Log-Linear Models for History-Based Parsing

Michael Collins, Columbia University

Log-Linear Taggers: Summary

- The input sentence is $w_{[1:n]} = w_1 \dots w_n$
- Each tag sequence $t_{[1:n]}$ has a conditional probability

$$p(t_{[1:n]} \mid w_{[1:n]}) = \prod_{j=1}^{n} p(t_j \mid w_1 \dots w_n, t_1 \dots t_{j-1})$$ Chain rule

$$= \prod_{j=1}^{n} p(t_j \mid w_1 \dots w_n, t_{j-2}, t_{j-1})$$ Independence assumptions

- Estimate $p(t_j \mid w_1 \dots w_n, t_{j-2}, t_{j-1})$ using log-linear models
- Use the Viterbi algorithm to compute

$$\text{argmax}_{t_{[1:n]}} \log p(t_{[1:n]} \mid w_{[1:n]})$$

A General Approach: (Conditional) History-Based Models

- We've shown how to define $p(t_{[1:n]} \mid w_{[1:n]})$ where $t_{[1:n]}$ is a tag sequence
- How do we define $p(T \mid S)$ if T is a parse tree (or another structure)? (We use the notation $S = w_{[1:n]}$)

A General Approach: (Conditional) History-Based Models

- Step 1: represent a tree as a sequence of **decisions** $d_1 \ldots d_m$

  $$T = \langle d_1, d_2, \ldots d_m \rangle$$

  m is **not** necessarily the length of the sentence

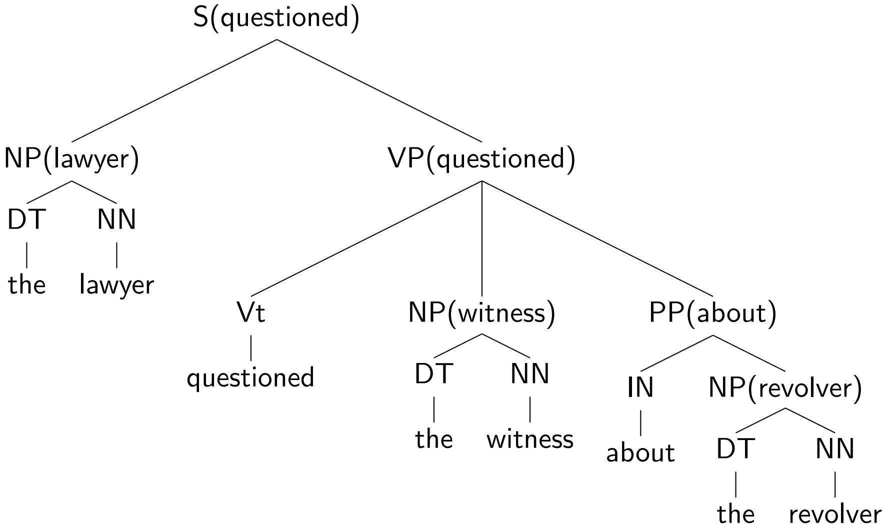
- Step 2: the probability of a tree is

  $$p(T \mid S) = \prod_{i=1}^{m} p(d_i \mid d_1 \ldots d_{i-1}, S)$$

- Step 3: Use a log-linear model to estimate $p(d_i \mid d_1 \ldots d_{i-1}, S)$

- Step 4: Search?? (answer we'll get to later: beam or heuristic search)

An Example Tree



Ratnaparkhi's Parser: Three Layers of Structure

1. Part-of-speech tags
2. Chunks
3. Remaining structure

Layer 1: Part-of-Speech Tags

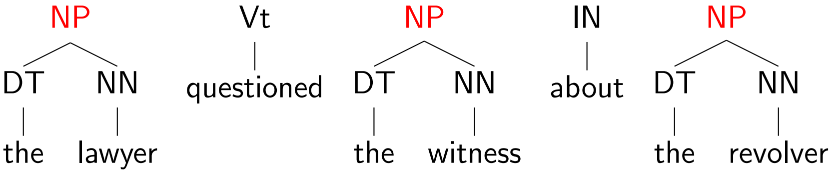
DT	NN	Vt	DT	NN	IN	DT	NN
the	lawyer	questioned	the	witness	about	the	revolver

- Step 1: represent a tree as a sequence of **decisions** $d_1 \dots d_m$

$$T = \langle d_1, d_2, \dots d_m \rangle$$

- First n decisions are tagging decisions
  $\langle d_1 \dots d_n \rangle = \langle$ DT, NN, Vt, DT, NN, IN, DT, NN $\rangle$

Layer 2: Chunks



Chunks are defined as any phrase where all children are part-of-speech tags

(Other common chunks are `ADJP`, `QP`)

Layer 2: Chunks

Start(NP)	Join(NP)	Other	Start(NP)	Join(NP)	Other	Start(NP)	Join(NP)
DT	NN	Vt	DT	NN	IN	DT	NN
the	lawyer	questioned	the	witness	about	the	revolver

- Step 1: represent a tree as a sequence of **decisions** $d_1 \ldots d_m$

$$T = \langle d_1, d_2, \ldots d_m \rangle$$

- First n decisions are tagging decisions
  Next n decisions are chunk tagging decisions

$$\langle d_1 \ldots d_{2n} \rangle = \langle \text{ DT, NN, Vt, DT, NN, IN, DT, NN, Start(NP), Join(NP), Other, Start(NP), Join(NP), Other, Start(NP), Join(NP)} \rangle$$

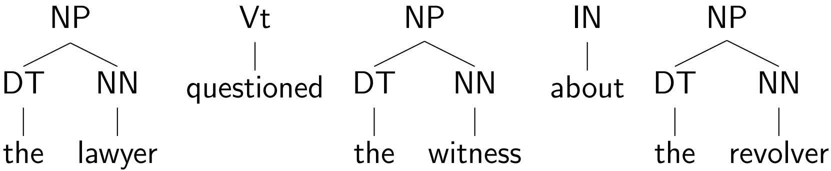
Layer 3: Remaining Structure

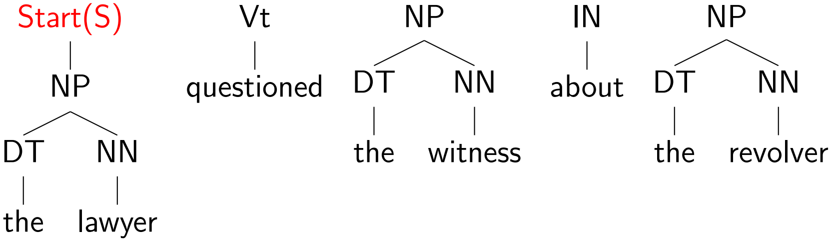
Alternate Between Two Classes of Actions:

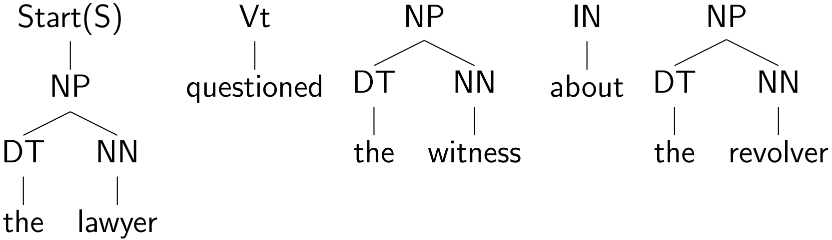
- Join(X) or Start(X), where X is a label (NP, S, VP etc.)
- Check=YES or Check=NO

Meaning of these actions:

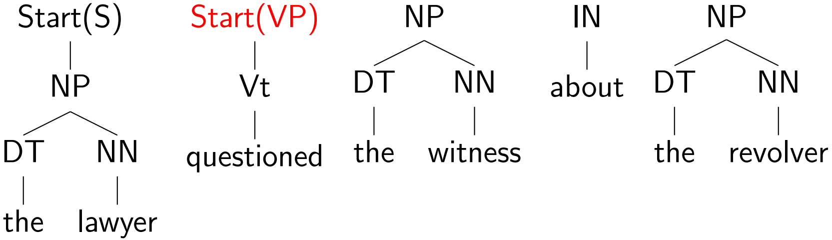
- Start(X) starts a new constituent with label X
  (always acts on leftmost constituent with no start or join label above it)
- Join(X) continues a constituent with label X
  (always acts on leftmost constituent with no start or join label above it)
- Check=NO does nothing
- Check=YES takes previous Join or Start action, and converts it into a completed constituent

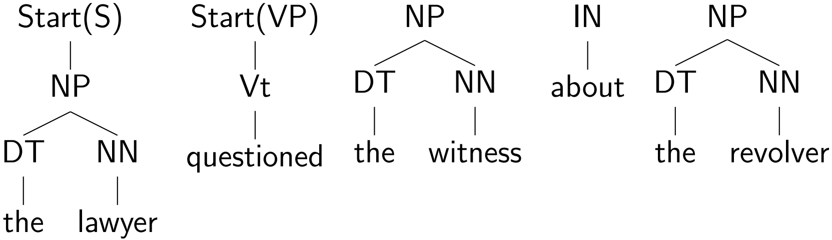

NP
DT
NN
the
lawyer
Vt
questioned
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver


Start(S)
NP
DT
NN
the
lawyer
Vt
questioned
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver

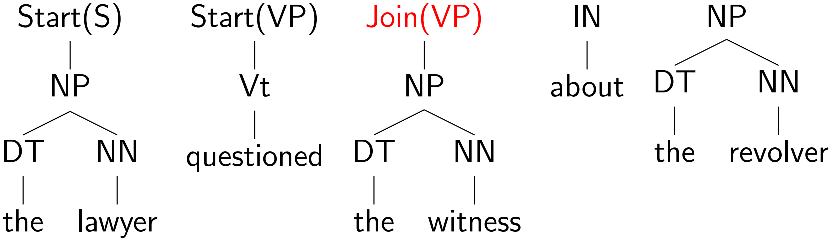

Start(S)
NP
DT
NN
the
lawyer
Vt
questioned
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver

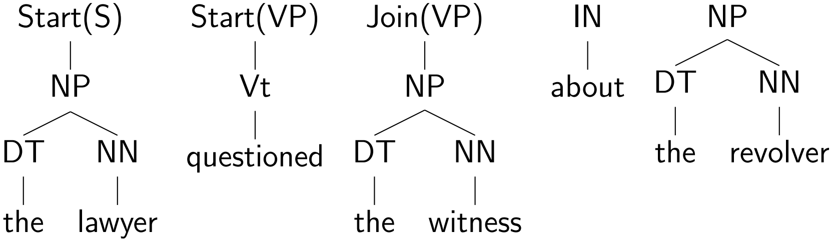
Check=NO


Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver

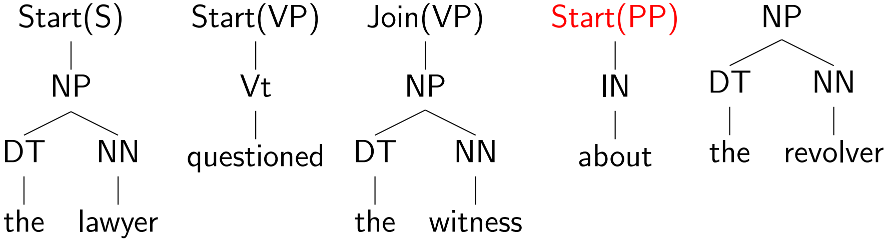

Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver

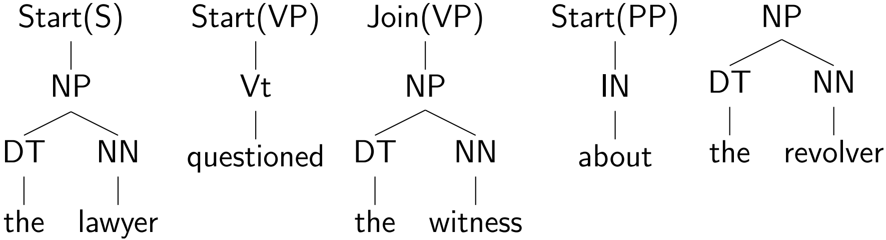
Check=NO


Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver

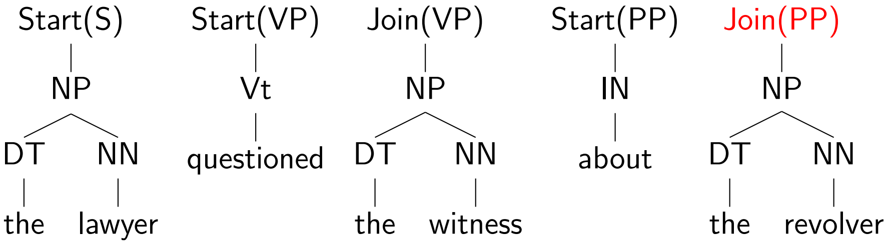

Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
IN
about
NP
DT
NN
the
revolver

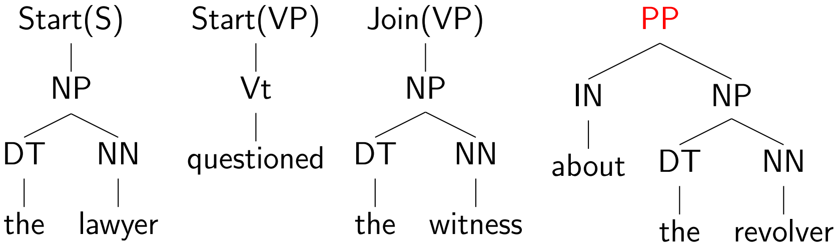
Check=NO


Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
Start(PP)
IN
about
NP
DT
NN
the
revolver

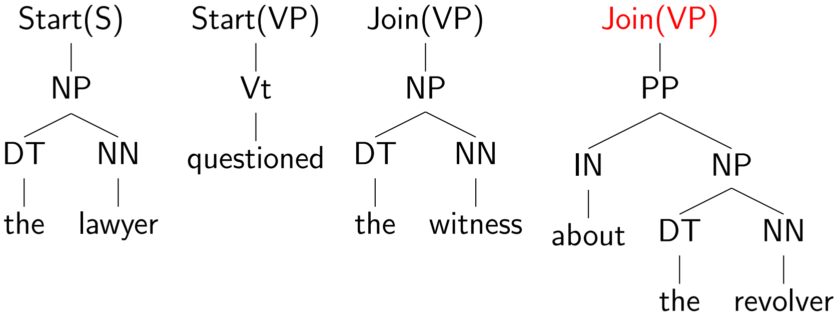

Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
Start(PP)
IN
about
NP
DT
NN
the
revolver

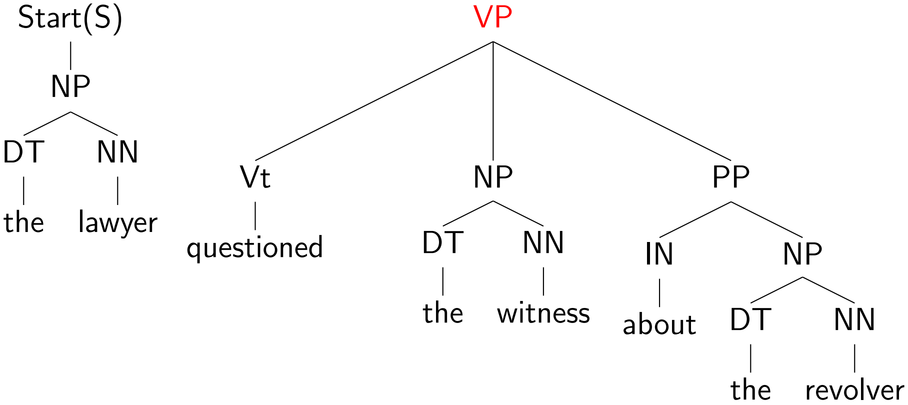
Check=NO


Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
Start(PP)
IN
about
Join(PP)
NP
DT
NN
the
revolver

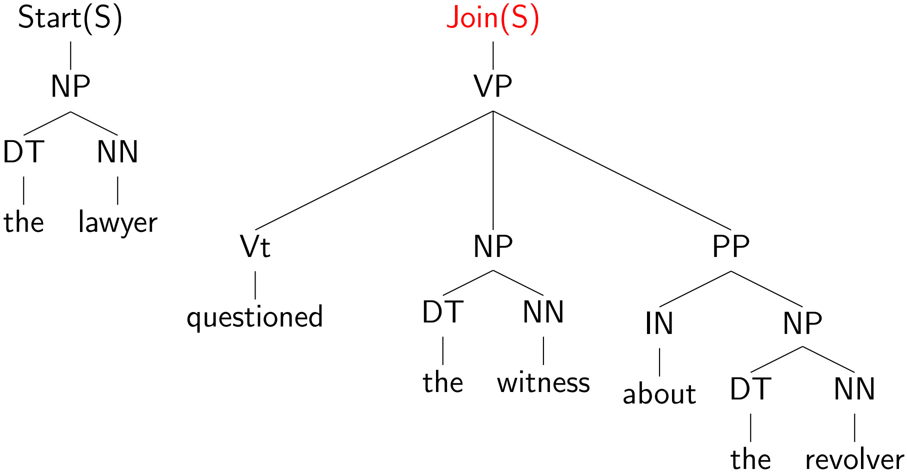

Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
PP
IN
NP
about
DT
NN
the
revolver

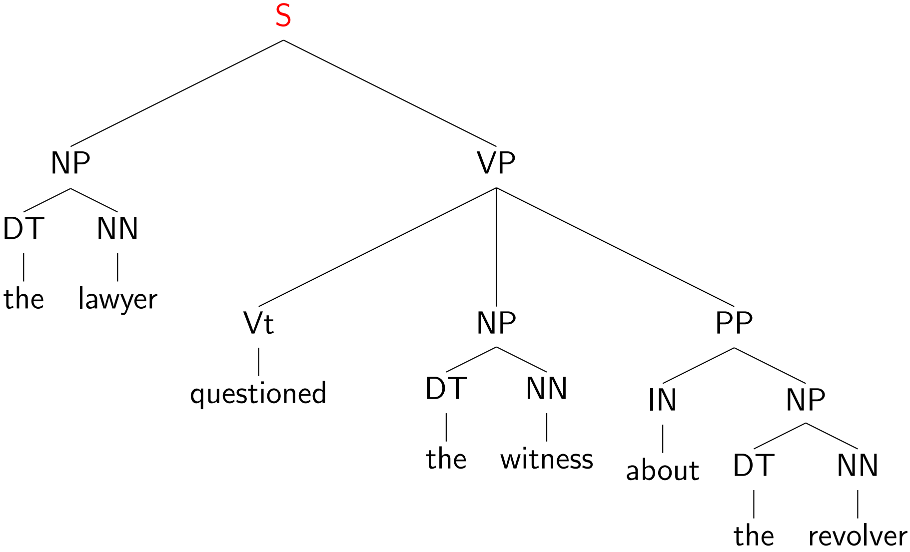
Check=YES


Start(S)
NP
DT
NN
the
lawyer
Start(VP)
Vt
questioned
Join(VP)
NP
DT
NN
the
witness
Join(VP)
PP
IN
NP
about
DT
NN
the
revolver


Start(S)
NP
DT
NN
the
lawyer
VP
Vt
questioned
NP
DT
NN
the
witness
PP
IN
about
NP
DT
NN
the
revolver

Check=YES


Start(S)
NP
DT
NN
the
lawyer
Join(S)
VP
Vt
NP
PP
questioned
DT
NN
the
witness
IN
NP
about
DT
NN
the
revolver


S
NP
VP
DT
NN
the
lawyer
Vt
NP
PP
questioned
DT
NN
IN
NP
the
witness
about
DT
NN
the
revolver

Check=YES

The Final Sequence of decisions

$\langle d_1 \ldots d_m \rangle = \langle$ DT, NN, Vt, DT, NN, IN, DT, NN,
Start(NP), Join(NP), Other, Start(NP), Join(NP),
Other, Start(NP), Join(NP),
Start(S), Check=NO, Start(VP), Check=NO,
Join(VP), Check=NO, Start(PP), Check=NO,
Join(PP), Check=YES, Join(VP), Check=YES,
Join(S), Check=YES $\rangle$

A General Approach: (Conditional) History-Based Models

- Step 1: represent a tree as a sequence of **decisions** $d_1 \ldots d_m$

  $$T = \langle d_1, d_2, \ldots d_m \rangle$$

  m is **not** necessarily the length of the sentence

- Step 2: the probability of a tree is $p(T \mid S) = \prod_{i=1}^{m} p(d_i \mid d_1 \ldots d_{i-1}, S)$

- Step 3: Use a log-linear model to estimate

  $$p(d_i \mid d_1 \ldots d_{i-1}, S)$$

- Step 4: Search?? (answer we'll get to later: beam or heuristic search)

Applying a Log-Linear Model

- Step 3: Use a log-linear model to estimate

$$p(d_i \mid d_1 \ldots d_{i-1}, S)$$

- A reminder:

$$p(d_i \mid d_1 \ldots d_{i-1}, S) = \frac{e^{f(\langle d_1 \ldots d_{i-1}, S \rangle, d_i) \cdot v}}{\sum_{d \in \mathcal{A}} e^{f(\langle d_1 \ldots d_{i-1}, S \rangle, d) \cdot v}}$$

where:

$\langle d_1 \ldots d_{i-1}, S \rangle$	is the history
d_i	is the outcome
f	maps a history/outcome pair to a feature vector
v	is a parameter vector
$\mathcal{A}$	is set of possible actions

Applying a Log-Linear Model

- Step 3: Use a log-linear model to estimate

$$p(d_i \mid d_1 \ldots d_{i-1}, S) = \frac{e^{f(\langle d_1 \ldots d_{i-1}, S \rangle, d_i) \cdot v}}{\sum_{d \in \mathcal{A}} e^{f(\langle d_1 \ldots d_{i-1}, S \rangle, d) \cdot v}}$$

- The big question: how do we define f?
- Ratnaparkhi's method defines f differently depending on whether next decision is:
  - A tagging decision
    (same features as before for POS tagging!)
  - A chunking decision
  - A start/join decision after chunking
  - A check=no/check=yes decision

Layer 3: Join or Start

- Looks at head word, constituent (or POS) label, and start/join annotation of n'th tree relative to the decision, where $n = -2, -1$
- Looks at head word, constituent (or POS) label of n'th tree relative to the decision, where $n = 0, 1, 2$
- Looks at bigram features of the above for (-1,0) and (0,1)
- Looks at trigram features of the above for (-2,-1,0), (-1,0,1) and (0, 1, 2)
- The above features with all combinations of head words excluded
- Various punctuation features

Layer 3: Check=NO or Check=YES

- A variety of questions concerning the proposed constituent

The Search Problem

- In POS tagging, we could use the Viterbi algorithm because

$$p(t_j \mid w_1 \dots w_n, j, t_1 \dots t_{j-1}) = p(t_j \mid w_1 \dots w_n, j, t_{j-2} \dots t_{j-1})$$

- Now: Decision d_i could depend on arbitrary decisions in the "past" $\Rightarrow$ no chance for dynamic programming

- Instead, Ratnaparkhi uses a beam search method