6.864 (Fall 2007)
Global Linear Models

Overview

- A brief review of history-based methods
- A new framework: Global linear models
- Parsing problems in this framework:
  Reranking problems
- Parameter estimation method 1:
  A variant of the perceptron algorithm

Techniques

- So far:
  - Smoothed estimation
  - Probabilistic context-free grammars
  - Log-linear models
  - Hidden markov models
  - The EM Algorithm
  - History-based models
- Today:
  - Global linear models

Supervised Learning in Natural Language

- General task: induce a function F from members of a set $\mathcal{X}$ to members of a set $\mathcal{Y}$. e.g.,

Problem	$x \in \mathcal{X}$	$y \in \mathcal{Y}$
Parsing	sentence	parse tree
Machine translation	French sentence	English sentence
POS tagging	sentence	sequence of tags

- Supervised learning:
  we have a *training set* (x_i, y_i) for $i = 1 \ldots n$

The Models so far

- Most of the models we've seen so far are **history-based models**:
  - We break structures down into a derivation, or sequence of decisions
  - Each decision has an associated conditional probability
  - Probability of a structure is a product of decision probabilities
  - Parameter values are estimated using variants of maximum-likelihood estimation
  - Function $F : \mathcal{X} \rightarrow \mathcal{Y}$ is defined as

$$F(x) = \text{argmax}_y P(y, x \mid \Theta) \quad \text{or} \quad F(x) = \text{argmax}_y P(y \mid x, \Theta)$$

Example 1: PCFGs

- We break structures down into a derivation, or sequence of decisions
  We have a top-down derivation, where each decision is to expand some non-terminal α with a rule $\alpha \rightarrow \beta$
- Each decision has an associated conditional probability
  $\alpha \rightarrow \beta$ has probability $P(\alpha \rightarrow \beta \mid \alpha)$
- Probability of a structure is a product of decision probabilities

$$P(T, S) = \prod_{i=1}^{n} P(\alpha_i \rightarrow \beta_i \mid \alpha_i)$$

  where $\alpha_i \rightarrow \beta_i$ for $i = 1 \ldots n$ are the n rules in the tree
- Parameter values are estimated using variants of maximum-likelihood estimation

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

- Function $F : \mathcal{X} \rightarrow \mathcal{Y}$ is defined as

$$F(x) = \text{argmax}_y P(y, x \mid \Theta)$$

  Can be computed using dynamic programming

Example 2: Log-linear Taggers

- We break structures down into a derivation, or sequence of decisions
  For a sentence of length n we have n tagging decisions, in left-to-right order
- Each decision has an associated conditional probability

$$P(t_i \mid t_{i-1}, t_{i-2}, w_1 \ldots w_n)$$

  where t_i is the i'th tagging decision, w_i is the i'th word
- Probability of a structure is a product of decision probabilities

$$P(t_1 \ldots t_n \mid w_1 \ldots w_n) = \prod_{i=1}^{n} P(t_i \mid t_{i-1}, t_{i-2}, w_1 \ldots w_n)$$

- Parameter values are estimated using variants of maximum-likelihood estimation
  $P(t_i \mid t_{i-1}, t_{i-2}, w_1 \ldots w_n)$ is estimated using a log-linear model

- Function $F : \mathcal{X} \rightarrow \mathcal{Y}$ is defined as

$$F(x) = \text{argmax}_y P(y \mid x, \Theta)$$

Can be computed using dynamic programming

A New Set of Techniques: Global Linear Models

Overview of today's lecture:

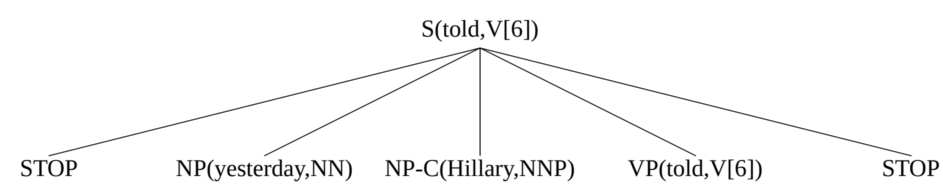
- Global linear models as a framework
- Parsing problems in this framework:
  - Reranking problems
- A variant of the perceptron algorithm

Global Linear Models as a Framework

- We'll move away from history-based models
  No idea of a "derivation", or attaching probabilities to "decisions"
- Instead, we'll have feature vectors over entire structures
  "Global features"
- First piece of motivation:
  Freedom in defining features

An Example: Parsing

- In lecture 4, we described lexicalized models for parsing
- Showed how a rule can be generated in a number of steps



Model 2

- Step 1: generate category of head child

S(told,V[6])

$\Downarrow$

S(told,V[6])

VP(told,V[6])

P_h(VP | S, told, V[6])

Model 2

- Step 2: choose left **subcategorization frame**

S(told,V[6])

VP(told,V[6])

$\Downarrow$

S(told,V[6])

VP(told,V[6])
{NP-C}

P_h(VP | S, told, V[6]) $\times$ P_{lc}({NP-C} | S, VP, told, V[6])

- Step 3: generate left modifiers in a Markov chain

S(told,V[6])

?? VP(told,V[6])
{NP-C}

$\Downarrow$

S(told,V[6])

NP-C(Hillary,NNP) VP(told,V[6])
{}

P_h(VP | S, told, V[6]) $\times$ P_{lc}({NP-C} | S, VP, told, V[6])$\times$
P_d(NP-C(Hillary,NNP) | S,VP,told,V[6],LEFT,{NP-C})

S(told,V[6])

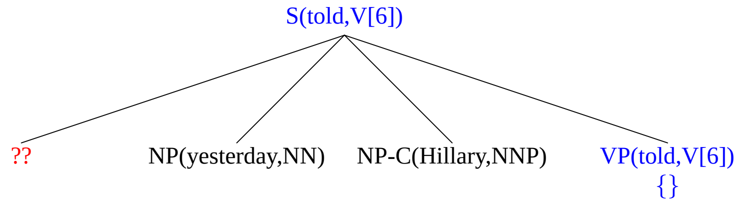
?? NP-C(Hillary,NNP) VP(told,V[6])
{}

$\Downarrow$

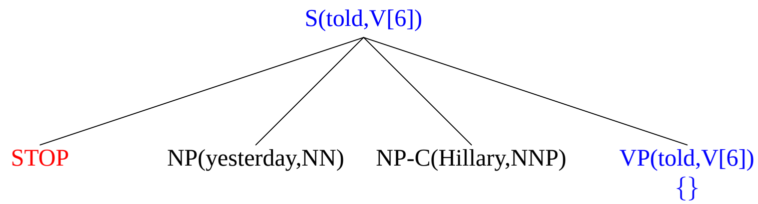
S(told,V[6])

NP(yesterday,NN) NP-C(Hillary,NNP) VP(told,V[6])
{}

P_h(VP | S, told, V[6]) $\times$ P_{lc}({NP-C} | S, VP, told, V[6])
P_d(NP-C(Hillary,NNP) | S,VP,told,V[6],LEFT,{NP-C})$\times$
P_d(NP(yesterday,NN) | S,VP,told,V[6],LEFT,{})

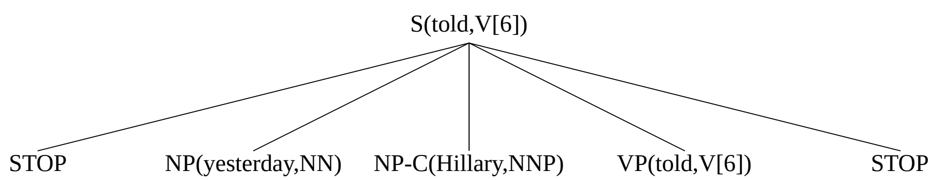


$\Downarrow$



P_h(VP | S, told, V[6]) $\times$ P_{lc}({NP-C} | S, VP, told, V[6])
P_d(NP-C(Hillary,NNP) | S,VP,told,V[6],LEFT,{NP-C})$\times$
P_d(NP(yesterday,NN) | S,VP,told,V[6],LEFT,{})$\times$
P_d(STOP | S,VP,told,V[6],LEFT,{})

The Probabilities for One Rule



P_h(VP | S, told, V[6])$\times$
P_{lc}({NP-C} | S, VP, told, V[6])$\times$
P_{rc}({} | S, VP, told, V[6])$\times$
P_d(NP-C(Hillary,NNP) | S,VP,told,V[6],LEFT,$\Delta = 1$,{NP-C})$\times$
P_d(NP(yesterday,NN) | S,VP,told,V[6],LEFT,$\Delta = 0$,{})$\times$
P_d(STOP | S,VP,told,V[6],LEFT,$\Delta = 0$,{})$\times$
P_d(STOP | S,VP,told,V[6],RIGHT,$\Delta = 1$,{})

Three parameter types:
Head parameters, Subcategorization parameters, Dependency parameters

Smoothed Estimation

$$P(\text{NP(\_\_,NN) VP} \mid \text{S(questioned,Vt)}) =$$

$$\lambda_1 \times \frac{Count(\text{S(questioned,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(questioned,Vt)})}$$

$$+\lambda_2 \times \frac{Count(\text{S(\_\_,Vt)}\rightarrow\text{NP(\_\_,NN) VP})}{Count(\text{S(\_\_,Vt)})}$$

- Where $0 \leq \lambda_1, \lambda_2 \leq 1$, and $\lambda_1 + \lambda_2 = 1$

Smoothed Estimation

$$P(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt}) =$$

$$\lambda_1 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,questioned,Vt})}{Count(\text{S,VP,NP,NN,questioned,Vt})}$$

$$+\lambda_2 \times \frac{Count(\text{lawyer} \mid \text{S,VP,NP,NN,Vt})}{Count(\text{S,VP,NP,NN,Vt})}$$

$$+\lambda_3 \times \frac{Count(\text{lawyer} \mid \text{NN})}{Count(\text{NN})}$$

- Where $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$

An Example: Parsing

- In lecture 4, we described lexicalized models for parsing
- Showed how a rule can be generated in a number of steps
- The end result:
  - We have head, dependency, and subcategorization parameters
  - Smoothed estimation means "probability" of a tree depends on counts of many different types of events
- **What if we want to add new features?**
  Can be very awkward to incorporate some features in history-based models

A Need for Flexible Features

Example 1 Parallelism in coordination [Johnson et. al 1999]

Constituents with similar structure tend to be coordinated
⇒ how do we allow the parser to learn this preference?

Bars in New York and pubs in London
vs. Bars in New York and pubs

Example 2 Semantic features

We might have an ontology giving properties of various nouns/verbs
⇒ how do we allow the parser to use this information?

pour the **cappucino**
vs. pour the **book**

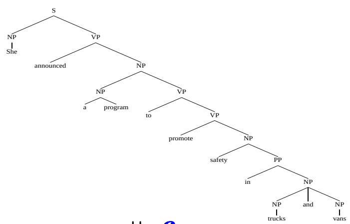
Ontology states that **cappucino** has the `+liquid` feature, **book** does not.

Three Components of Global Linear Models

- $\mathbf{f}$ is a function that maps a structure (x, y) to a **feature vector** $\mathbf{f}(x, y) \in \mathbb{R}^d$
- $\mathbf{GEN}$ is a function that maps an input x to a set of **candidates** $\mathbf{GEN}(x)$
- $\mathbf{W}$ is a parameter vector (also a member of $\mathbb{R}^d$)
- Training data is used to set the value of $\mathbf{W}$

Component 1: f

- $\mathbf{f}$ maps a candidate to a **feature vector** $\in \mathbb{R}^d$
- $\mathbf{f}$ defines the **representation** of a candidate

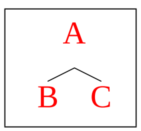


$\Downarrow \mathbf{f}$

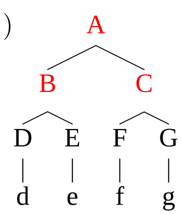
$\langle 1, 0, 2, 0, 0, 15, 5 \rangle$

Features

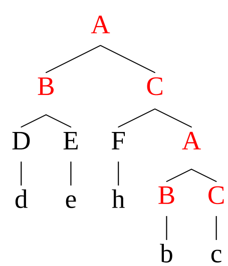
- A "feature" is a function on a structure, e.g.,

$h(x, y) =$ Number of times  is seen in (x, y)

(x_1, y_1)



(x_2, y_2)



$h(x_1, y_1) = 1$

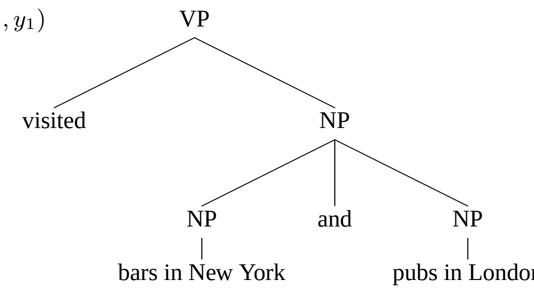
$h(x_2, y_2) = 2$

Another Example

- A "feature" is a function on a structure, e.g.,

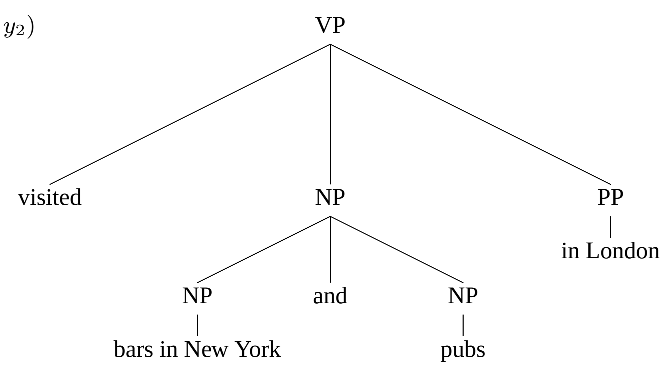
$$h(x, y) = \begin{cases} 1 & \text{if } (x, y) \text{ has an instance of non-parallel coordination} \\ 0 & \text{otherwise} \end{cases}$$

(x_1, y_1)



$h(x_1, y_1) = 0$

(x_2, y_2)

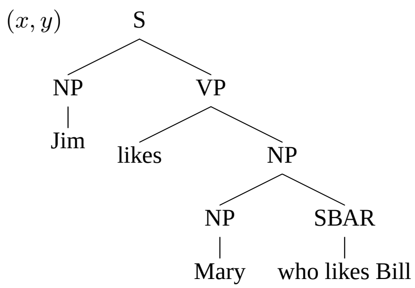


$h(x_2, y_2) = 1$

A Third Example

- A "feature" is a function on a structure, e.g.,

$$h_1(x, y) = \text{number of times } \textit{Mary} \text{ is subject of } \textit{likes}$$
$$h_2(x, y) = \text{number of times } \textit{Mary} \text{ is object of } \textit{likes}$$

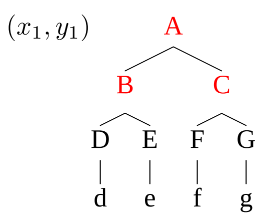


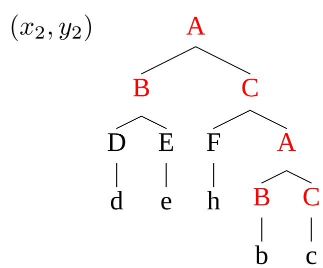
$h_1(x, y) = 1$ $\qquad$ $h_2(x, y) = 1$

A Final Example

- A "feature" is a function on a structure, e.g.,

$$h(x, y) = \text{log probability of } (x, y) \text{ under Model 2}$$

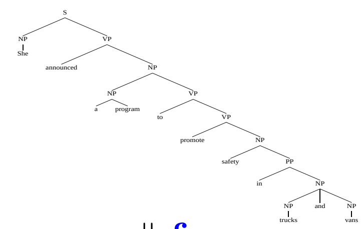




$h(x_1, y_1) = -1.56$ $\qquad$ $h(x_2, y_2) = -1.98$

Component 1: f

- **f** maps a candidate to a **feature vector** $\in \mathbb{R}^d$
- **f** defines the **representation** of a candidate



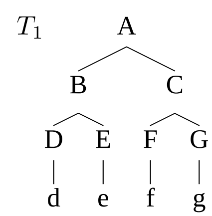
$\Downarrow$ **f**

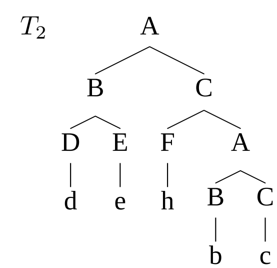
$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle$$

Feature Vectors

- A set of functions $h_1 \ldots h_d$ define a **feature vector**

$$\mathbf{f}(x) = \langle h_1(x), h_2(x) \ldots h_d(x) \rangle$$





$\mathbf{f}(T_1) = \langle 1, 0, 0, 3 \rangle$ $\qquad$ $\mathbf{f}(T_2) = \langle 2, 0, 1, 1 \rangle$

Feature Vectors

- Our goal is to come up with learning methods which allow us to define any features over parse trees

- Avoids the intermediate steps of a history-based model: defining a **derivation** which takes the features into account, and then attaching probabilities to **decisions**
Our claim is that this can be an unwieldy, indirect way of getting desired features into a model

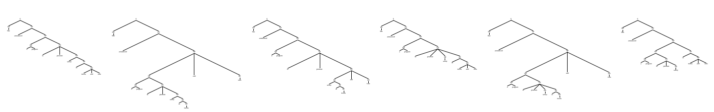
- Problem of **representation** now boils down to the choice of the function $\mathbf{f}(x, y)$

Component 2: GEN

- **GEN** enumerates a set of **candidates** for a sentence

She announced a program to promote safety in trucks and vans

$$\Downarrow \textbf{GEN}$$



Component 2: GEN

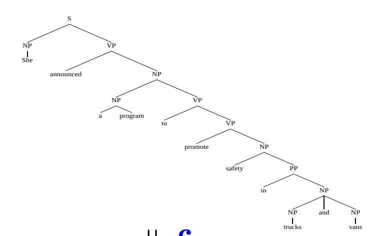
- **GEN** enumerates a set of **candidates** for an input x

- Some examples of how $\mathbf{GEN}(x)$ can be defined:
  - Parsing: $\mathbf{GEN}(x)$ is the set of parses for x under a grammar
  - Any task: $\mathbf{GEN}(x)$ is the top N most probable parses under a history-based model
  - Tagging: $\mathbf{GEN}(x)$ is the set of all possible tag sequences with the same length as x
  - Translation: $\mathbf{GEN}(x)$ is the set of all possible English translations for the French sentence x

Component 3: W

- $\mathbf{W}$ is a **parameter vector** $\in \mathbb{R}^d$
- $\mathbf{f}$ and $\mathbf{W}$ together map a candidate to a real-valued score

---



$$\Downarrow \mathbf{f}$$

$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle$$

$$\Downarrow \mathbf{f} \cdot \mathbf{W}$$

$$\langle 1, 0, 2, 0, 0, 15, 5 \rangle \cdot \langle 1.9, -0.3, 0.2, 1.3, 0, 1.0, -2.3 \rangle = 5.8$$

Putting it all Together

- $\mathcal{X}$ is set of sentences, $\mathcal{Y}$ is set of possible outputs (e.g. trees)
- Need to learn a function $F : \mathcal{X} \rightarrow \mathcal{Y}$
- **GEN**, **f**, **W** define

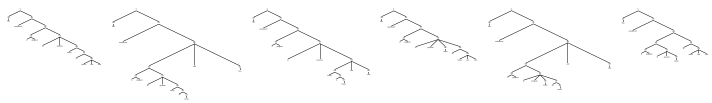
$$F(x) = \arg\max_{y \in \mathbf{GEN}(x)} \mathbf{f}(x, y) \cdot \mathbf{W}$$

Choose the highest scoring candidate as the most plausible structure

- Given examples (x_i, y_i), how to set $\mathbf{W}$?

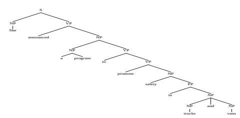
She announced a program to promote safety in trucks and vans

$\Downarrow$ **GEN**



$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$	$\Downarrow \mathbf{f}$
$\langle 1, 1, 3, 5 \rangle$	$\langle 2, 0, 0, 5 \rangle$	$\langle 1, 0, 1, 5 \rangle$	$\langle 0, 0, 3, 0 \rangle$	$\langle 0, 1, 0, 5 \rangle$	$\langle 0, 0, 1, 5 \rangle$
$\Downarrow \mathbf{f} \cdot \mathbf{W}$	$\Downarrow \mathbf{f} \cdot \mathbf{W}$	$\Downarrow \mathbf{f} \cdot \mathbf{W}$	$\Downarrow \mathbf{f} \cdot \mathbf{W}$	$\Downarrow \mathbf{f} \cdot \mathbf{W}$	$\Downarrow \mathbf{f} \cdot \mathbf{W}$
13.6	12.2	12.1	3.3	9.4	11.1

$\Downarrow \arg\max$



Overview

- A brief review of history-based methods
- A new framework: Global linear models
- Parsing problems in this framework:
  Reranking problems
- Parameter estimation method 1:
  A variant of the perceptron algorithm

Reranking Approaches to Parsing

- Use a **baseline** parser to produce top N parses for each sentence in training and test data
  $\mathbf{GEN}(x)$ is the top N parses for x under the baseline model
- One method: use Model 2 to generate a number of parses
  (in our experiments, around 25 parses on average for 40,000 training sentences, giving $\approx$ 1 million training parses)
- **Supervision:** for each x_i take y_i to be the parse that is "closest" to the treebank parse in $\mathbf{GEN}(x_i)$

The Representation $\mathbf{f}$

- Each component of $\mathbf{f}$ could be essentially *any* feature over parse trees

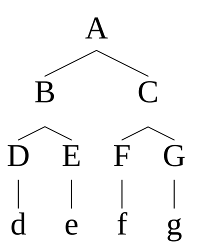
- For example:

$$f_1(x,y) = \text{log probability of } (x,y) \text{ under the baseline model}$$

$$f_2(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ includes the rule } \texttt{VP} \rightarrow \texttt{PP VBD NP} \\ 0 & \text{otherwise} \end{cases}$$

Practical Issues in Creating $\mathbf{f}$

- Step 1: map a tree to a number of "strings" representing features

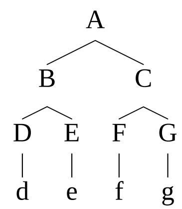


$\Rightarrow$

```
HASRULE:A->B;C
HASRULE:B->D;E
HASRULE:C->F;G
HASRULE:D->d
HASRULE:E->e
HASRULE:F->f
HASRULE:G->g
```


Practical Issues in Creating $\mathbf{f}$

- Step 2: hash the strings to integers



$\Rightarrow$

HASRULE:A->B;C	54
HASRULE:B->D;E	118
HASRULE:C->F;G	14
HASRULE:D->d	10078
HASRULE:E->e	9000
HASRULE:F->f	1078
HASRULE:G->g	101

- In our experiments, tree is then represented as log probability under the baseline model, plus a sparse feature array:

$$f_1(x,y) = \text{log probability of } (x,y) \text{ under the baseline model}$$

$$f_i(x,y) = 1 \text{ for } i = \{54, 118, 14, 10078, 9000, 1078, 101\}$$

$$f_i(x,y) = 0 \text{ for all other } i$$

The Score for Each Parse

- In our experiments, tree is then represented as log probability under the baseline model, plus a sparse feature array:

$$f_1(x,y) = \text{log probability of } (x,y) \text{ under the baseline model}$$

$$f_i(x,y) = 1 \text{ for } i = \{54, 118, 14, 10078, 9000, 1078, 101\}$$

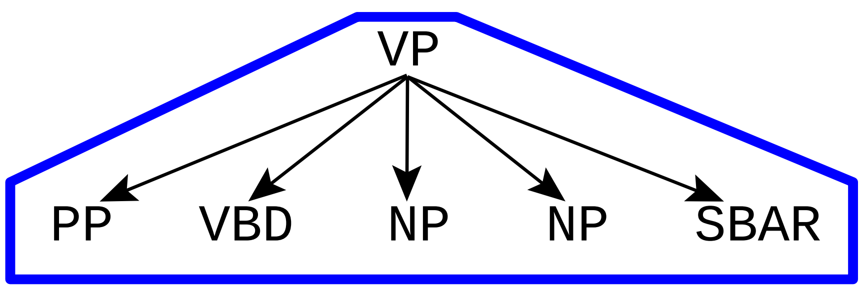
$$f_i(x,y) = 0 \text{ for all other } i$$

- Score for the tree (x,y):

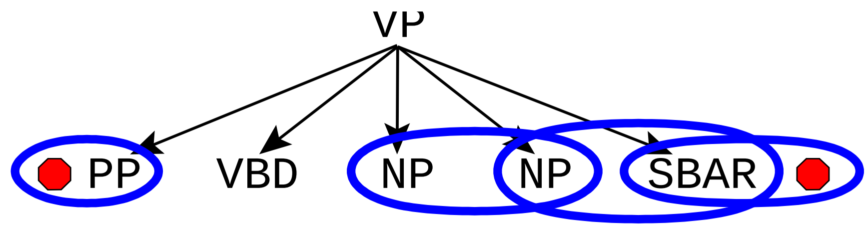
$$\begin{aligned} & \mathbf{f}(x,y) \cdot \mathbf{W} \\ = & \sum_i f_i(x,y)\mathbf{W}_i \\ = & \mathbf{W}_1 f_1(x,y) + \mathbf{W}_{54} + \mathbf{W}_{118} + \mathbf{W}_{14} + \mathbf{W}_{10078} + \mathbf{W}_{9000} + \mathbf{W}_{1078} + \mathbf{W}_{101} \end{aligned}$$

From [Collins and Koo, 2005]:
The following types of features were included in the model. We will use the rule `VP -> PP VBD NP NP SBAR` with head `VBD` as an example. Note that the output of our baseline parser produces syntactic trees with headword annotations.

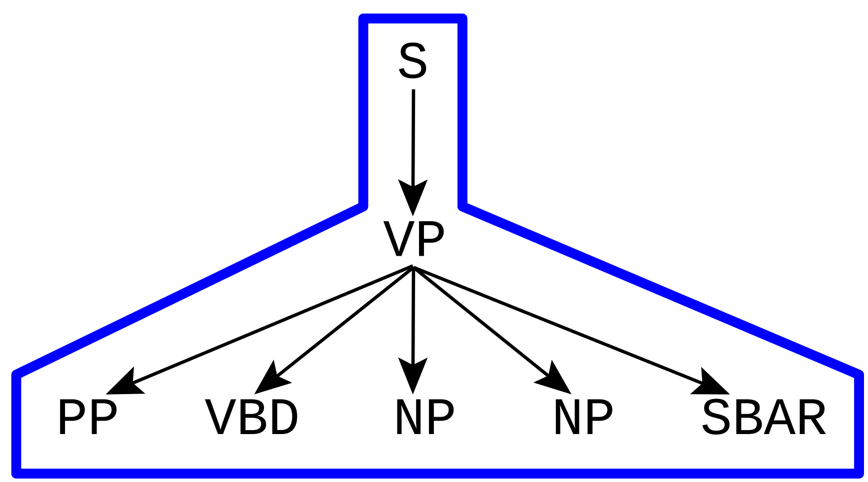
Rules These include all context-free rules in the tree, for example `VP -> PP VBD NP NP SBAR`.



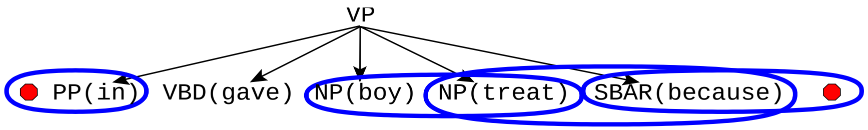
Bigrams These are adjacent pairs of non-terminals to the left and right of the head. As shown, the example rule would contribute the bigrams `(Right,VP,NP,NP)`, `(Right,VP,NP,SBAR)`, `(Right,VP,SBAR,STOP)`, and `(Left,VP,PP,STOP)` to the left of the head.



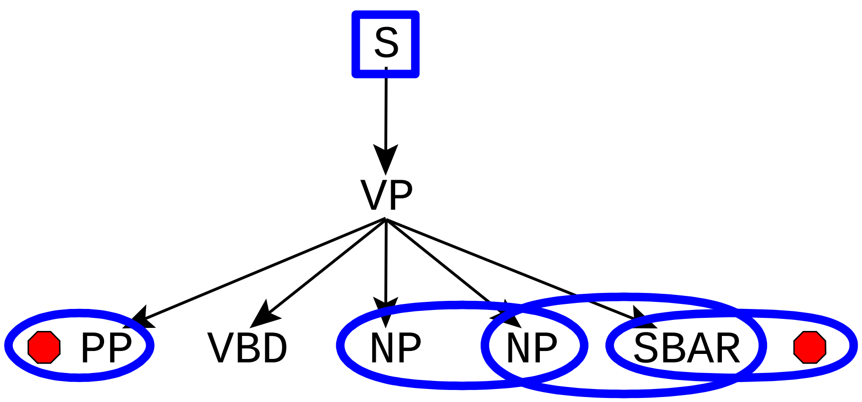
Grandparent Rules Same as **Rules**, but also including the non-terminal above the rule.



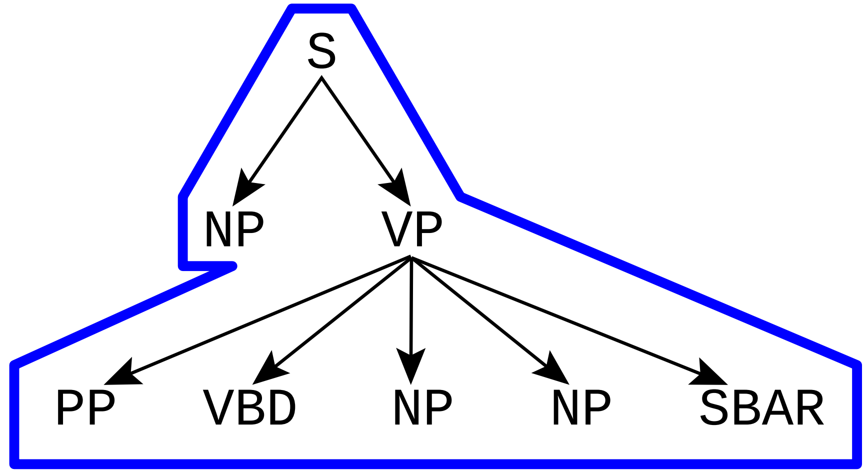
Lexical Bigrams Same as **Bigrams**, but with the lexical heads of the two non-terminals also included.



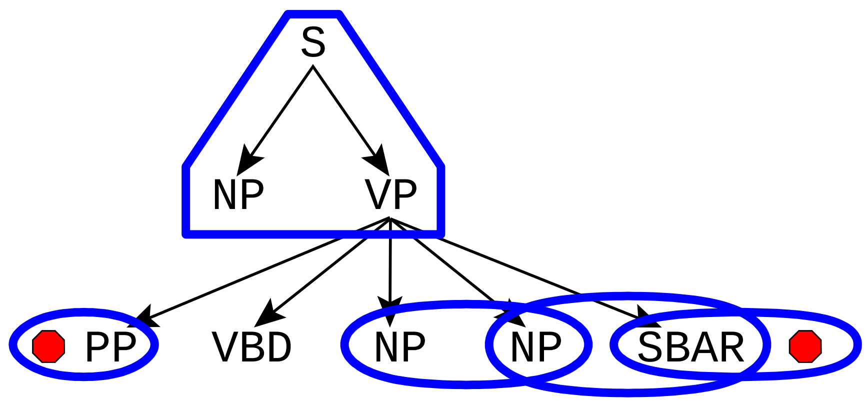
Grandparent Bigrams Same as **Bigrams**, but also including the non-terminal above the bigrams.



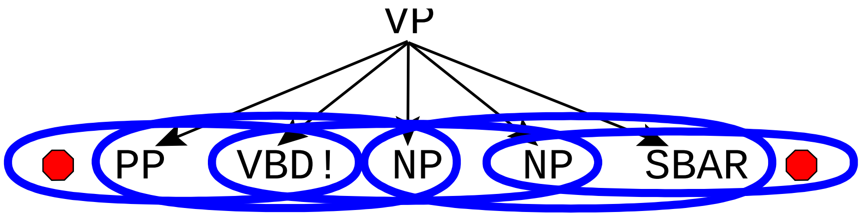
Two-level Rules Same as **Rules**, but also including the entire rule above the rule.



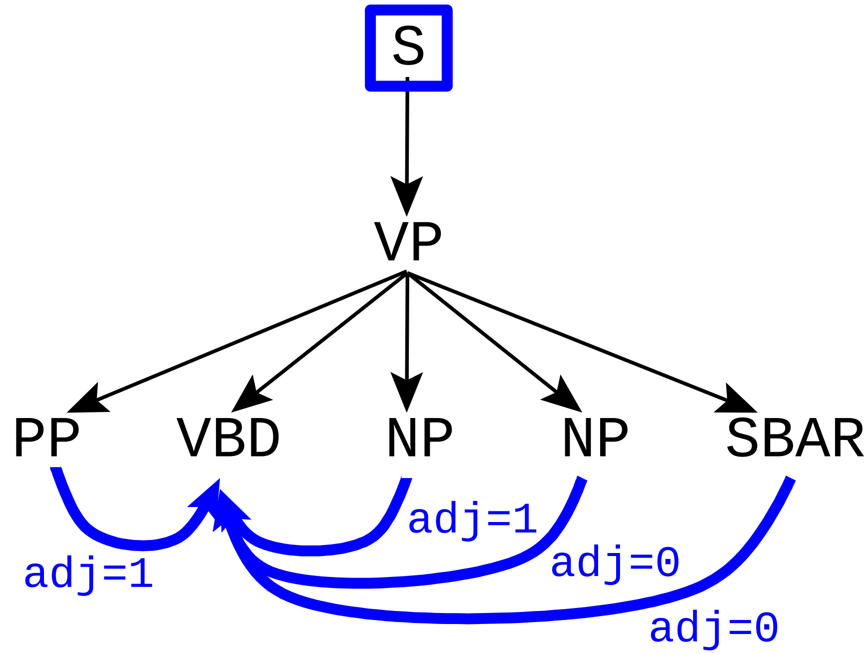
Two-level Bigrams Same as **Bigrams**, but also including the entire rule above the rule.



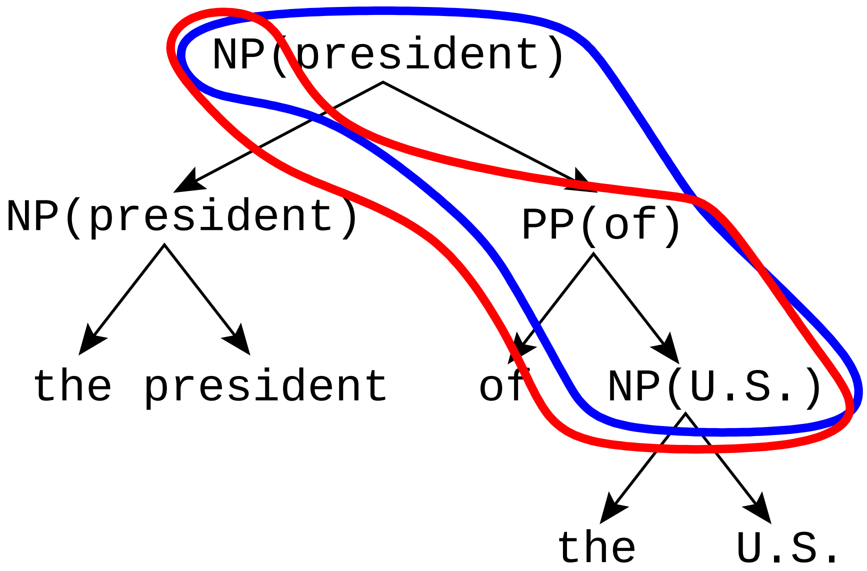
Trigrams All trigrams within the rule. The example rule would contribute the trigrams (VP,STOP,PP,VBD!), (VP,PP,VBD!,NP), (VP, VBD!,NP,NP), (VP,NP,NP,SBAR) and (VP,NP, SBAR,STOP) (! is used to mark the head of the rule)



Head-Modifiers All head-modifier pairs, with the grandparent non-terminal also included. An adj flag is also included, which is 1 if the modifier is adjacent to the head, 0 otherwise. As an example, say the non-terminal dominating the example rule is S. The example rule would contribute (Left,S,VP,VBD,PP,adj=1), (Right,S,VP,VBD,NP,adj=1), (Right,S,VP, VBD,NP,adj=0), and (Right,S,VP,VBD,SBAR, adj=0).



PPs Lexical trigrams involving the heads of arguments of prepositional phrases. The example shown at right would contribute the trigram (NP,NP,PP,NP,president,of,U.S.), in addition to the more general trigram relation (NP,NP,PP,NP,of,U.S.).



Distance Head-Modifiers Features involving the distance between head words. For example, assume $dist$ is the number of words between the head words of the `VBD` and `SBAR` in the `(VP,VBD,SBAR)` head-modifier relation in the above rule. This relation would then generate features `(VP,VBD,SBAR,`$= dist$`)`, and `(VP,VBD,SBAR,`$\leq x$`)` for all $dist \leq x \leq 9$ and `(VP,VBD,SBAR,`$\geq x$`)` for all $1 \leq x \leq dist$.

Further Lexicalization In order to generate more features, a second pass was made where all non-terminals were augmented with their lexical heads when these headwords were closed-class words. All features apart from **Head-Modifiers**, **PPs** and **Distance Head-Modifiers** were then generated with these augmented non-terminals.

Overview

- A brief review of history-based methods
- A new framework: Global linear models
- Parsing problems in this framework:
  Reranking problems
- Parameter estimation method 1:
  A variant of the perceptron algorithm

A Variant of the Perceptron Algorithm

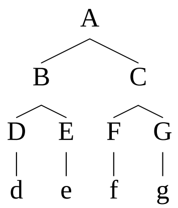
Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{f}(x, y) \cdot \mathbf{W}$

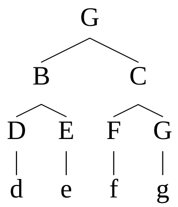
Algorithm: For $t = 1 \ldots T$, $i = 1 \ldots n$
$z_i = F(x_i)$
If $(z_i \neq y_i)$ $\quad \mathbf{W} = \mathbf{W} + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i)$

Output: Parameters $\mathbf{W}$



$\Rightarrow$

log probability	-1.56
`HASRULE:A->B;C`	54
`HASRULE:B->D;E`	118
`HASRULE:C->F;G`	14
`HASRULE:D->d`	10078
`HASRULE:E->e`	9000
`HASRULE:F->f`	1078
`HASRULE:G->g`	101



$\Rightarrow$

log probability	-1.13
`HASRULE:G->B;C`	89
`HASRULE:B->D;E`	118
`HASRULE:C->F;G`	14
`HASRULE:D->d`	10078
`HASRULE:E->e`	9000
`HASRULE:F->f`	1078
`HASRULE:G->g`	101

Say first tree is correct, but second tree has higher $\mathbf{W}\cdot\mathbf{f}(x,y)$ under current parameters: Then $\mathbf{W} = \mathbf{W} + \mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,z_i)$ implies

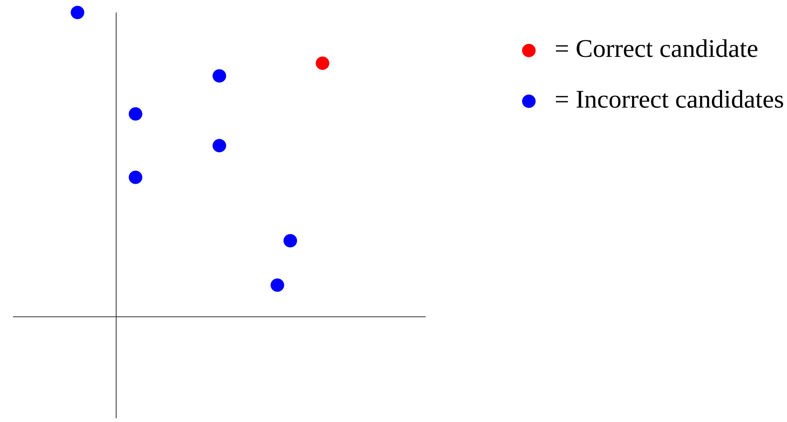
$$\begin{aligned}\mathbf{W}_1 &\mathrel{+}= -1.56-(-1.13) = -0.43\\ \mathbf{W}_{54} &\mathrel{+}= 1\\ \mathbf{W}_{89} &\mathrel{+}= -1\end{aligned}$$

Theory Underlying the Algorithm

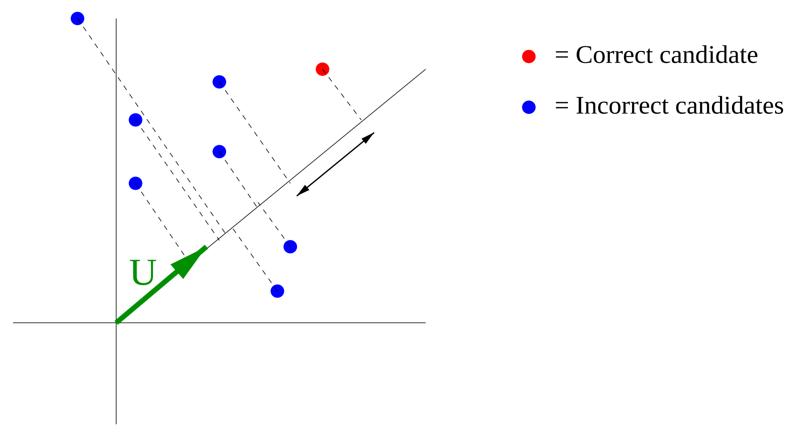
- **Definition:** $\overline{\mathbf{GEN}}(x_i) = \mathbf{GEN}(x_i) - \{y_i\}$
- **Definition:** The training set is **separable with margin** δ, if there is a vector $\mathbf{U}\in\mathbb{R}^d$ with $||\mathbf{U}|| = 1$ such that

$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i) \quad \mathbf{U}\cdot\mathbf{f}(x_i,y_i) - \mathbf{U}\cdot\mathbf{f}(x_i,z) \ge \delta$$

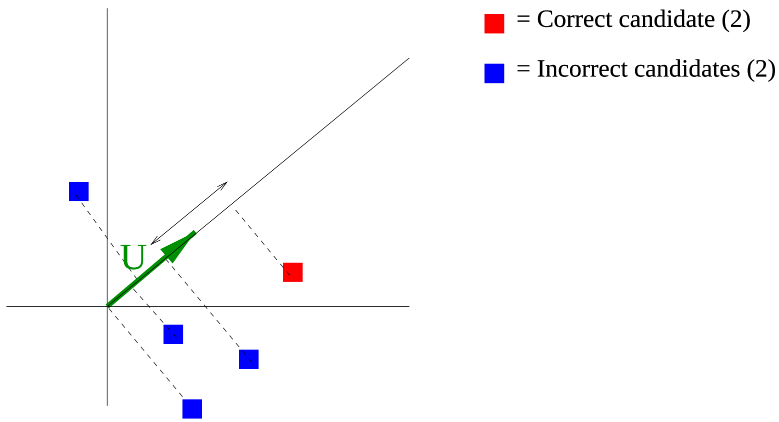
GEOMETRIC INTUITION BEHIND SEPARATION



GEOMETRIC INTUITION BEHIND SEPARATION



ALL EXAMPLES ARE SEPARATED



THEORY UNDERLYING THE ALGORITHM

Theorem: For any training sequence (x_i, y_i) which is separable with margin δ, then for the perceptron algorithm

$$\text{Number of mistakes} \leq \frac{R^2}{\delta^2}$$

where R is a constant such that $\forall i, \forall z \in \mathbf{GEN}(x_i)$
$||\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z)|| \leq R$

Proof: Direct modifi cation of the proof for the classifi cation case.

Proof:

Let $\mathbf{W}^k$ be the weights before the k'th mistake. $\mathbf{W}^1 = 0$
If the k'th mistake is made at i'th example,
and $z_i = \text{argmax}_{y \in \mathbf{GEN}(x_i)} \mathbf{f}(x_i, y) \cdot \mathbf{W}^k$, then

$$\begin{aligned}
\mathbf{W}^{k+1} &= \mathbf{W}^k + \mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i) \\
\Rightarrow \mathbf{U} \cdot \mathbf{W}^{k+1} &= \mathbf{U} \cdot \mathbf{W}^k + \mathbf{U} \cdot \mathbf{f}(x_i, y_i) - \mathbf{U} \cdot \mathbf{f}(x_i, z_i) \\
&\geq \mathbf{U} \cdot \mathbf{W}^k + \delta \\
&\geq k\delta \\
\Rightarrow ||\mathbf{W}^{k+1}|| &\geq k\delta
\end{aligned}$$

Also,

$$\begin{aligned}
||\mathbf{W}^{k+1}||^2 &= ||\mathbf{W}^k||^2 + ||\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i)||^2 + 2\mathbf{W}^k \cdot (\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, z_i)) \\
&\leq ||\mathbf{W}^k||^2 + R^2 \\
\Rightarrow ||\mathbf{W}^{k+1}||^2 &\leq kR^2 \\
\Rightarrow k^2\delta^2 &\leq ||\mathbf{W}^{k+1}||^2 \leq kR^2 \\
\Rightarrow k &\leq R^2/\delta^2
\end{aligned}$$

Perceptron Experiments: Parse Reranking

Parsing the Wall Street Journal Treebank
Training set = 40,000 sentences, test = 2,416 sentences
Generative model (Collins 1999): 88.2% F-measure
Reranked model: 89.5% F-measure (11% relative error reduction)
Boosting: 89.7% F-measure (13% relative error reduction)

- Results from Charniak and Johnson, 2005:
  - Improvement from 89.7% (baseline generative model) to 91.0% accuracy
  - Uses a log-linear model
  - Gains from improved n-best lists, better features

Summary

- A new framework: **global linear models**
  $\mathbf{GEN}$, $\mathbf{f}$, $\mathbf{W}$
- There are several ways to train the parameters $\mathbf{W}$:
  - Perceptron
  - Boosting
  - Log-linear models (maximum-likelihood)
- Applications:
  - Reranking models
  - LFG parsing
  - Generation
  - Machine translation
  - Tagging problems
  - Speech recognition