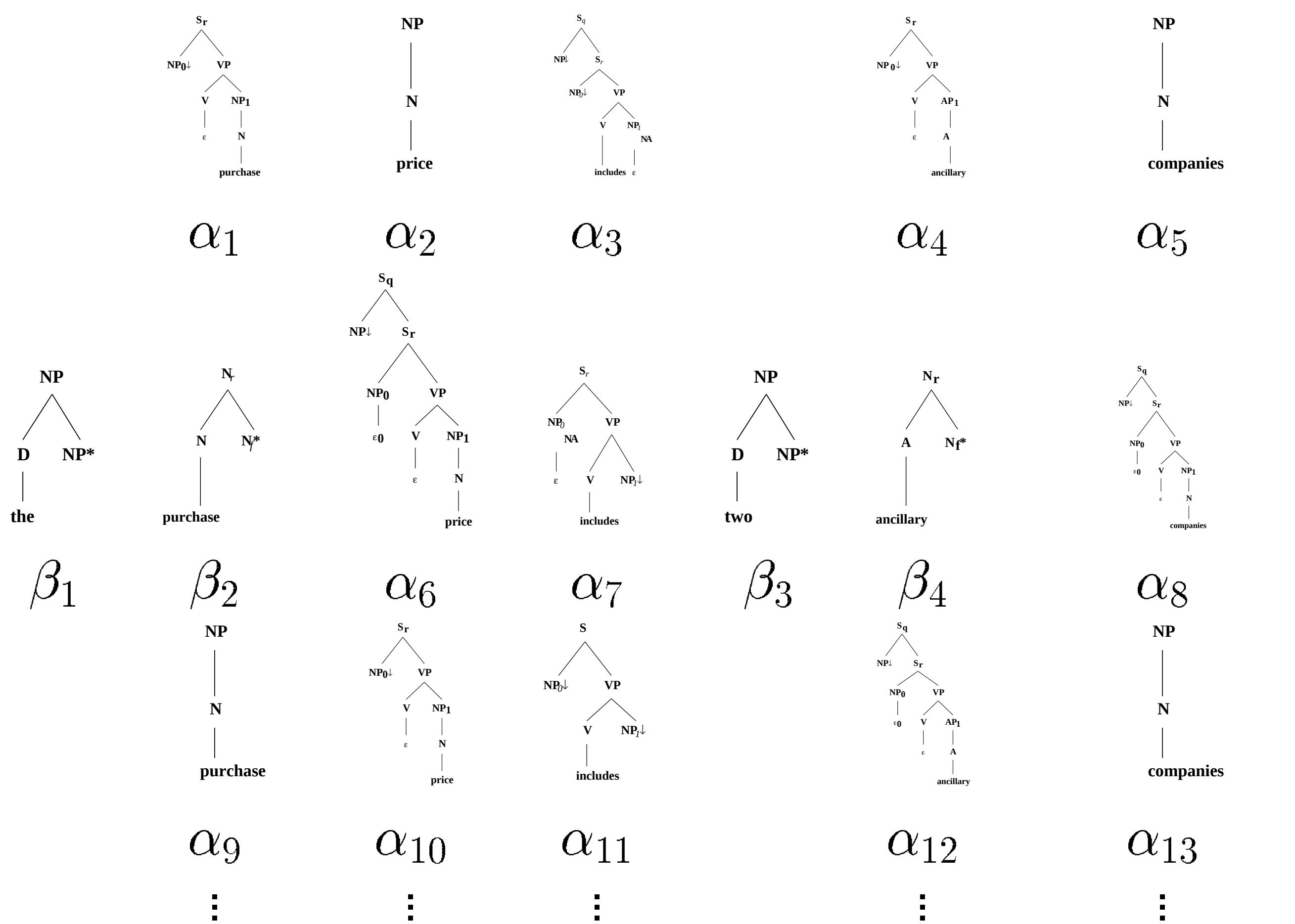


Supertags

- Elementary trees are called Supertags.
- Localize head-complement and filler-gap dependencies.



- Supertags
  - more complex than part-of-speech tags
  - more supertags associated with word than part-of-speech tags



the purchase price includes two ancillary companies.

Supertagging

Sent:	the	purchase	price	includes	two	ancillary	companies.
Initial		α_1	α_2	α_3		α_4	α_5
Assig.	β_1	β_2	α_6	α_7	β_3	β_4	α_8
		α_9	α_{10}	α_{11}		α_{12}	α_{13}
		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Final Assig.	β_1	β_2	α_2	α_{11}	β_3	β_4	α_{13}

- Supertagging: Select most appropriate supertag for each word.
- Supertag disambiguation before parsing.
- Supertag disambiguation results in an "almost parse".

Models for Supertag Disambiguation

- N-gram models
  - Trigram model
  - Head trigram model
- Dependency based model (COLING 94)
  - More like full parsing

Trigram Model for Supertagging

- Find the most likely Supertag sequence for a given word sequence.

$$\hat{T} = \text{argmax}_T \ \text{Pr}(T_1, T_2, \ldots, T_N | W_1, W_2, \ldots, W_N)$$

- By Bayes Rule

$$\hat{T} = \text{argmax}_T \ \frac{\text{Pr}(W_1, W_2, \ldots, W_N | T_1, T_2, \ldots, T_N) * \text{Pr}(T_1, T_2, \ldots, T_N)}{\text{Pr}(W_1, W_2, \ldots, W_N)}$$

- Since the word sequence is given

$$\hat{T} = \text{argmax}_T \ \text{Pr}(W_1, W_2, \ldots, W_N | T_1, T_2, \ldots, T_N) * \text{Pr}(T_1, T_2, \ldots, T_N)$$

Trigram Model for Supertagging

- Contextual probability

$$\Pr(T_1,T_2,\ldots,T_N) \approx \Pi_{i=1}^{N} \Pr(T_i \mid T_{i-2}, T_{i-1})$$

- Word Emit probability

$$\Pr(W_1,W_2,\ldots,W_N|T_1,T_2,\ldots,T_N) \approx \Pi_{i=1}^{N} \Pr(W_i \mid T_i)$$

- Trigram Model

$$\hat{T} = \mathrm{argmax}_T \ \Pi_{i=1}^{N} \Pr(T_i \mid T_{i-2}, T_{i-1}) * \Pr(W_i \mid T_i)$$

where T_i is the supertag for word W_i.

- Unseen events
  - Good-Turing discounting with Katz's Back-off Model.

Training and Test Data

- Training Set A:
  - 200,000 word-supertag pairs
  - collected by bootstrapping and hand correction.
  - WSJ sections 15 through 18
- Training Set B:
  - 1,000,000 word-supertag pairs
  - collected by heuristically mapping from Penn Treebank
  - WSJ sections 0-19 and 21-24
- Test Set: section 20 of WSJ.

Performance of Trigram Supertagger

- Performance of the supertagger on the WSJ corpus
- Correct supertag implies that a word is assigned the same supertag as it would be in the correct parse of the sentence.

Size of training corpus	Size of test corpus	# of words correctly supertagged	% correct
Baseline	47,000	35,391	75.3%
200,000	47,000	42,723	90.9%
1 Million	47,000	43,334	92.2%

- Errors:
  - PP attachment
  - Verbs with more than two complements.

Head Trigram Model for Supertagging

- Head Trigram Model

  $\hat{T} = \text{argmax}_T \, \Pi_{i=1}^{N} \, \Pr(T_i \mid T_{H_{i-2}}, T_{H_{i-1}})*\Pr(W_i \mid T_i)$

- ...saw the big man with ...

  Trigram Model computes:Pr(with$|T$)*Pr($T|T_{man}$,T_{big})

  Head Trigram Model computes:Pr(with$|T$)*Pr($T|T_{man}$,T_{saw})

- Head identification
- Head Propagation

  (1)

  $$\begin{array}{lll} \text{Initialize: } (H_{-2}, H_{-1}) & = (-2, -1) & \\ \text{Update : } (H_{i-1}, H_i) & = (H_{i-2}, H_{i-1}) & \text{if } W_i \text{ is not a head word} \\ & = (H_{i-1}, i) & \text{if } W_i \text{ is a head word} \end{array}$$

Head Trigram Model for Supertagging

- Head-word tagger: Identify the head words given a sentence

Size of training corpus	Size of test corpus	# of words correctly supertagged	% correct
Baseline	47,000	38,258	81.4%
1 Million	47,000	42,864	91.2%

- Performance of the head trigram supertagger:

Size of training corpus	Size of test corpus	# of words correctly supertagged	% correct
Baseline	47,000	35,391	75.3%
1 Million	47,000	40,890	87%

Chunking using Supertagged output

- Noun chunks
  - Non-recursive noun phrases
- Scan right to left starting with the noun initial supertag and collect all functors of a noun or a noun modifier.
- Examples:
  - New Jersey Turnpike Authority
  - its increasingly rebellious citizens
  - two $ 400 million real estate mortgage investment conduits

Chunking using Supertagged output

- Noun group performance: Comparison with Ramshaw and Marcus (1995).

System	Training Size	Recall	Precision
R&M	Baseline	81.9%	78.2%
R&M	200K	92.3%	91.8%
Supertags	Baseline	74.0%	58.4%
Supertags	200K	93.0%	91.8%
Supertags	1000K	93.8%	92.5%

- Internal structure of the noun phrases.

Chunking using Supertagged output

- Verbs chunks
  - Sequence of verbs or verbal modifiers.
- Scan left to right starting with the verb or verbal modifier supertag and collect all functors of a verb or a verb modifier.
- Examples
  - would not have been stymied
  - did n't even care
  - just beginning to collect

Chunking using Supertagged output

- Verb group performance: Comparison with Ramshaw and Marcus (1995).

System	Training Size	Recall	Precision
R&M	Baseline	60.0%	47.8%
R&M	200K	88.5%	87.7%
Supertags	Baseline	76.3%	67.9%
Supertags	200K	86.5%	91.4%

- Differences in verb groups
  - (has involved simply buying) (and then holding)
  - predicatives
- Internal structure of the Verb phrases.
  - Sentential complement information.

Lightweight Dependency Analyzer

- Information associated with supertags:
  - Slots: substitution and foot nodes
- Fillers of substitution nodes are argument words and fillers of foot nodes are modified words.
- Two pass algorithm:
  - Establish dependencies for auxiliary supertags
  - Mark all the words that serve as arguments as unavailable for the next pass
  - Establish dependencies for initial supertags.
- Establish dependencies – local search
  - first supertag with root node same as the argument type.

Lightweight Dependency Analyzer

The implicit interior state of the iteration over the hash table entries has dynamic extent

Pos	Word	Supertag	Slot req.	Pass 1	Pass 2	Dep Links
0	The	α_1	–	–	–	–
1	implicit	β_2	+N*	2*		2*
2	interior	β_2	+N*	3*		3*
3	state	α_2	-D.		0.	0.
4	of	β_1	-NP* +NP.	3* 6.		3* 6.
5	the	α_1	–	–	–	–
6	iteration	α_2	-D.		5.	5.
7	over	β_1	-NP* +NP.	6* 11.		6* 11.
8	the	α_1	–	–	–	–
9	hash	β_3	+N*	10*		10*
10	table	β_3	+N*	11*		11*
11	entries	α_2	-D.		8.	8.
12	has	α_3	+NP. -NP.		3. 14.	3. 14.
13	dynamic	β_2	+N*	14*		14*
14	extent	α_4	–	–	–	–

Lightweight Dependency Analyzer

- Trigram supertagger trained on one million supertagged WSJ words.
- Performance on pairwise dependency links
  - A link in output must be in gold standard

Corpus	# of dependency links	# produced by LDA	# correct	Recall	Precision
Brown	140,280	126,493	112,420	80.1%	88.8%
WSJ	47,333	41,009	38,480	82.3%	93.8%

Lightweight Dependency Analyzer

- Test corpus was parsed using the XTAG system
- Performance on pairwise dependency links

Training Size (words)	Test Size (words)	Recall	Precision
200,000	12,000	83.6%	83.5%
1,000,000	12,000	85.0%	85.0%

- Performance at the sentence level
  (Matching against XTAG derivation trees)

	% sentences with 0 errors	% sentences with $\leq$1 error	% sentences with $\leq$2 errors	% sentences with $\leq$3 errors
200K	35%	60.3%	78%	89.8%
1M	40%	63.0%	80.1%	91.0%

Dependency Based Model

- Data Representation

(P.O.S,Supertag)	Direction of Dependent Supertag	Ordinal position	Dependent Supertag	Prob
(D,α_1)	()	-	-	-
(N,α_{13})	()	-	-	-
(N,α_2)	$(-)$	-1	α_1	0.975
(V,α_{15})	$(-, +)$	-1	α_{13}	0.700
(V,α_{15})	$(-, +)$	1	α_{13}	0.420

- For example, the fourth entry reads
  - the supertag α_{15}, anchored by a verb (V)
  - has a left and a right dependent $(-,+)$
  - the first word to the left (-1) with the supertag α_{13} serves as a dependent and
  - the strength of this association is represented by the probability 0.700

Dependency Based Model

Sent:	the	purchase	price	includes	two	ancillary	companies.
POS:	D	N	N	V	D	A	N
Initial Assig.	α_1	α_2	α_3	α_4	β_1	α_5	α_6
	α_7	β_2	α_8	α_9	α_{10}	β_3	α_{11}
	α_{12}	α_{13}	α_{14}	α_{15}	α_{16}	α_{17}	α_{18}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Final Assig.	α_1	β_2	α_3	α_{15}	α_{10}	β_3	α_6

- Every anchor must find its dependents.
- Every dependent must be linked to a anchor.
- No two dependency arcs may cross one another.

Dependency Based Model

- Performance results on Wall Street Journal (WSJ) sentences

Criterion	Total number	Number correct	% correct
Supertags	915	707	77.26%
Dependency links	815	620	76.07%

- Issues:
  - Needs a parsed corpus as training material
  - Attempts at getting a complete linkage
  - Worst-case complexity: $O(n^3)$
  - Lots of parameters to train: $O(S^{2*D^A})$
  - More like parsing than not