Ling 5801: Lecture Notes 17
Semantic Parsing

CFG parsers and Markov chains are generally weak predictors. Let's combine and improve them!

Contents

17.1 Problem: parse trees depend on meaning

Consider the prepositional phrases with ... in the following sentences:

- She ate [$_{\text{N}}$ [$_{\text{N}}$ pasta] with pesto]. – attaches at noun phrase
- She [$_{\text{V-aN}}$ [$_{\text{V-aN}}$ ate pasta] with a fork]. – attaches at verb phrase

Preferred attachment seems to depend on the difference between words pesto and a fork.

However, [$_{\text{N}}$ [$_{\text{N}}$ pesto] and [$_{\text{N}}$ a fork]] coordinate just fine, so they are not different categories.

17.2 Expensive Solution: word vectors

We want to model preferred attachment, so we will use 'prod_pair' and 'max_argmax' operators, but marginalize over words.

Use a linear algebraic marginal (over words) inside a Viterbi parse (over trees and categories)!

Assume $a = \langle p_a, \tau_a, \mathbf{v}_a \rangle \in \mathbb{R} \times \text{Tree} \times \mathbb{R}^D$ and $b = \langle p_b, \tau_b, \mathbf{v}_b \rangle \in \mathbb{R} \times \text{Tree} \times \mathbb{R}^D$:

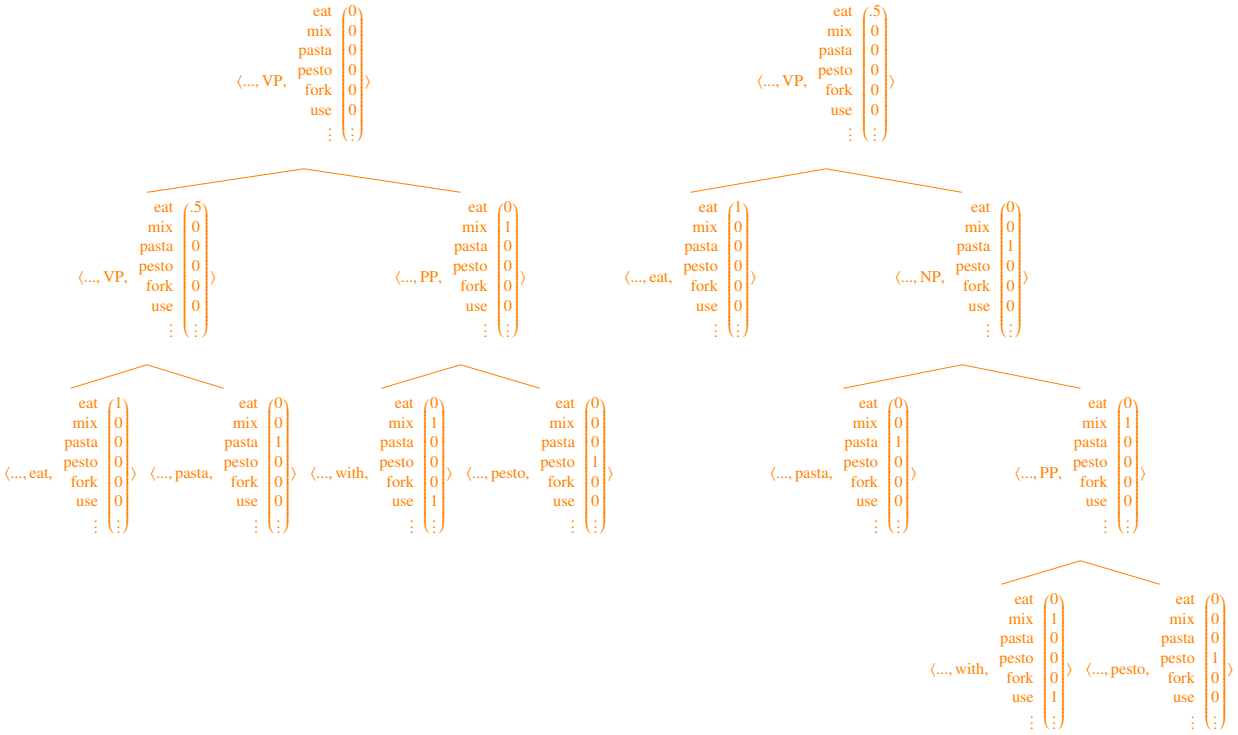
$$\text{prod\_pair}(a, b) = \langle p_a \cdot p_b, \ \langle \tau_a, \tau_b \rangle, \ \mathbf{G}\ (\mathbf{v}_a \otimes \mathbf{v}_b) \rangle$$

$$\text{max\_argmax}(a, b) = \begin{cases} a & \text{if} \quad p_a\, \mathbf{v}_a{}^\top \mathbf{1} \geq p_b\, \mathbf{v}_b{}^\top \mathbf{1} \\ b & \text{if} \quad p_a\, \mathbf{v}_a{}^\top \mathbf{1} < p_b\, \mathbf{v}_b{}^\top \mathbf{1} \end{cases}$$

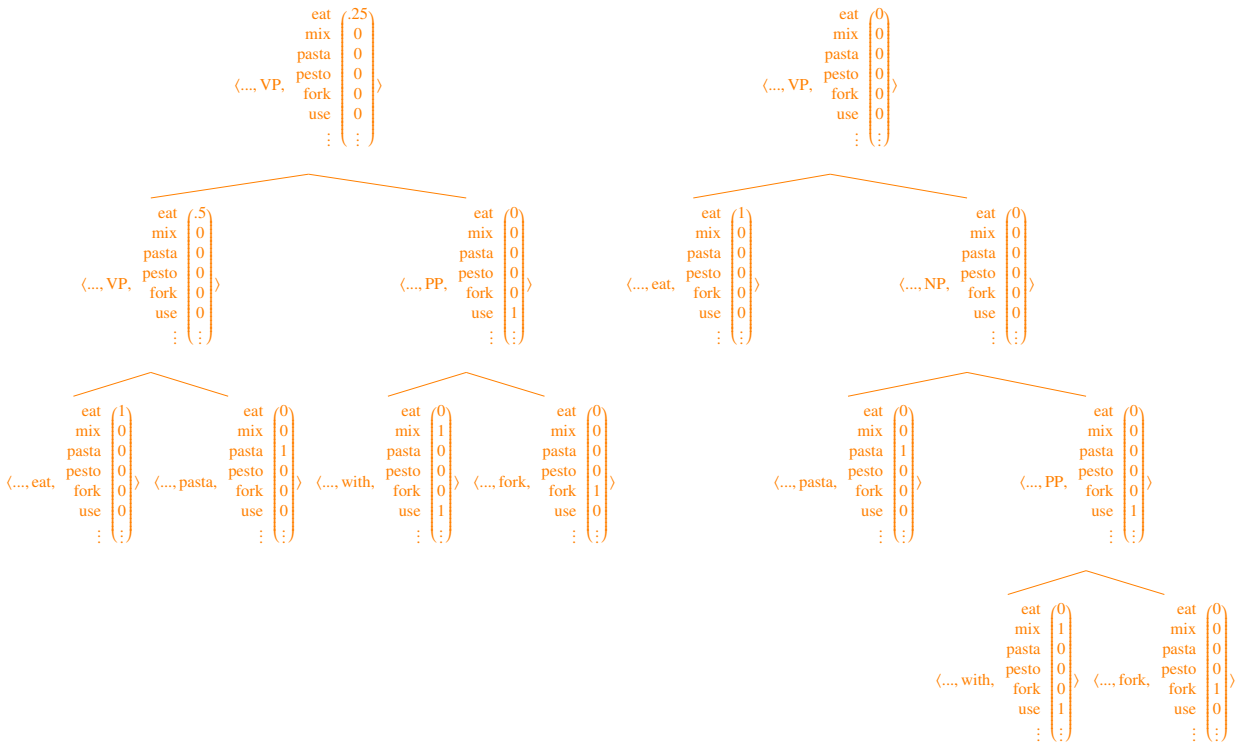
Use matrix **G** to combine vectors of word senses:

$$\mathbf{G} = \begin{array}{c} \\ \text{eat} \\ \text{mix} \\ \text{pasta} \\ \text{pesto} \\ \text{fork} \\ \text{use} \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccccccc} \cdots & \text{eat pasta} & \cdots & \text{eat use} & \cdots & \text{mix pesto} & \cdots & \text{pasta mix} & \cdots & \text{use fork} & \cdots \end{array} \\ \begin{pmatrix} \cdots & .5 & \cdots & .5 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots \\ \cdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots \end{pmatrix} \end{array}$$

Dispreferred and preferred trees for *. . . ate pasta with pesto*:



Preferred and dispreferred trees for *. . . ate pasta with a fork*:



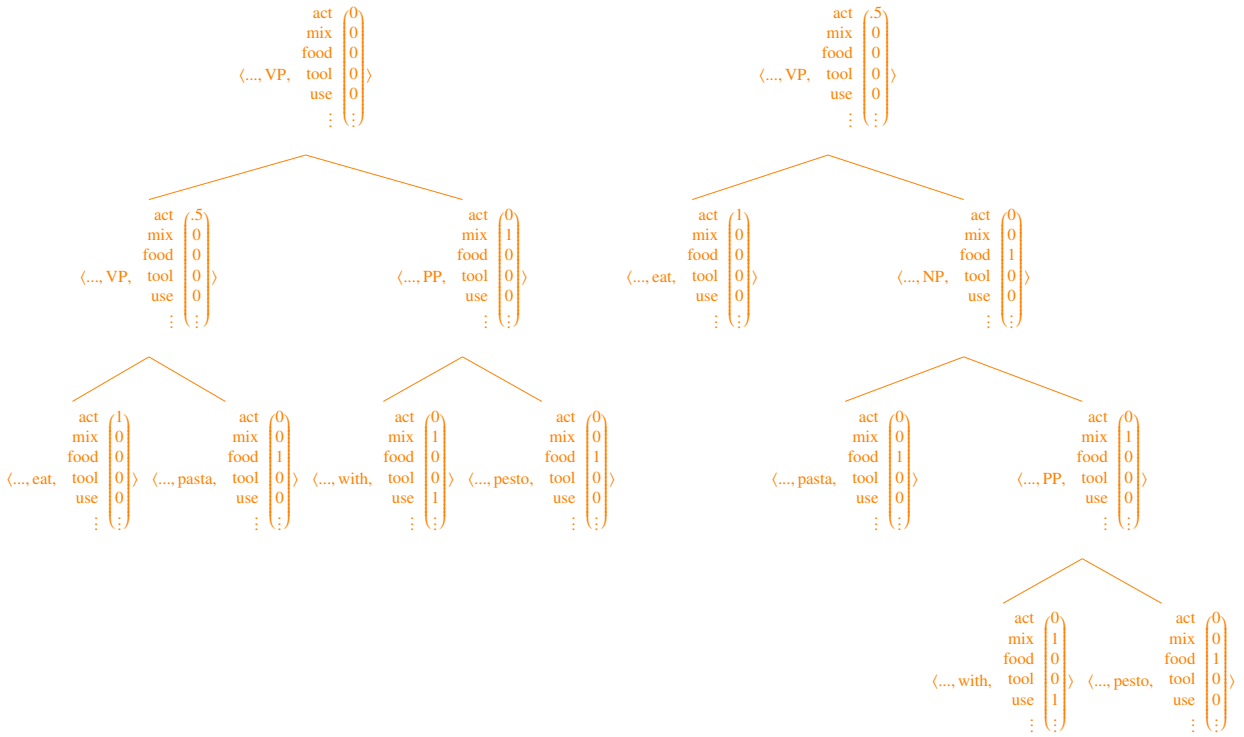
Problem: there are lots of words! Training **G** will require lots of data!

17.3 Cheaper Solution: semantic class vectors

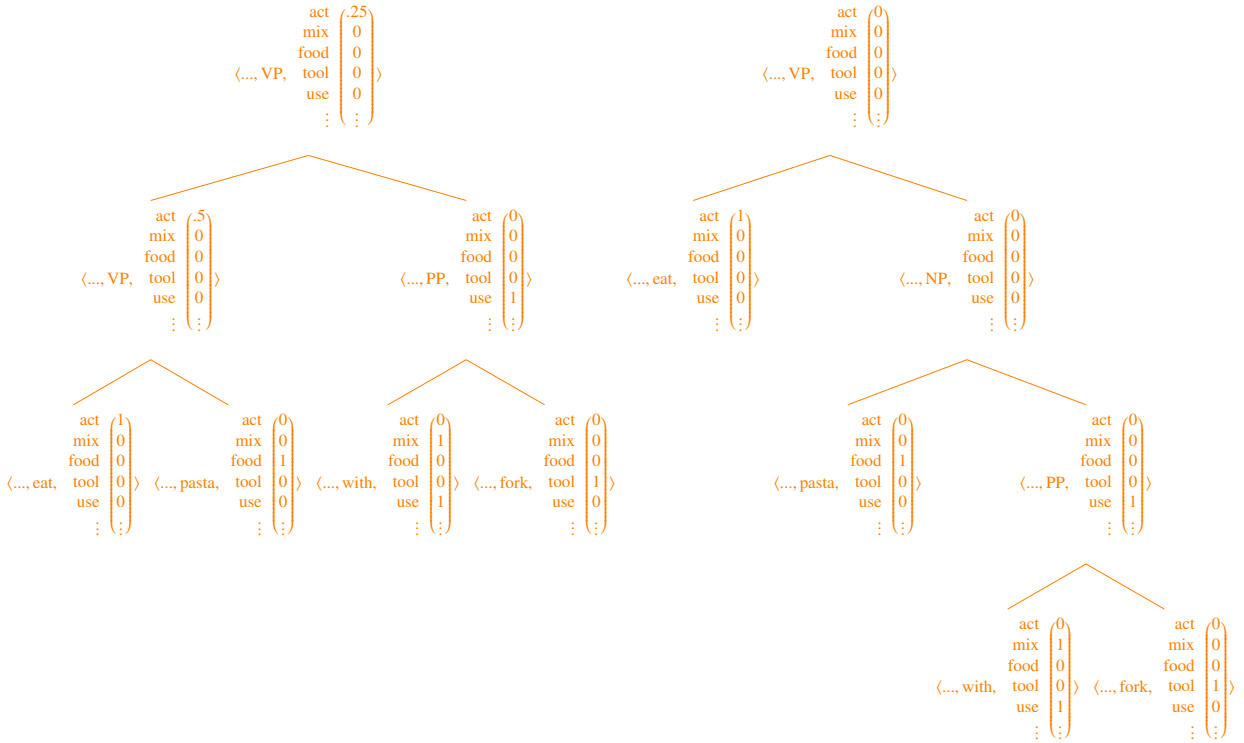
Shrink **G** to combine vectors of semantic classes:

$$\mathbf{G} = \begin{array}{c} \\ \text{act} \\ \text{mix} \\ \text{food} \\ \text{tool} \\ \text{use} \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccccc} \text{act food} & \cdots & \text{act use} & \cdots & \text{mix food} & \cdots & \text{food mix} & \cdots & \text{use tool} \end{array} \\ \begin{pmatrix} .5 & \cdots & .5 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \end{pmatrix} \end{array}$$

Dispreferred and preferred trees for *. . . ate pasta with pesto*:



Preferred and dispreferred trees for *. . . ate pasta with a fork*:



There are fewer of these, but someone has to classify everything

(and that's difficult because polysemous words may map to several classes).

17.4 Better Solution: dimensionality reduction

Shrink **G** matrix by capturing variation in fewer dimensions...