MaxEnt (IV): Case study and Beam search

LING 572

Advanced Statistical Methods for NLP

February 6, 2020

Case study

POS tagging
(Ratnaparkhi, 1996)

- Notation variation:
  - $f_j(x, y)$: x: input, y: output
  - $f_j(h, t)$: h: history, t: tag for the word

- History: $h_i = \{w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}, t_{i-1}, t_{i-2}\}$

- Training data:
  - Treat a sentence as a set of (h_i, t_i) pairs.
  - How many pairs are there for a sentence?

Using a MaxEnt Model

- Modeling:
- Training:
  - Define feature templates
  - Create the feature set
  - Determine the optimum feature weights via GIS or IIS
- Decoding:

Modeling

$$P(t_1,\ldots,t_n \mid w_1,\ldots,w_n)$$

$$= \prod_{i=1}^{n} p(t_i \mid w_1^n, t_1^{i-1})$$

$$\approx \prod_{i=1}^{n} p(t_i \mid h_i)$$

$$p(t \mid h) = \frac{p(h,t)}{\sum_{t' \in T} p(h,t')}$$

Training step 1: define feature templates

Condition	Features	
w_i is not rare	$w_i = X$	$\& \; t_i = T$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$	$\& \; t_i = T$
	X is suffix of w_i, $\|X\| \leq 4$	$\& \; t_i = T$
	w_i contains number	$\& \; t_i = T$
	w_i contains uppercase character	$\& \; t_i = T$
	w_i contains hyphen	$\& \; t_i = T$
$\forall \; w_i$	$t_{i-1} = X$	$\& \; t_i = T$
	$t_{i-2}t_{i-1} = XY$	$\& \; t_i = T$
	$w_{i-1} = X$	$\& \; t_i = T$
	$w_{i-2} = X$	$\& \; t_i = T$
	$w_{i+1} = X$	$\& \; t_i = T$
	$w_{i+2} = X$	$\& \; t_i = T$

History h_i

Tag t_i

Step 2: Create feature set

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

$$
\begin{aligned}
&w_i = \texttt{about} && \& \; t_i = \texttt{IN} \\
&w_{i-1} = \texttt{stories} && \& \; t_i = \texttt{IN} \\
&w_{i-2} = \texttt{the} && \& \; t_i = \texttt{IN} \\
&w_{i+1} = \texttt{well-heeled} && \& \; t_i = \texttt{IN} \\
&w_{i+2} = \texttt{communities} && \& \; t_i = \texttt{IN} \\
&t_{i-1} = \text{NNS} && \& \; t_i = \texttt{IN} \\
&t_{i-2}t_{i-1} = \text{DT NNS} && \& \; t_i = \texttt{IN}
\end{aligned}
$$

➔ Collect all the features from the training data

➔ Throw away features that appear less than 10 times

The thresholds

- Rare words: words that occur < 5 in the training data.

- Features (not feature functions):
  - All curWord features will be kept.
  - For the rest of features, keep them if they occur >= 10 in the training data.

Step 3: determine the weights of feature functions

- GIS

- Training time:
  - Each iteration: O(NTA):
    - N: the training set size
    - T: the number of allowable tags
    - A: average number of features that are active for a (h, t).
  - About 24 hours on a 1996 machine (an IBM RS/6000 Model 380)
    - Much much faster now

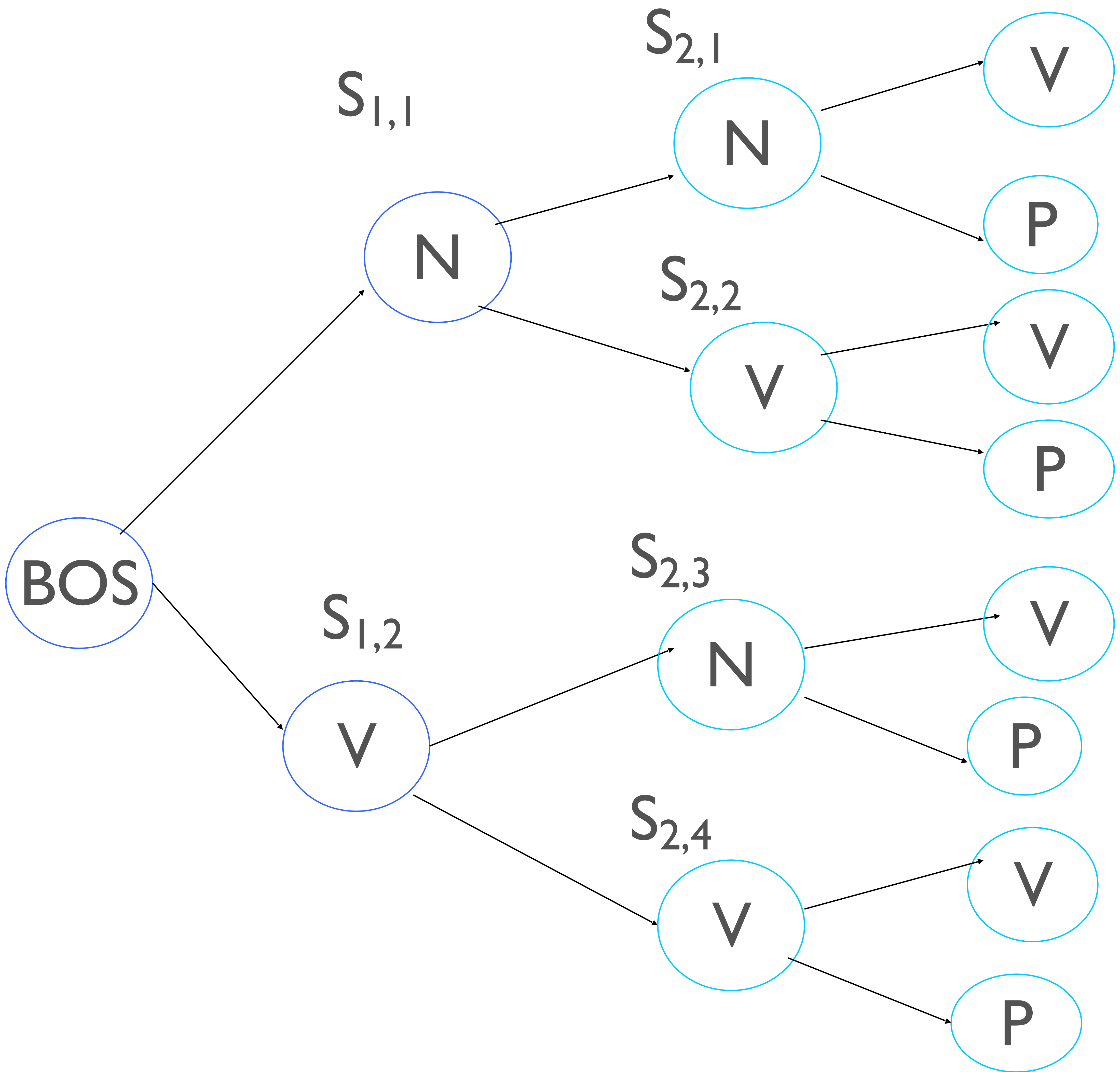
Beam search

Why do we need beam search?

- Features refer to tags of previous words, which are not available for the TEST data.
- Knowing only the best tag of the previous word is not good enough.
- So let’s keep multiple tag sequences available during decoding.

Beam search

time flies like an arrow



Beam Search

- Intuition:
  - Breadth-first search explores all paths
  - Lots of paths are (pretty obviously) bad
  - Why explore bad paths?
  - Restrict to (apparently best) paths

- Approach:
  - Perform breadth-first search, *but*
  - Retain only top *k* ‘best’ paths thus far

Parameters: topN, topK, beam_size

(1) Get topN tags for w_1 and form nodes $s_{1,j}$

(2) For i=2 to n (n is the sentence length)

For each surviving node $s_{i-1,j}$

form the vector for w_i

get $\boxed{\text{topN}}$ tags for w_i and

form new nodes

Prune nodes at position i

(3) Pick the node at position n with highest prob

Pruning at Position i

Each node at Position i should store a tag for w_i and a prob, where the prob is $\prod_{k=1}^{i} P(t_k|h_k)$.

Let max_prob be the highest prob among the nodes at Position i

For each node $s_{i,j}$ at Position i

Let $prob_{i,j}$ be the probability stored at the node

keep the node iff $prob_{i,j}$ is among the $\boxed{\text{topK}}$ of the nodes

and $lg(prob_{i,j}) + \boxed{\text{beam\_size}} \geq lg(max\_prob)$

Decoding (cont)

- Tags for words:
  - Known words: use tag dictionary
  - Unknown words: try all possible tags

- Ex: “time flies like an arrow”

- Running time: O(NTAB)
  - N: sentence length
  - B: beam size
  - T: tagset size
  - A: average number of features that are active for a given event

POS Tagging

- Overall accuracy: 96.3+%
- Unseen word accuracy: 86.2%

- Comparable to HMM tagging accuracy or TBL

- Provides
  - Probabilistic framework
  - Better able to model different info sources

- Topline accuracy 96-97%
  - Consistency issues

Experiment results

MF tag	O	7.66	
Markov 1-gram	B	6.74	
Markov 3-gram	W	3.7	
Markov 3-gram	B	3.64	
Decision tree	M	3.5	
Transformation	B	3.39	
Maxent	R	3.37	
Maxent	O	3.11	±.07
Multi-tagger Voting	B	2.84	±.03

Beam Search

- Beam search decoding:
  - Variant of breadth first search
  - At each layer, keep only top sequences

- Advantages:
  - Efficient in practice: beam 3-5 near optimal
    - Empirically, beam 5-10% of search space; prunes 90-95%
  - Simple to implement
    - Just extensions + sorting, no dynamic programming

- Applies much more broadly than just MaxEnt models
- Disadvantage: Not guaranteed optimal (or complete)

MaxEnt POS Tagging

- Part-of-speech tagging by classification:
  - Feature design
    - word and tag context features
    - orthographic features for rare words
- Sequence classification problems:
  - Tag features depend on prior classification
- Beam search decoding
  - Efficient, but inexact
    - Near optimal in practice

Comparison with other learners

- HMM: MaxEnt can use more context

- DT: MaxEnt does not split data

- Naïve Bayes: MaxEnt does not assume that features are independent given the class.