Natural Language Processing

The Viterbi algorithm

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

Sequence labelling with global search

- Given an input sequence $\boldsymbol{x}$, we want to find a candidate output sequence $\boldsymbol{y}$ with maximal score.
- For each input sequence, there are exponentially many different output sequences, each one with its own score.

  combinatorial explosion
- However, for a factorised scoring function, the highest-scoring sequence can be found in polynomial time.

Factorised scoring function

candidate output sequence

sequence length

$$\mathrm{score}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_{i=1}^{|\boldsymbol{x}|} \mathrm{score}_1(\boldsymbol{x}, i, y_i; \boldsymbol{\theta}) + \sum_{i=1}^{|\boldsymbol{x}|} \mathrm{score}_2(\boldsymbol{x}, i, y_{i-1}, y_i; \boldsymbol{\theta})$$

input sequence

score for a single label

score for a pair of labels

High-level description

- The Viterbi algorithm takes a factorised scoring function and an input sequence, and returns a highest-scoring label sequence.
- Its central data structure is a matrix with a row for each possible label and a column for each position in the input.

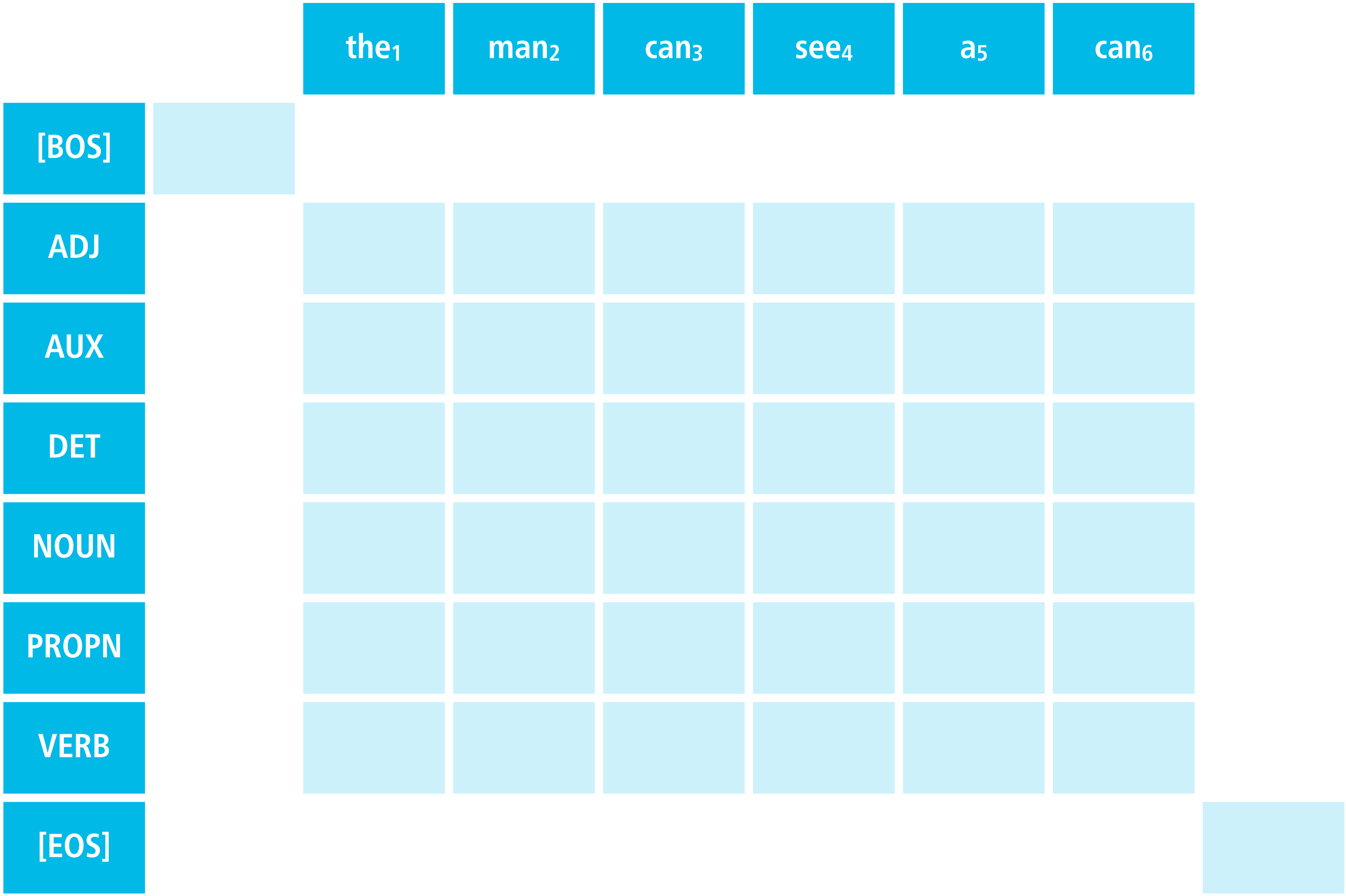
  including [BOS], [EOS]

Example unigram label scores ($score_1$)

	a	can	man	see	the
ADJ	−14.04	−14.04	−14.04	−14.04	−14.04
AUX	−9.42	−3.08	−14.04	−14.04	−14.04
DET	−1.52	−14.31	−14.31	−14.31	−0.70
NOUN	−9.07	−9.54	−6.93	−15.06	−15.06
PROPN	−14.08	−14.08	−14.08	−14.08	−14.08
VERB	−14.65	−14.65	−14.65	−4.81	−14.65

Example bigram label scores ($score_2$)

	ADJ	AUX	DET	NOUN	PROPN	VERB	[EOS]
[BOS]	−3.25	−3.68	−2.30	−2.78	−2.17	−2.81	−9.99
ADJ	−2.96	−5.72	−5.32	−0.61	−3.60	−4.70	−5.52
AUX	−2.22	−2.54	−2.49	−4.41	−5.07	−1.05	−8.73
DET	−1.51	−6.33	−4.59	−0.53	−2.41	−3.97	−9.99
NOUN	−4.49	−2.69	−4.93	−2.09	−4.79	−2.85	−4.12
PROPN	−4.80	−2.82	−5.39	−2.57	−1.35	−2.79	−3.01
VERB	−3.15	−5.12	−1.71	−2.56	−3.75	−4.14	−6.36



the1
man2
can3
see4
a5
can6
[BOS]
ADJ
AUX
DET
NOUN
PROPN
VERB
[EOS]

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$	
[BOS]	−0.00							
ADJ		−17.29	−18.55	−28.99	−32.49	−39.28	−40.87	
AUX		−17.72	−23.37	−16.23	−32.81	−36.63	−34.72	
DET		−3.00	−21.90	−29.70	−33.03	−25.31	−44.21	
NOUN		−17.85	−10.46	−22.09	−35.70	−33.72	−35.38	
PROPN		−16.25	−19.49	−29.34	−35.38	−39.91	−41.80	
VERB		−17.47	−21.62	−27.97	−22.09	−40.88	−43.93	
[EOS]								−39.50

The central invariant

- We ensure that the value in row t, column i is the maximal score that can be obtained when tagging the first i words in the sentence in such a way that word number i is tagged as t.

  column = maximal scores, sub-categorised by last tag

- If the algorithm can establish and maintain this invariant, then we can read off the maximal score for the complete sentence from the last column (end-of-sentence).

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$
[BOS]	0.00						
ADJ							
AUX							
DET							
NOUN							
PROPN							
VERB							
[EOS]							

$$\mathrm{score}_2([\textsc{bos}], \textsc{adj}) + \mathrm{score}_1(\textsc{adj}, \text{the}) = -3.25 + -14.04 = -17.29$$

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$
[BOS]	0.00						
ADJ		−17.29					
AUX							
DET							
NOUN							
PROPN							
VERB							
[EOS]							

$$\text{score}_2([\textsc{bos}], \textsc{adj}) + \text{score}_1(\textsc{adj}, \text{the}) = -3.25 + -14.04 = -17.29$$

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$
[BOS]	0.00						
ADJ		−17.29					
AUX		−17.72					
DET		−3.00					
NOUN							
PROPN							
VERB							
[EOS]							

$$\text{score}_2([\textsc{bos}], \textsc{det}) + \text{score}_1(\textsc{det}, \text{the}) = -2.30 + -0.70 = -3.00$$

		the_1	man_2	can_3	see_4	a_5	can_6
[BOS]	0.00						
ADJ		−17.29	−18.55	−28.99	−32.49		
AUX		−17.72	−23.37	−16.23	−32.81		
DET		−3.00	−21.90	−29.70	−33.03		
NOUN		−17.85	−10.46	−22.09	−35.70		
PROPN		−16.25	−19.49	−29.34	−35.38		
VERB		−17.47	−21.62	−27.97	−22.09		
[EOS]							

$$-16.23 + \text{score}_2(\textsc{aux}, \textsc{verb}) + \text{score}_1(\textsc{verb}, \text{see}) = -16.23 + -1.05 + -4.81 = -22.09$$

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$
[BOS]	0.00						
ADJ		−17.29	−18.55	−28.99	−32.49		
AUX		−17.72	−23.37	−16.23	−32.81		
DET		−3.00	−21.90	−29.70	−33.03		
NOUN		−17.85	−10.46	−22.09	−35.70		
PROPN		−16.25	−19.49	−29.34	−35.38		
VERB		−17.47	−21.62	−27.97	−22.09		
[EOS]							

$$-22.09 + \text{score}_2(\textsc{noun}, \textsc{verb}) + \text{score}_1(\textsc{verb}, \text{see}) = -22.09 + -2.85 + -4.81 = -29.75$$

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$	
[BOS]	0.00							
ADJ		−17.29	−18.55	−28.99	−32.49	−39.28	−40.87	
AUX		−17.72	−23.37	−16.23	−32.81	−36.63	−34.72	
DET		−3.00	−21.90	−29.70	−33.03	−25.31	−44.21	
NOUN		−17.85	−10.46	−22.09	−35.70	−33.72	−35.38	
PROPN		−16.25	−19.49	−29.34	−35.38	−39.91	−41.80	
VERB		−17.47	−21.62	−27.97	−22.09	−40.88	−43.93	
[EOS]								−43.45

$$-34.72 + \text{score}_2(\textsc{aux}, [\textsc{bos}]) = -34.72 + -8.73 = -43.45$$

		the_1	man_2	can_3	see_4	a_5	can_6	
[BOS]	0.00							
ADJ		−17.29	−18.55	−28.99	−32.49	−39.28	−40.87	
AUX		−17.72	−23.37	−16.23	−32.81	−36.63	−34.72	
DET		−3.00	−21.90	−29.70	−33.03	−25.31	−44.21	
NOUN		−17.85	−10.46	−22.09	−35.70	−33.72	−35.38	
PROPN		−16.25	−19.49	−29.34	−35.38	−39.91	−41.80	
VERB		−17.47	−21.62	−27.97	−22.09	−40.88	−43.93	
[EOS]								−39.50

$$-35.38 + \mathrm{score}_2(\textsc{noun}, <\mathrm{eos}>) = -35.38 + -4.12 = -39.50$$

		the$_1$	man$_2$	can$_3$	see$_4$	a$_5$	can$_6$	
[BOS]	0.00							
ADJ		−17.29	−18.55	−28.99	−32.49	−39.28	−40.87	
AUX		−17.72	−23.37	$\boxed{-16.23}$	−32.81	−36.63	−34.72	
DET		$\boxed{-3.00}$	−21.90	−29.70	−33.03	$\boxed{-25.31}$	−44.21	
NOUN		−17.85	$\boxed{-10.46}$	−22.09	−35.70	−33.72	$\boxed{-35.38}$	
PROPN		−16.25	−19.49	−29.34	−35.38	−39.91	−41.80	
VERB		−17.47	−21.62	−27.97	$\boxed{-22.09}$	−40.88	−43.93	
[EOS]								−39.50

Follow the backpointers to read off the tags.

Computational complexity

- Let m, n denote the number of tags and the length of the input sentence, respectively.
- The memory required by the Viterbi algorithm is in $O(mn)$; this corresponds to the size of the matrix.
- The runtime required by the Viterbi algorithm is in $O(m^2n)$: We need to fill $O(mn)$ cells, and each cell requires us to look at $O(m)$ cells in the previous column.

Final comments

- We have presented the Viterbi algorithm as an algorithm for decoding. However, it can also be used for training.

  Essentially, we only need to replace “sum” by “multiply” and “max” by “sum”.

- The Viterbi algorithm is a special case of the max-sum algorithm (or max-product algorithm) for graphical models.