Deep Learning for Natural Language Processing

Factorized Sequence Models

UNIVERSITY OF GOTHENBURG

CHALMERS

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Richard Johansson

richard.johansson@gu.se

Algorithmic approaches

- **Exhaustive search**

  Cast structured prediction as a combinatorial optimisation problem over the set of target representations.

  Viterbi algorithm, Eisner algorithm

- **Greedy search**

  Cast structured prediction as a sequence of classification problems: at each point in time, predict one of several options.

  window-based part-of-speech tagging, arc-standard algorithm

Algorithmic approaches

- **Exhaustive search**

  Cast structured prediction as a combinatorial optimisation problem over the set of target representations.

  Viterbi algorithm, Eisner algorithm

- **Greedy search**

  Cast structured prediction as a sequence of classification problems: at each point in time, predict one of several options.

  window-based part-of-speech tagging, arc-standard algorithm

general approach

- we define a scoring function over the **whole sequence**
- we maximize this function **over all possible sequences**

predicted output input

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y}} \operatorname{score}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$$

candidate output model parameters

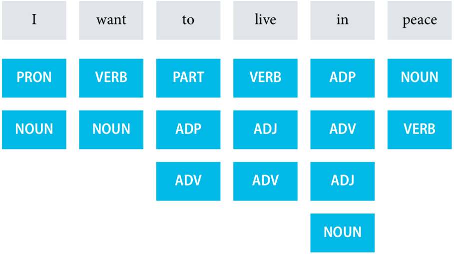
Eisenstein (2019), § 1.2.2

there are many possible sequences. . .

I	want	to	live	in	peace
PRON	VERB	PART	VERB	ADP	NOUN
NOUN	NOUN	ADP	ADJ	ADV	VERB
		ADV	ADV	ADJ	
				NOUN	

making the arg max problem tractable

- the number of possible sequences is typically **exponential** in the length of the input
- we will need to make **assumptions** about the scoring function
- then we can design special algorithms to compute the arg max



first-order factorized scoring function

- we will work with scoring functions where we compute a sum over "parts" or "factors":

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{L} \phi^e(\boldsymbol{x}, y_i) + \sum_{i=1}^{L} \phi^t(\boldsymbol{x}, y_{i-1}, y_i)$$

- the part scoring functions ϕ^e and ϕ^t compute scores for single labels and pairs of adjacent labels, respectively
- it's a **first-order** factorization: we model 1-step interactions

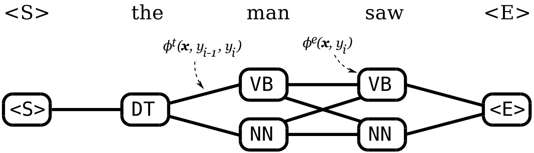
what are these scores?

$$\phi^e(\boldsymbol{x}, y_i) \qquad \phi^t(\boldsymbol{x}, y_{i-1}, y_i)$$

- following HMM terminology, we call them "**emission** scores" (ϕ^e) and "**transition** scores" (ϕ^t)
  - in a HMM, they are log probabilities
  - we will use neural networks to compute them instead
- in the next lecture, we'll discuss how to train them
- for now, let's focus on **decoding**: how to maximize score$(\boldsymbol{x}, \boldsymbol{y})$

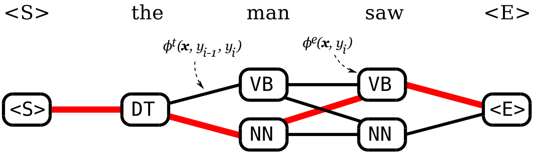
first-order factorization as a graph

- we can construct a **graph** where each node represents a possible label
- we compute node scores with ϕ^e and edge scores with ϕ^t



first-order factorization as a graph

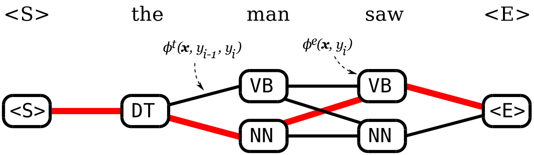
- we can construct a **graph** where each node represents a possible label
- we compute node scores with ϕ^e and edge scores with ϕ^t



- maximizing the scoring function is equivalent to finding the **highest-scoring path**

the Viterbi algorithm

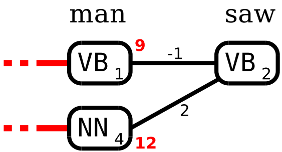
- the **Viterbi algorithm** can be used to find the highest-scoring path in this type of graph
- you may recognize this as a special case of the **max-sum algorithm** for graphical models



- it is a **dynamic programming** algorithm
  - it proceeds from left to right
  - the optimal paths in step i are computed by considering the optimal paths in step $i-1$

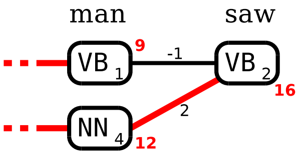
dynamic programming in the Viterbi algorithm

- to compute the best path ending with *saw* as a verb, **consider the best paths for the previous word** and the **transition scores** $\phi^t(\boldsymbol{x}, y_{i-1}, y_i)$



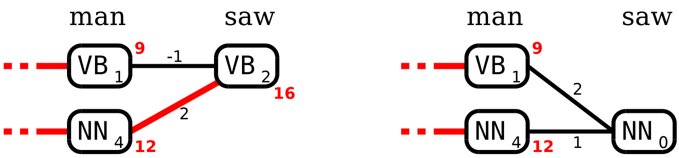
dynamic programming in the Viterbi algorithm

- to compute the best path ending with *saw* as a verb, **consider the best paths for the previous word** and the **transition scores** $\phi^t(\boldsymbol{x}, y_{i-1}, y_i)$



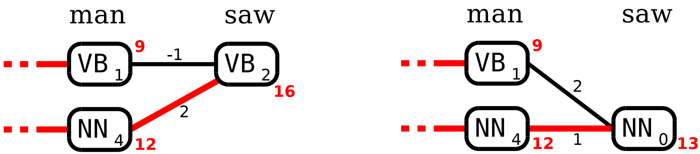
dynamic programming in the Viterbi algorithm

- to compute the best path ending with *saw* as a verb, **consider the best paths for the previous word** and the **transition scores** $\phi^t(\boldsymbol{x}, y_{i-1}, y_i)$



dynamic programming in the Viterbi algorithm

- to compute the best path ending with *saw* as a verb, **consider the best paths for the previous word** and the **transition scores** $\phi^t(\boldsymbol{x}, y_{i-1}, y_i)$



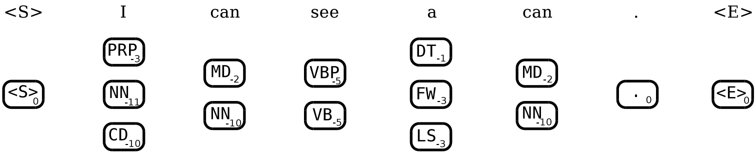
Viterbi example

<S> I can see a can . <E>

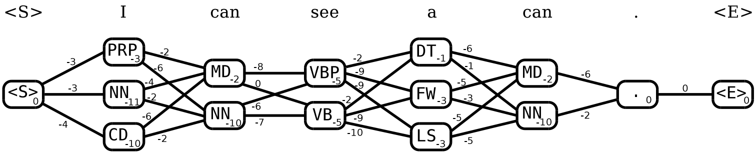
Viterbi example

<S>	I	can	see	a	can	.	<E>
	PRP			DT			
		MD	VBP		MD		
<S>	NN			FW		.	<E>
		NN	VB		NN		
	CD			LS			

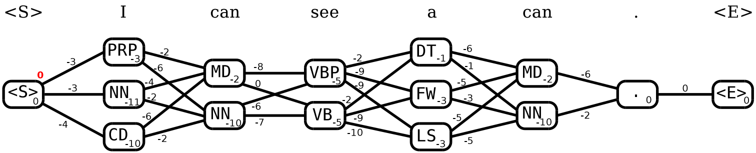
Viterbi example



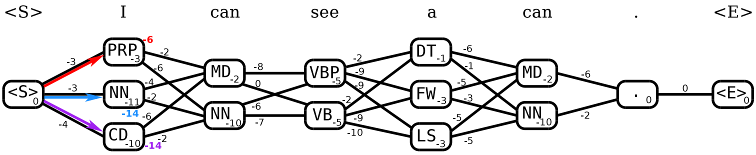
Viterbi example



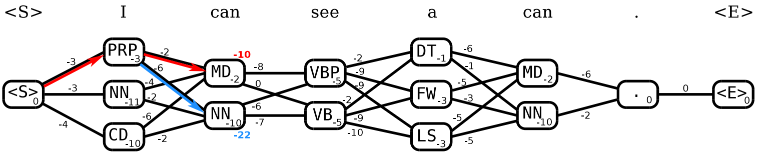
Viterbi example



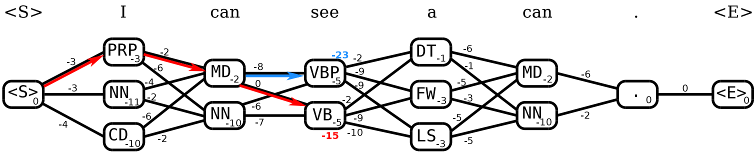
Viterbi example



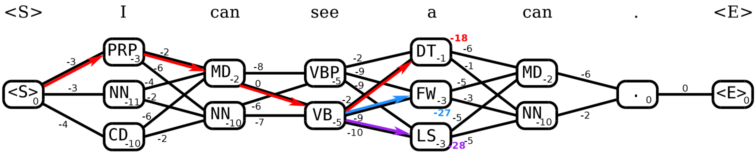
Viterbi example



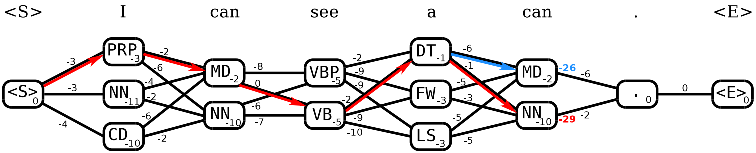
Viterbi example



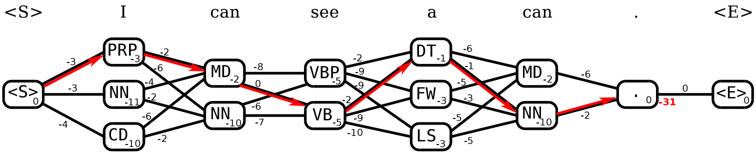
Viterbi example



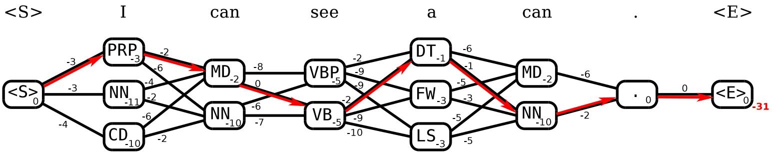
Viterbi example



Viterbi example



Viterbi example



next up: how to train the scoring functions ϕ^e and ϕ^t