

---

# Advanced Natural Language Processing

## Lecture 13

### Supertagging

Frank Keller (slides by Bonnie Webber)

21 October 2011



## Questions to be answered

1. What are supertags?
2. How is a (super)tagset chosen and frequencies collected for each element?
3. How is a string supertagged?
4. How are supertags used?

## PoS tags and Supertags

Phrase structure grammars (PSGs) for NLP are generally specified in terms of the parts-of-speech (PoS) that lexical items realise – eg. *tv*, *det*, *n* in

$VP \rightarrow tv \ NP$

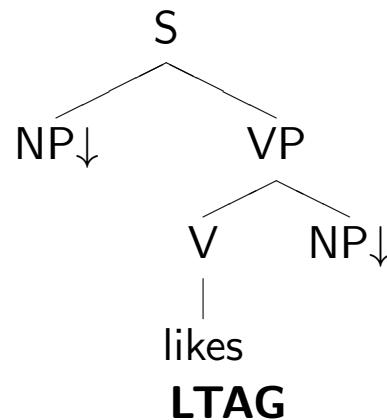
$NP \rightarrow det \ n$

...

## PoS tags and Supertags

LTAG and Categorical Grammar assign syntactically richer specifications to lexical items. Instead of the category *tv* for *likes*, we get *inter alia*

*likes*:  $(S \setminus NP) / NP$



**Categorical Grammar**

Such specifications of lexical categories can be termed **supertags**.

## How is a supertag set collected?

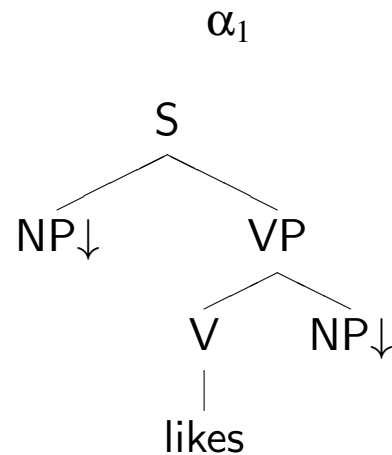
PoS tagsets are selected by corpus developers:

1. CLAWS tag (used for BNC); 62 tags;
2. Brown tag (used for Brown corpus); 87 tags;
3. Penn tag set (used for the Penn Treebank); 45 tags.

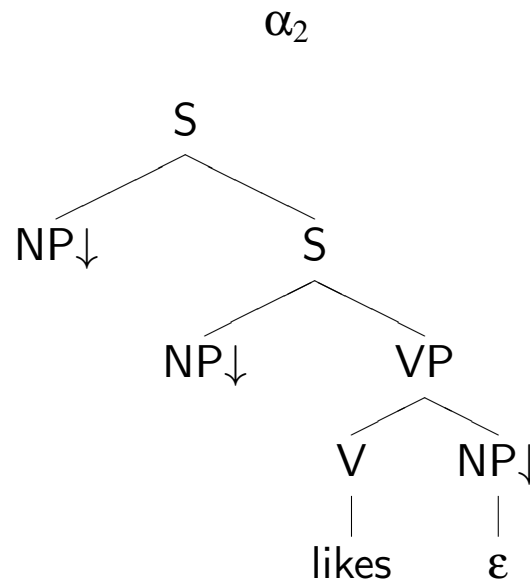
Category	Examples	CLAWS	Brown	Penn
Adjective	happy, bad	AJ0	JJ	JJ
Noun singular	woman, book	NN1	NN	NNS
Noun plural	women, books	NN2	NN	NN
Noun proper singular	London, Michael	NP0	NP	NNP
Noun proper plural	Finns, Hearts	NP0	NPS	NNPS
reflexive pro	itself, ourselves	PNX		
plural reflexive pro	ourselves, . . .		PPLS	
Verb past participle	given, found	VVN	VBN	VBN

## How is a supertag set collected?

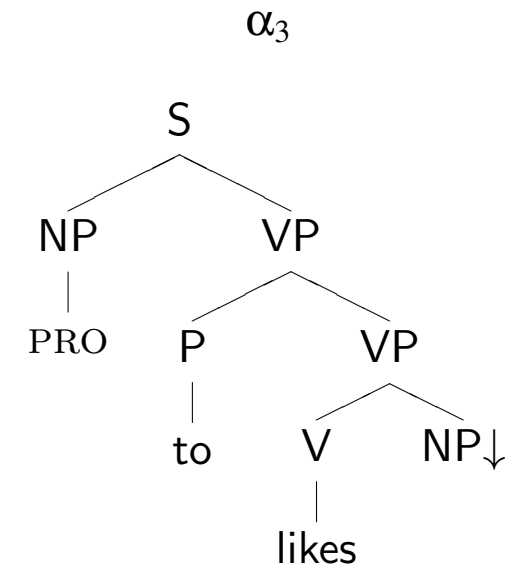
For LTAG, research on the theory specifies some of the lexical categories (supertags) needed to handle different constructions.



*likes*: **simple transitive**



*likes*: **topicalization**



*likes*: **control**

## How is a supertag set collected?

An an example, consider the set of supertags for a whole sentence:

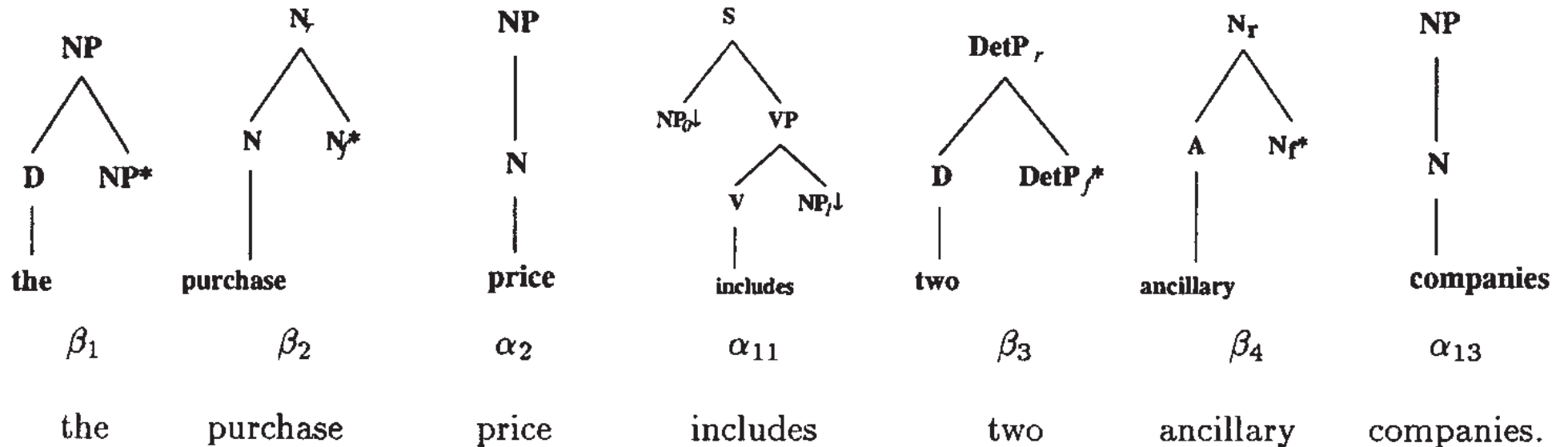


Figure from Bangalore and Joshi (1999).

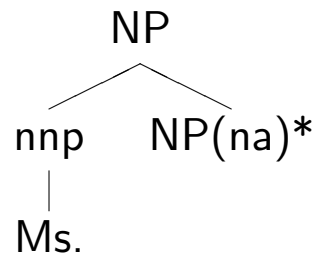
## How is a supertag set collected?

For a wide-coverage LTAG, the Supertag set needs to be derived from a parsed corpus. This was done in two ways (Bangalore and Joshi, 1999):

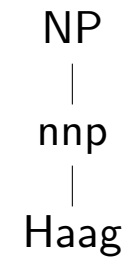
1. Parsing three corpora: Penn TreeBank (sentences  $\leq 15$  words), IBM manual and ATIS, to get their *derivation trees*, where supertags were then taken from the correct derivation tree for each sentence. (**Problem**: Incomplete coverage)
2. Inferring supertags from the Penn TreeBank sentences, using heuristics based on local tree contexts such as a word's dominating nodes (parent, grandparent, etc.), its siblings (left and right), and the siblings of its parent. (**Problem**: Imperfect)

## Derived trees and Derivation trees

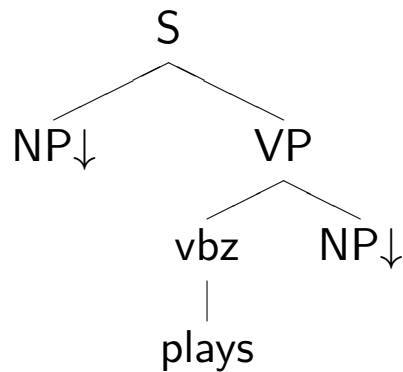
$\beta$ (Ms.):



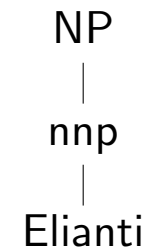
$\alpha$ (Haag):



$\alpha$ (plays):



$\alpha$ (Elianti):



### Elementary Trees



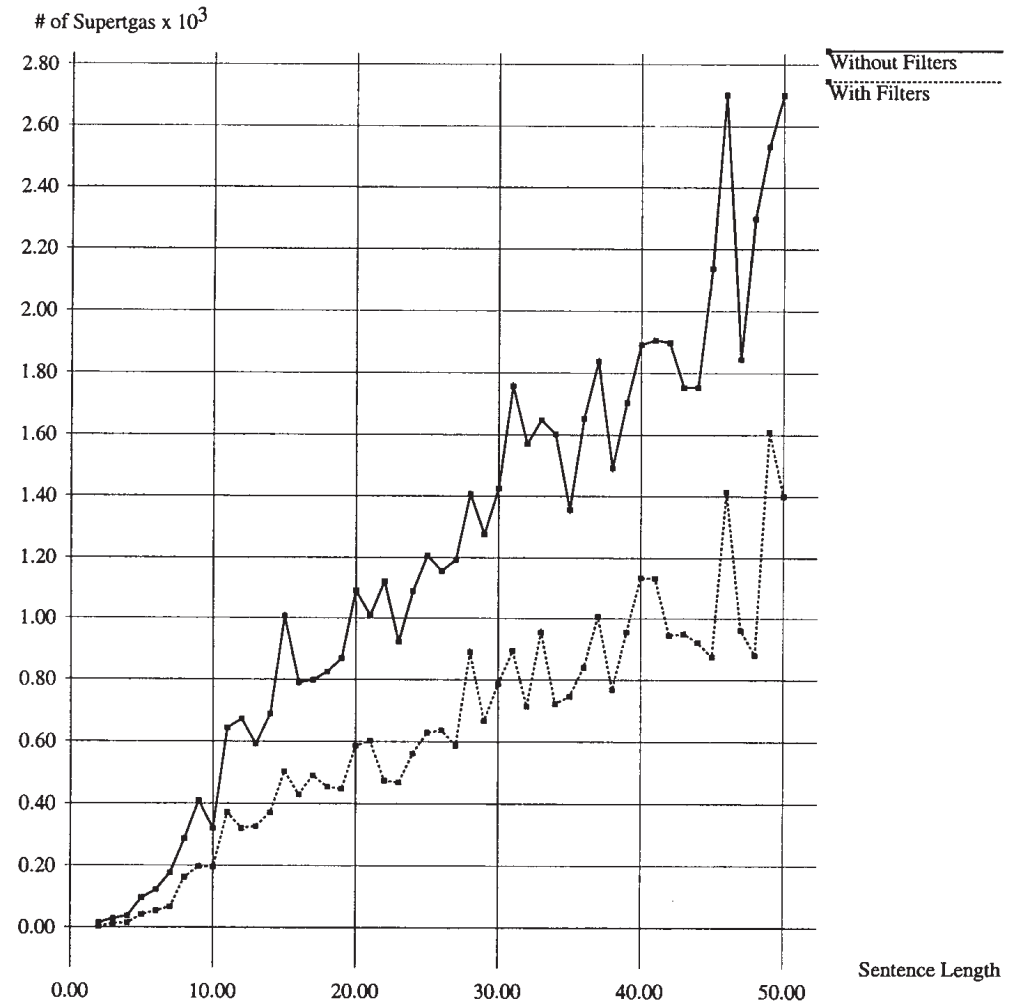
## How is a string supertagged?

Lecture 6 showed how an HMM and the Viterbi Algorithm could together be used to assign PoS tags to words. We noted that other tasks can be framed as tagging problems: Supertagging is one of them.

But supertags also represent *admissibility constraints* that can be checked locally. A potential supertag can be excluded if:

- its span is larger than the input string;
- its span to the left/right of the lexical anchor is larger than the string to the left/right of the anchor;
- if not all the terminals on its frontier do not appear in the string.

Effectiveness of admissibility constraints  
(figure from Bangalore and Joshi 1999):



## The Unigram Model

The unigram model is given by:

$$\text{Supertag}(w_i) = t_k \in \arg \max_{t_k} Pr(t_k|w_i)$$

$$Pr(t_k|w_i) = \frac{\text{frequency}(t_k, w_i)}{\text{frequency}(w_i)}$$

The most frequent supertag of a word is selected. For words that do not appear in the training corpus, we use the most frequent supertag associated with the word's PoS.

## The n-gram Model

The n-gram model takes into account the contextual dependencies between supertags within a window of n words. The most probable supertag sequence for an n-word sentence is given by:

$$\hat{T} = \arg \max_T Pr(T_1, T_2, \dots, T_N) * Pr(W_1, W_2, \dots, W_N | T_1, T_2, \dots, T_N)$$

where  $T_i$  is the supertag for word  $W_i$ .

The **word emission probabilities** are approximated as:

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

We assume that the probability of a word depends only on its supertag.

## The n-gram Model

The **contextual probabilities** are approximated using a trigram assumption:

$$Pr(T_1, T_2, \dots, T_N) \approx \prod_{i=1}^N Pr(T_i | T_{i-2}, T_{i-1})$$

We also need:

- Kneser-Ney smoothing
- treatment of unknown words
- treatment of rare supertags

## Comparison: Supertagging vs. PoS Tagging

The current LTAG system has, on average, 15–20 supertags per word.

For comparison with POS tags, the 1M word Brown corpus contains 39,440 different word **types**:

- 35,340 of these have only 1 POS tag anywhere in corpus (89.6%);
- The remaining 4,100 (10.4%) have 2-7 POS tags
- Because high-frequency words often have  $>1$  POS tag, over 40% of the word **tokens** have  $>1$  POS tag.

## Results

Performance of the supertagger on the WSJ corpus (Bangalore and Joshi, 1999):

Data Set	Size of Training Set (Words)	Training	Size of Set Test (Words)	Correct
XTAG Parses	8,000	Unigram	3,000	73.4%
		Trigram	3,000	86.0%
Converted Penn Treebank Parses	200,000	Unigram	47,000	75.3%
		Trigram	47,000	90.9%
Parses	1,000,000	Unigram	47,000	77.2%
		Trigram	47,000	92.2%

## Supertag use in Two-stage Parsing

An LTAG parser works in two stages:

1. Select all the supertags associated with each word of the sentence to be parsed.
2. Search the lattice of selected supertags, attempt to combine them using substitution and adjunction to get a derivation that spans the input.

The supertagger can be used to prune the search space at step 1. It selects the appropriate supertag for each word, so step 2 only needs to combine these supertags.

Bangalore and Joshi (1999) report a speedup of factor 30.

## Results

Performance of the LTAG parser on the WSJ corpus (Bangalore and Joshi, 1999), depending on the number of supertags considered:

Data Set	Size of Training Set (Words)	Training	Size of Set Test (Words)	Correct
Converted Penn Treebank Parses	200,000	Unigram (Best Supertag)	47,000	90.9%
		Trigram (3-Best Supertags)	47,000	95.5%
	1,000,000	Unigram (Best Supertag)	47,000	92.2%
		Trigram (3-Best Supertags)	47,000	97.1%

## Limitations of the Two-stage Approach

Parsers fails completely if a word is incorrectly tagged by the supertagger. We could select the n-best supertags instead.

Alternative: interactive approach:

- First try to parse using the most restricted set of supertags assigned by supertagger.
- If parser fails to find an analysis, relax threshold  $\beta$  to associate more supertags with each word, and try again.

The supertagger returns the supertags whose probability is within some factor  $\beta$  of the most probable supertag for a given word.

## References

- Bangalore, Srinivas, and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics* 25: 237–265.
- Sarkar, Anoop. 2010. Combining supertagging and lexicalized tree-adjoining grammar parsing. In Srinivas Bangalore and Aravind Joshi, eds., *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*, 133–158. MIT Press.