Sequence models

IN4080
Natural Language Processing

Yves Scherrer

Sequence labeling

Task:

- Predict a label for each element of a sequence
- I.e. predict a label for each word of a sentence

Applications:

- Part-of-speech tagging
- Named entity recognition

Models:

- Naive Bayes (unigram model)
- Hidden Markov Model with greedy decoding
- Hidden Markov Model with Viterbi decoding

Bigram Hidden Markov Models

Training:

- For each x, y in training data:
  - Compute and store $P_E(x|y) = \frac{Count(x,y)}{Count(y)}$
- For each bigram y_{i-1}, y_i in training data:
  - Compute and store $P_T(y_i|y_{i-1}) = \frac{Count(y_{i-1},y_i)}{Count(y_{i-1})}$

Same for greedy and Viterbi decoding

Testing/Prediction:

- For each sentence in test data:

$$\widehat{y_{1\ldots n}} = \arg\max_{y_{1..n} \in Y^*} \left(\prod_{i=1}^{n} P_E(x_i|y_i) \cdot P_T(y_i|y_{i-1}) \right)$$

  - That's one big computation for the whole sentence
  - This computation is intractable – we need some tricks…

Trick 1: Dynamic programming

A lot of repetitions!

Assume 4 words, 2 tags (A, B). That's $2^4 = 16$ computations, 112 operations:

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$

Trick 1: Dynamic programming

Let's proceed one position at a time and save the intermediate results:

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) = \pi_A$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) = \pi_B$

- $\pi_A \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) = \pi_{AA}$
- $\pi_A \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) = \pi_{AB}$
- $\pi_B \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) = \pi_{BA}$
- $\pi_B \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) = \pi_{BB}$

- $\pi_{AA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{AAA}$
- …

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats	
2 sequences:	4 sequences:	8 sequences:	16 sequences:	
* N	* N N	* N N N	* N N N N	* V N N N
* V	* N V	* N N V	* N N N V	* V N N V
	* V N	* N V N	* N N V N	* V N V N
	* V V	* N V V	* N N V V	* V N V V
		* V N N	* N V N N	* V V N N
		* V N V	* N V N V	* V V N V
		* V V N	* N V V N	* V V V N
		* V V V	* N V V V	* V V V V

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats	
2 sequences:	4 sequences:	8 sequences:	16 sequences:	
* N	* N N	* N N N	* N N N N	* V N N N
* V	* N V	* N N V	* N N N V	* V N N V
	* V N	* N V N	* N N V N	* V N V N
	* V V	* N V V	* N N V V	* V N V V
		* V N N	* N V N N	* V V N N
		* V N V	* N V N V	* V V N V
		* V V N	* N V V N	* V V V N
		* V V V	* N V V V	* V V V V

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats	
2 sequences:	4 sequences:	8 sequences:	16 sequences:	
* N	* N N	* N N N	* N N N N	* V N N N
* V	* N V	* N N V	* N N N V	* V N N V
	* V N	* N V N	* N N V N	* V N V N
	* V V	* N V V	* N N V V	* V N V V
		* V N N	* N V N N	* V V N N
		* V N V	* N V N V	* V V N V
		* V V N	* N V V N	* V V V N
		* V V V	* N V V V	* V V V V

Example

fish/N $1 \cdot \frac{5}{6} \cdot \frac{3}{13} = \frac{5}{26}$

fish/V $1 \cdot \frac{1}{6} \cdot \frac{2}{10} = \frac{1}{30}$

fish/N dogs/N $\frac{5}{26} \cdot \frac{1}{6} \cdot \frac{3}{13} = \frac{5}{676}$

fish/N dogs/V $\frac{5}{26} \cdot \frac{5}{6} \cdot \frac{1}{10} = \frac{5}{312}$

fish/V dogs/N $\frac{1}{30} \cdot \frac{4}{5} \cdot \frac{3}{13} = \frac{2}{325}$

fish/V dogs/V $\frac{1}{30} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{1500}$

fish/N dogs/N like/N $\frac{5}{676} \cdot \frac{1}{6} \cdot \frac{1}{13} = \frac{5}{52728}$

fish/N dogs/N like/V $\frac{5}{676} \cdot \frac{5}{6} \cdot \frac{2}{10} = \frac{5}{4056}$

fish/N dogs/V like/N $\frac{5}{312} \cdot \frac{4}{5} \cdot \frac{1}{13} = \frac{1}{1014}$

fish/N dogs/V like/V $\frac{5}{312} \cdot \frac{1}{5} \cdot \frac{2}{10} = \frac{1}{1560}$

fish/V dogs/N like/N $\frac{2}{325} \cdot \frac{1}{6} \cdot \frac{1}{13} = \frac{1}{12675}$

fish/V dogs/N like/V $\frac{2}{325} \cdot \frac{5}{6} \cdot \frac{2}{10} = \frac{1}{975}$

fish/V dogs/V like/N $\frac{1}{1500} \cdot \frac{4}{5} \cdot \frac{1}{13} = \frac{1}{24375}$

fish/V dogs/V like/V $\frac{1}{1500} \cdot \frac{1}{5} \cdot \frac{2}{10} = \frac{1}{37500}$

Trick 2: The Markov assumption

Ultimately, we are interested in the sequence with the maximum probability. We can identify uninteresting paths and skip them.

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) = \pi_A$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) = \pi_B$

- $\pi_A \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) = \pi_{AA}$
- $\pi_A \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) = \pi_{AB}$
- $\pi_B \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) = \pi_{BA}$
- $\pi_B \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) = \pi_{BB}$

If $\pi_{AA} < \pi_{BA}$, then $\pi_{AAi} < \pi_{BAi}$ for any i.

- ~~$\pi_{AA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{AAA}$~~
- $\pi_{BA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{BAA}$

We can skip all computations starting with π_{AA}.

Trick 2: The Markov assumption

The prediction formula only depends on the previous label, not on all labels back to y_0:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^*} \prod_{i=1}^{n} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

This is called the (bigram) **Markov assumption**.

- At each position, we have to consider each label and each path from a previous label.
- But there is only one best path towards that previous label.
- The number of paths to consider does not grow exponentially, but remains at $|Y|^2$ at each position.

Trick 2: The Markov assumption

Ultimately, we are interested in the sequence with the maximum probability. We can identify uninteresting paths and skip them.

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) = \pi_A$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) = \pi_B$

- $\pi_A \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) = \pi_{AA}$
- $\pi_A \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) = \pi_{AB}$
- $\pi_B \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) = \pi_{BA}$
- $\pi_B \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) = \pi_{BB}$

- ~~$\pi_{AA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{AAA}$~~
- $\pi_{BA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{BAA}$

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats
2 sequences: * N * V	4 sequences: * N N * N V * V N * V V	8 sequences: * N N N * N N V * N V N * N V V ~~* V N N~~ * V N V * V V N * V V V	16 sequences: * N N V V * V N V V * N V V N * V V V N * N V V V * V V V V

$P(* \; N \; N) = 0.007$

$P(* \; V \; N) = 0.006$

$P(* \; N \; N \; N) =$
$0.007 \cdot P_T(N \rightarrow N) \cdot P_E(N \rightarrow like)$

$P(* \; V \; N \; N) =$
$0.006 \cdot P_T(N \rightarrow N) \cdot P_E(N \rightarrow like)$

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats
2 sequences: * N * V	4 sequences: * N N * N V * V N * V V	8 sequences: * N N N * N N V * N V N * N V V * V N N ~~* V N V~~ * V V N * V V V	16 sequences: * N N N N * V N N N ... V * N N V N * V N V N * N N V V * V N V V ... N ... V * N V V N * V V V N * N V V V * V V V V

$P(* \; N \; N \; V) =$
$0.007 \cdot P_T(N \rightarrow V) \cdot P_E(V \rightarrow like)$

$P(* \; V \; N \; V) =$
$0.006 \cdot P_T(N \rightarrow V) \cdot P_E(V \rightarrow like)$

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats	
2 sequences:	4 sequences:	8 sequences:	16 sequences:	
* N	* N N	* N N N	* N N N N	~~* V N N N~~
* V	* N V	* N N V	* N N N V	~~* V N N V~~
	* V N	* N V N	* N N V N	~~* V N V N~~
	* V V	* N V V	* N N V V	~~* V N V V~~
		~~* V N N~~	* N V N N	~~* V V N N~~
		~~* V N V~~	* N V N V	~~* V V N V~~
		~~* V V N~~	* N V V N	~~* V V V N~~
		~~* V V V~~	* N V V V	~~* V V V V~~

$P(* \, N \, V) > P(* \, V \, V)$

Example

Test sentence: fish dogs like cats

fish	fish dogs	fish dogs like	fish dogs like cats
2 sequences: * N * V	4 sequences: * N N * N V * V N * V V	4 sequences: * N N N * N N V * N V N * N V V ~~* V N N~~ ~~* V N V~~ ~~* V V N~~ ~~* V V V~~	4 sequences: ~~* N N N N~~ ~~* N N N V~~ * N N V N * N N V V * N V N N * N V N V ~~* N V V N~~ ~~* N V V V~~ ~~* V N N N~~ ~~* V N N V~~ ~~* V N V N~~ ~~* V N V V~~ ~~* V V N N~~ ~~* V V N V~~ ~~* V V V N~~ ~~* V V V V~~

$$P(* \ N \ N \ N) < P(* \ N \ V \ N)$$

$$P(* \ N \ N \ V) > P(* \ N \ V \ V)$$

The Viterbi algorithm

Setup		Max computations	Multiplications
m words, n labels	Brute force	m^n	$m^n \cdot 2 \cdot m$
	Viterbi	$m \cdot n^2$	$m \cdot n^2 \cdot 2$
4 words, 2 labels	Brute force	$2^4 = 16$	$16 \cdot 2 \cdot 4 = 122$
	Viterbi	$4 \cdot (2^2) = 16$	$16 \cdot 2 = 32$
5 words, 2 labels	Brute force	$2^5 = 32$	$32 \cdot 2 \cdot 5 = 320$
	Viterbi	$5 \cdot (2^2) = 20$	$20 \cdot 2 = 40$
6 words, 2 labels	Brute force	$2^6 = 64$	$64 \cdot 2 \cdot 6 = 768$
	Viterbi	$6 \cdot (2^2) = 24$	$24 \cdot 2 = 48$
20 words, 17 labels	Brute force	$17^{20} = 4e24$	$4e24 \cdot 2 \cdot 17 = 1.4e26$
	Viterbi	$20 \cdot (17^2) = 5780$	$5780 \cdot 2 = 11560$

The Viterbi algorithm

- Use a table $\pi[k, y_k]$ to store computations
  - Contains the maximum probability of the sequence sequence $x_1 \dots x_k$ ending in tag y_k
- Initialization:
  - $\pi[0,*] = 1$
- Recursive definition:
  - Fill π for all positions $k \in \{1 \dots n\}$ and all labels:
  - $\pi[k, y_k] = \max_{y_{k-1} \in Y} (\pi[k-1, y_{k-1}] \cdot P_T(y_k | y_{k-1}) \cdot P_E(x_k | y_k))$

The Viterbi algorithm

Input:

- sequence $X = x_1 \dots x_n$, label set Y, parameters P_E, P_T

Initialization:

- $\pi[0,*] = 1$

Algorithm:

- for each $k = 1 \dots n$: (columns / positions)
  - for each $y_k \in Y$: (rows / labels)
    - $\pi[k, y_k] = \max_{y_{k-1} \in Y} (\pi[k-1, y_{k-1}] \cdot P_T(y_k | y_{k-1}) \cdot P_E(x_k | y_k))$
- $p = \max_{y_n} (\pi[n, y_n])$
- return p

Example

Test sentence: fish dogs like cats

Computations:

	fish	dogs	like	cats
N	$1 \cdot \frac{5}{6} \cdot \frac{3}{13} = \mathbf{\frac{5}{26}}$	$\frac{5}{26} \cdot \frac{1}{6} \cdot \frac{3}{13} = \mathbf{\frac{5}{676}}$ $\frac{1}{30} \cdot \frac{4}{5} \cdot \frac{3}{13} = \frac{2}{325}$	$\frac{5}{676} \cdot \frac{1}{6} \cdot \frac{1}{13} = \frac{5}{52728}$ $\frac{5}{312} \cdot \frac{4}{5} \cdot \frac{1}{13} = \mathbf{\frac{1}{1014}}$	$\frac{1}{1014} \cdot \frac{1}{6} \cdot \frac{3}{13} = \frac{1}{26364}$ $\frac{5}{4056} \cdot \frac{4}{5} \cdot \frac{3}{13} = \mathbf{\frac{1}{4394}}$
V	$1 \cdot \frac{1}{6} \cdot \frac{2}{10} = \mathbf{\frac{1}{30}}$	$\frac{5}{26} \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{\frac{5}{312}}$ $\frac{1}{30} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{1500}$	$\frac{5}{676} \cdot \frac{5}{6} \cdot \frac{2}{10} = \mathbf{\frac{5}{4056}}$ $\frac{5}{312} \cdot \frac{1}{5} \cdot \frac{2}{10} = \frac{1}{1560}$	$\frac{1}{1014} \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{\frac{1}{12168}}$ $\frac{5}{4056} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{40560}$

Backpointers

- The given algorithm just gives the final probability.
- In most cases, we are not so much interested in the probability, but rather in the actual labels predicted at each step.
- Solution: Identify the path through the table and trace backwards to the beginning.
  - Implementation: Use a second table to store these **backpointers**

Example

Test sentence: fish dogs like cats

Computations:

	fish	dogs	like	cats
N	$1 \cdot \frac{5}{6} \cdot \frac{3}{13} = \mathbf{\frac{5}{26}}$	$\frac{5}{26} \cdot \frac{1}{6} \cdot \frac{3}{13} = \mathbf{\frac{5}{676}}$ $\frac{1}{30} \cdot \frac{4}{5} \cdot \frac{3}{13} = \frac{2}{325}$	$\frac{5}{676} \cdot \frac{1}{6} \cdot \frac{1}{13} = \frac{5}{52728}$ $\frac{5}{312} \cdot \frac{4}{5} \cdot \frac{1}{13} = \mathbf{\frac{1}{1014}}$	$\frac{1}{1014} \cdot \frac{1}{6} \cdot \frac{3}{13} = \frac{1}{26364}$ $\frac{5}{4056} \cdot \frac{4}{5} \cdot \frac{3}{13} = \mathbf{\frac{1}{4394}}$
V	$1 \cdot \frac{1}{6} \cdot \frac{2}{10} = \mathbf{\frac{1}{30}}$	$\frac{5}{26} \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{\frac{5}{312}}$ $\frac{1}{30} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{1500}$	$\frac{5}{676} \cdot \frac{5}{6} \cdot \frac{2}{10} = \mathbf{\frac{5}{4056}}$ $\frac{5}{312} \cdot \frac{1}{5} \cdot \frac{2}{10} = \frac{1}{1560}$	$\frac{1}{1014} \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{\frac{1}{12168}}$ $\frac{5}{4056} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{40560}$
	N	N	V	N

Pseudo-code

- $\pi[0,*] = 1$
- for each $k = 1 \ldots n$:
  - for each $y_k \in Y$:
    - $\pi[k, y_k] = \max_{y_{k-1} \in Y} (\pi[k-1, y_{k-1}] \cdot P_T(y_k | y_{k-1}) \cdot P_E(x_k | y_k))$
    - $bp[k, y_k] = \arg\max_{y_{k-1} \in Y} (\pi[k-1, y_{k-1}] \cdot P_T(y_k | y_{k-1}) \cdot P_E(x_k | y_k))$
- $p = \max_{y_n \in Y} (\pi[n, y_n] \cdot P_T(\dagger \,| y_n))$
- $labels[n] = \arg\max_{y_n \in Y} \left(\pi[k, y_n] \cdot P_T(\dagger \,| y_n)\right)$
- for each $k = n-1 \ldots 1$:
  - $labels[k] = bp[k+1, labels[k+1]]$
- return $labels$

Store both the max and the argmax.

As for the start of the sentence, one may include an additional transition probability $P_T(\dagger \,| y_n)$ at the end of the sentence.

Recover the labels from the end to the beginning.

Different ways to look at the context

Option 1: Look at the neighboring words:

$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}\, P(y_i | x_{i-1}, x_i, x_{i+1})$$

- Perceptron, logistic regression

Let's look at this option next!

Option 2: Look at the previous tag decisions:

$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}\, P(y_i | x_i, y_{i-1})$$

- HMM

Option 3: Combine the two:

$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}\, P(y_i | x_{i-1}, x_i, x_{i+1}, y_{i-1})$$

- Structured perceptron, CRF, MEMM

Predicting one label per word

Naive Bayes prediction function:

$$\hat{y} = \arg\max_{y \in Y} \left(P(y) \cdot P(x|y) \right)$$

- $P(x|y)$ is a single value
  (no bag-of-words decomposition)

Perceptron prediction function:

$$\hat{y} = \arg\max_{y \in Y} \left(\boldsymbol{w_y} \cdot \boldsymbol{f_x} \right)$$

- $\boldsymbol{f_x}$ represents a single word and is (typically) a **one-hot vector**: 0 0 0 0 1 0 0 0 0 0 0 0

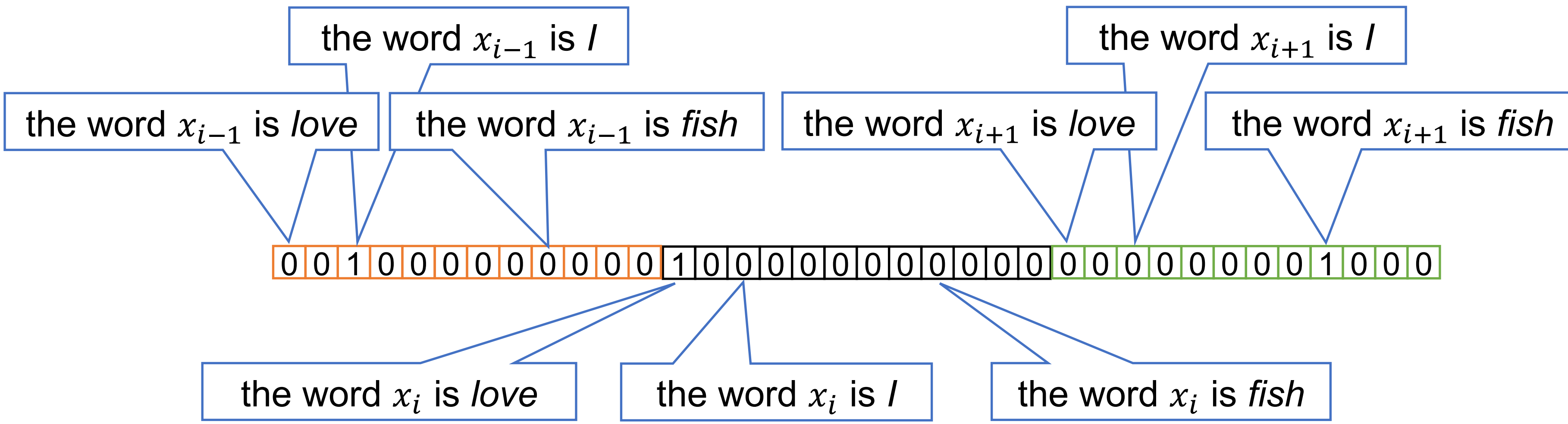
the word x is *love*

the word x is *I*

the word x is *fish*

Context word features

$$\boldsymbol{f}_{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{x}_{\boldsymbol{i-1}}} \oplus \boldsymbol{f}_{\boldsymbol{x}_{\boldsymbol{i}}} \oplus \boldsymbol{f}_{\boldsymbol{x}_{\boldsymbol{i+1}}}$$



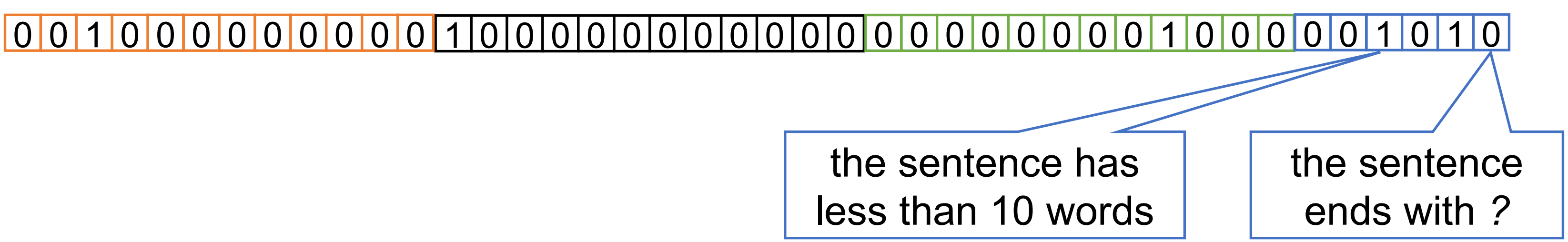
Example: *love* in the context *I* _ *fish*

Context word features

$$\boldsymbol{f}_{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{x}_{i-1}} \oplus \boldsymbol{f}_{\boldsymbol{x}_{i}} \oplus \boldsymbol{f}_{\boldsymbol{x}_{i+1}}$$

Using fixed-length pretrained word embeddings:

Using additional sentence-level features:



Features for unknown words

- x_i contains a particular prefix
- x_i contains a particular suffix
- x_i contains a number
- x_i contains an upper-case letter
- x_i contains a hyphen
- x_i is all upper case
- x_i is upper case and has a digit and a dash
- …

Discriminative classifiers with context word features

Perceptron prediction function:

$$\hat{y} = \arg\max_{y \in Y} (\boldsymbol{w_y} \cdot \boldsymbol{f_x})$$

- The feature vectors can be arbitrarily complex
- Training and prediction exactly the same as for document classification

Logistic regression:

- The feature vectors can be arbitrarily complex
- Training and prediction exactly the same as for document classification

Different ways to look at the context

Option 1: Look at the neighboring words:

$$\hat{y}_i = \arg\max_{y_i \in Y} P(y_i | x_{i-1}, x_i, x_{i+1})$$

- Perceptron, logistic regression

Option 2: Look at the previous tag decisions:

$$\hat{y}_i = \arg\max_{y_i \in Y} P(y_i | x_i, y_{i-1})$$

- HMM

Let's look at this option next!

Option 3: Combine the two:

$$\hat{y}_i = \arg\max_{y_i \in Y} P(y_i | x_{i-1}, x_i, x_{i+1}, y_{i-1})$$

- Structured perceptron, CRF, MEMM

Structured perceptron

(Greedy) HMM:

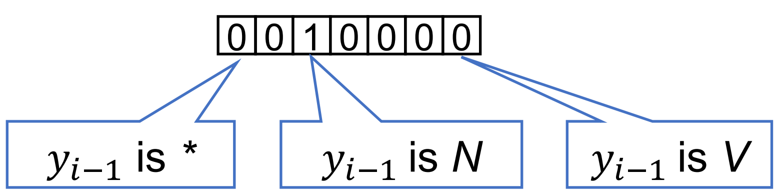
$$\hat{y}_i = \arg\max_{y_i \in Y}\left(P_T(y_i|y_{i-1}) \cdot P_E(x_i|y_i)\right)$$

- Contains transition and emission probabilities

(Greedy) structured perceptron:

$$\hat{y}_i = \arg\max_{y_i \in Y}\left(\boldsymbol{w}_T(y_i) \cdot \boldsymbol{f}_T(y_{i-1}) + \boldsymbol{w}_E(y_i) \cdot \boldsymbol{f}_E(x_i)\right)$$

- Contains transition and emission features
- Emission features can be arbitrarily complex
- Transition features: one-hot vector with $|Y|$ items

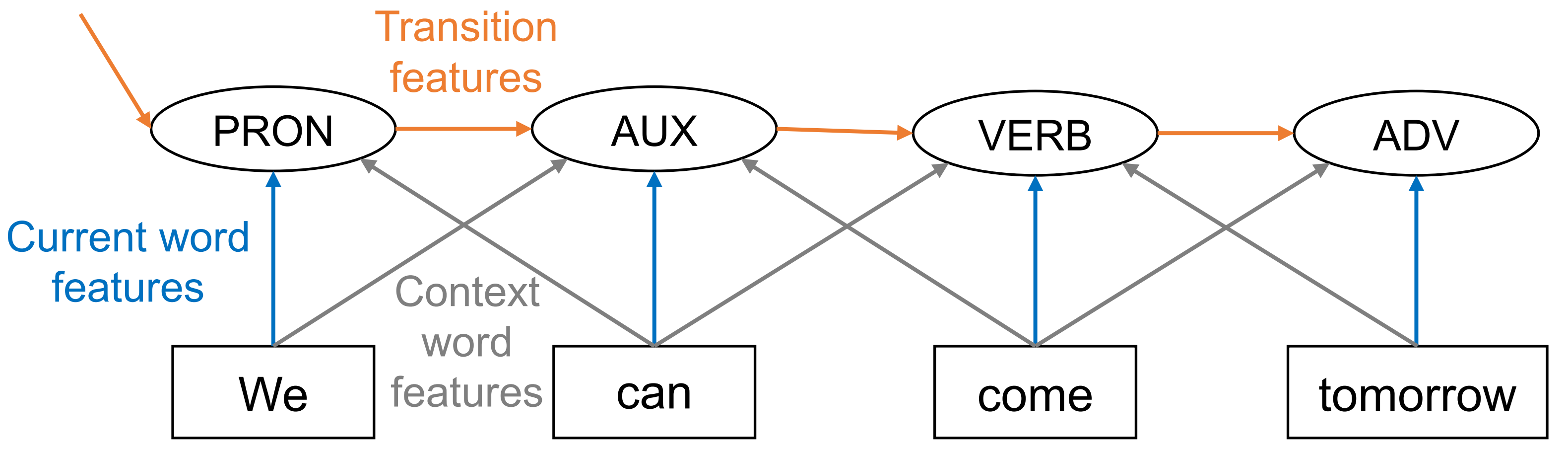


Transition features

- We don't actually need to build an explicit transition feature vector
- The transition weights can be thought of as a $Y \times Y$ matrix (as in HMMs): $\boldsymbol{w}_T(y_i, y_{i-1})$
  - The transition feature just selects one column of this matrix.

$$\widehat{y}_i = \underset{y_i \in Y}{\arg\max}(\boldsymbol{w}_T(y_i, y_{i-1}) + \boldsymbol{w}_E(y_i) \cdot \boldsymbol{f}_E(x_i))$$

Context and transition features



- Why / when can transition features be useful?

Structured perceptron
Decoding strategies

Greedy decoding:

$$S(x_{1..n}, \widehat{y_{1..n}}) = \sum_{i=1}^{n} \max_{y_i \in Y}(w_T(y_i, y_{i-1}) + w_E(y_i) \cdot f_E(x_i))$$

Exact (Viterbi) decoding:

$$S(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^n} \left(\sum_{i=1}^{n} (w_T(y_i, y_{i-1}) + w_E(y_i) \cdot f_E(x_i)) \right)$$

- Essentially identical to Viterbi for HMMs
- The main operations are sums, so the base case is 0, not 1

Structured perceptron Training

In a perceptron, training involves prediction:

- Take one training instance $\boldsymbol{x}_{1 \ldots n}$
- Predict the label sequence $\widehat{\boldsymbol{y}_{1 \ldots n}}$ using the current weight vectors
- For each position $1 \ldots n$:
  - If the prediction is correct, nothing happens
  - If the prediction is wrong, modify the parameters of the model:
    - add the feature values to the weight vectors of the correct label
    - subtract the feature values from the weight vectors of the predicted (wrong) label
- Continue "until tired"

Structured perceptron Training

procedure train_structured_perceptron (D):

initialize w_E and w_T

repeat:

for each sentence x with label sequence y in D:

y_hat = predict_viterbi(x, w)

for i in len(y):

if y_hat_i != y_i :

$w_E(y_i)$ += f(x, i)

$w_E(y\_hat_i)$ -= f(x, i)

if y_hat_i != y_i or y_hat_{i-1} != y_{i-1}:

$w_T(y_i, y_{i-1})$ += 1

$w_T(y\_hat_i, y\_hat_{i-1})$ -= 1

end if

end for

end for

until stopping condition met

return w_E, w_T

Update emission weights

Update transition weights

Machine learning models

Classification models	Sequence labeling models
Naive Bayes	Hidden Markov model (HMM)
Perceptron	Structured perceptron
Logistic regression (= Maximum entropy classifier)	Maximum entropy Markov model (MEMM) Conditional random field (CRF)

Sequence labeling logistic regression models

Structured perceptron prediction formula:

$$S(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^n} \left(\sum_{i=1}^{n} (w_T(y_i, y_{i-1}) + w_E(y_i) \cdot f_E(x_i)) \right)$$

Apply softmax (exponentiation and normalization):

$$P(\widehat{y_{1...n}} | x_{1...n}) = \max_{y_{1...n} \in Y^n} \left(\frac{\exp\left(\sum_{i=1}^{n+1} (\boldsymbol{w}_T(y_i, y_{i-1}) + \boldsymbol{w}_E(y_i) \cdot \boldsymbol{f}_E(\boldsymbol{x}, i))\right)}{Z(\boldsymbol{w}_E, \boldsymbol{w}_T, \boldsymbol{x})} \right)$$

- where $Z(\boldsymbol{w}_E, \boldsymbol{w}_T, \boldsymbol{x})$ is called the **partition function**:

$$Z(\boldsymbol{w}_E, \boldsymbol{w}_T, \boldsymbol{x}) = \sum_{\boldsymbol{z}_{1...n} \in Y^n} \exp\left(\sum_{i=1}^{n+1} (\boldsymbol{w}_T(z_i, z_{i-1}) + \boldsymbol{w}_E(z_i) \cdot \boldsymbol{f}_E(\boldsymbol{x}, i)) \right)$$

This model is called **conditional random field** (CRF).

Conditional random fields

- What do the features look like?
  - The same as for the structured perceptron
- How are the weight vectors trained/estimated?
  - Stochastic gradient descent
- How is the prediction function implemented?
  - Viterbi algorithm
  - We need to deal with the denominator

CRF prediction

If we are only interested in the argmax:

- we can get rid of the denominator
- we can get rid of the exponentiation
- the prediction function becomes identical to the structured perceptron one.

But sometimes we are interested in the probabilities:

- for training (update)
- for model analysis
- for semi-supervised learning.

CRF training

Assume a training set of m examples.
Each x_j is a sequence of words,
each y_j is a sequence of labels.

We want to find the parameters $\boldsymbol{w}_E$ and $\boldsymbol{w}_T$ (subsumed under $\boldsymbol{w}$) that maximize the likelihood of the data.

$$P(\boldsymbol{w}) = \prod_{j=1}^{m} P(y_j | x_j; \boldsymbol{w})$$

Or equivalently: we want to find the parameters $\boldsymbol{w}_E$ and $\boldsymbol{w}_T$ that minimize the negative log-likelihood loss.

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{j=1}^{m} \log P(y_j | x_j; \boldsymbol{w})$$

Stochastic gradient descent

Instead of taking the sum over all training examples, choose one example at a time and update the weights.

- Initialize $\boldsymbol{w}_E$ and $\boldsymbol{w}_T$
- Randomly choose an example $\boldsymbol{x_j}$ with labels $\boldsymbol{y_j}$
- Compute partial derivatives of the loss to get the update function
  - This update function will contain $P(\boldsymbol{y_j}|\boldsymbol{x_j};\boldsymbol{w})$
- Update weight vectors $\boldsymbol{w}_E$ and $\boldsymbol{w}_T$ accordingly
- Repeat until loss no longer decreases

Parameter estimation in CRFs

How do we compute $P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{w})$?

$$P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{w}) = \max_{y_{1\ldots n}\in Y^n} \frac{\exp\left(\sum_{i=1}^{n+1}\left(\boldsymbol{w}_T(y_i, y_{i-1}) + \boldsymbol{w}_E(y_i)\cdot \boldsymbol{f}_E(\boldsymbol{x}, i)\right)\right)}{Z(\boldsymbol{w},\boldsymbol{x})}$$

$$Z(\boldsymbol{w},\boldsymbol{x}) = \sum_{\boldsymbol{z}_{1\ldots n}\in Y^n} \exp\left(\sum_{i=1}^{n+1}\left(\boldsymbol{w}_T(z_i, z_{i-1}) + \boldsymbol{w}_E(z_i)\cdot \boldsymbol{f}_E(\boldsymbol{x}, i)\right)\right)$$

- We can simplify this formula only a little bit...

Parameter estimation in CRFs

We can take Z out of the max:

$$P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{w},\boldsymbol{x})} \cdot \max_{\boldsymbol{y}_{1\dots n} \in Y^n} \exp\left(\sum_{i=1}^{n+1} \left(\boldsymbol{w}_T(y_i, y_{i-1}) + \boldsymbol{w}_E(y_i) \cdot \boldsymbol{f}_E(\boldsymbol{x}, i)\right)\right)$$

Viterbi

$$Z(\boldsymbol{w},\boldsymbol{x}) = \sum_{\boldsymbol{z}_{1\dots n} \in Y^n} \exp\left(\sum_{i=1}^{n+1} \left(\boldsymbol{w}_T(z_i, z_{i-1}) + \boldsymbol{w}_E(z_i) \cdot \boldsymbol{f}_E(\boldsymbol{x}, i)\right)\right)$$

A variant of Viterbi where the max is replaced by a sum: the **forward-backward algorithm**

- CRF training is computationally expensive!

Maximum Entropy Markov models (MEMMs)

Maximum Entropy Markov Models are a simpler alternative to CRFs:

- Compute the Z normalization term per word
- More efficient to train, but less performant

CRF

$$P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{w}) = \max_{\boldsymbol{y}_{1\ldots n}\in Y^n} \frac{\exp\left(\sum_{i=1}^{n+1}(\boldsymbol{w}_T(y_i, y_{i-1}) + \boldsymbol{w}_E(y_i)\cdot \boldsymbol{f}_E(\boldsymbol{x}, i))\right)}{Z(\boldsymbol{w},\boldsymbol{x})}$$

$$Z(\boldsymbol{w},\boldsymbol{x}) = \sum_{\boldsymbol{z}_{1\ldots n}\in Y^n} \exp\left(\sum_{i=1}^{n+1}\left(\boldsymbol{w}_T(z_i, z_{i-1}) + \boldsymbol{w}_E(z_i)\cdot \boldsymbol{f}_E(\boldsymbol{x}, i)\right)\right)$$

MEMM

$$P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{w}) = \max_{\boldsymbol{y}_{1\ldots n}\in Y^n} \left(\prod_{i=1}^{n+1} \frac{\exp\left(\boldsymbol{w}_T(y_i, y_{i-1}) + \boldsymbol{w}_E(y_i)\cdot \boldsymbol{f}_E(\boldsymbol{x}, i)\right)}{Z(\boldsymbol{w},\boldsymbol{x})}\right)$$

$$Z(\boldsymbol{w},\boldsymbol{x}) = \sum_{\boldsymbol{z}_i\in Y} \exp(\boldsymbol{w}_T(z_i, y_{i-1}) + \boldsymbol{w}_E(z_i)\cdot \boldsymbol{f}_E(\boldsymbol{x}, i))$$

Models

- Simple classification models:
  - (Naïve Bayes)
  - Perceptron with context word features
  - Logistic regression with context word features
- Sequence models with greedy decoding:
  - Hidden Markov model
  - Structured perceptron
  - CRF, MEMM
- Sequence models with exact/Viterbi decoding:
  - Hidden Markov model
  - Structured perceptron
  - CRF, MEMM

Combined approach: use greedy decoding to speed up training, but use Viterbi decoding for best prediction accuracy

Decoding strategies

Matthew Honnibal, 2013:

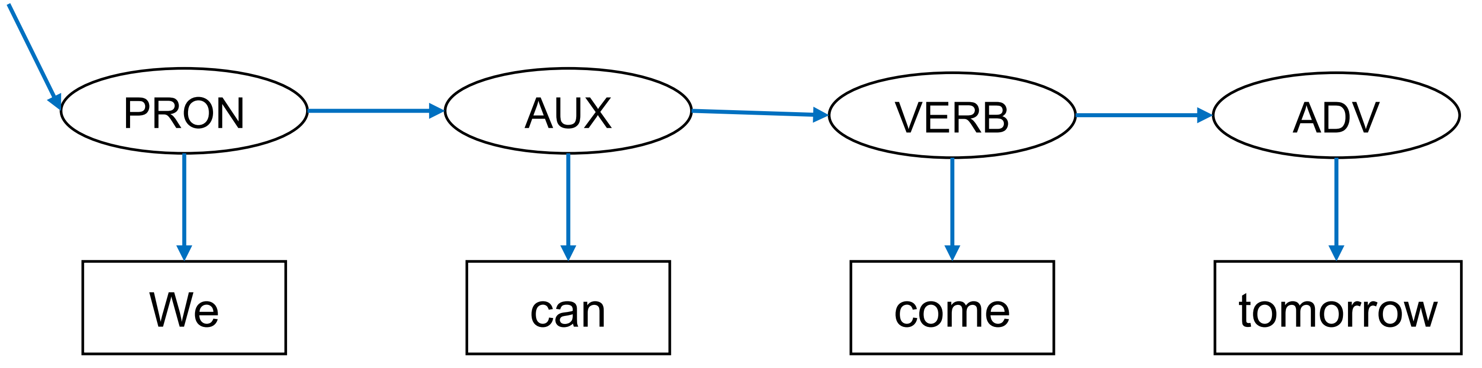
"Unless you really, really can't do without an extra 0.1% of accuracy, you probably shouldn't bother with any kind of search strategy, you should just use a greedy model."

https://explosion.ai/blog/part-of-speech-pos-tagger-in-python

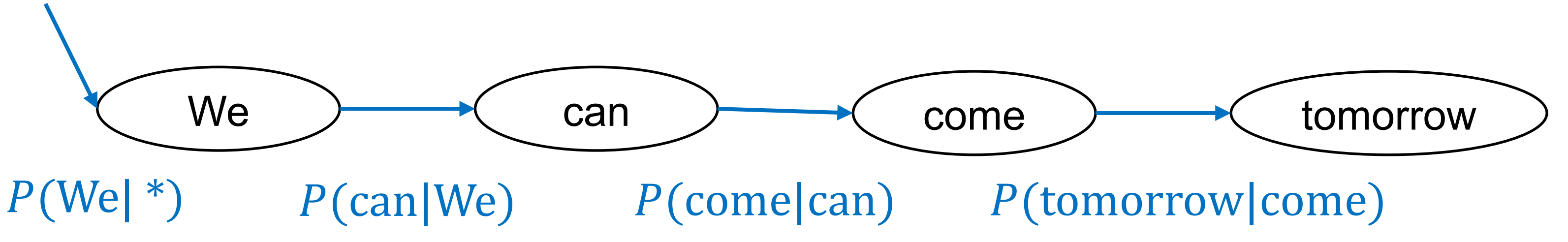
"[Viterbi] search can only help you when you make a mistake. It can prevent that error from throwing off your subsequent decisions, or sometimes your future choices will correct the mistake. [... A greedy] model is so good straight-up that your past predictions are almost always true. So you really need the planets to align for search to matter at all."

Sequence labeling and language modeling

HMM for sequence labeling:



Bigram language model:



Reading

- Jurafsky & Martin, chapter 8
  - HMM, Viterbi algorithm, CRF
- Jurafsky & Martin, chapter 3
  - N-gram language models