Lecture 9

Sequence Tagging

Part 1: Sequence Tagging

Machine Learning for Structured Data
Vlad Niculae · LTL, UvA · https://vene.ro/mlsd

Outline:

Sequence Tagging

Given a sequence of n items $\boldsymbol{x} = (x_1, \ldots, x_n)$, assign to each of them one of K tags:

$$\boldsymbol{y} = (y_1, \ldots, y_n) \quad \text{where each} \quad y_i \in \{1, \ldots, K\}.$$

Sequence Tagging

Given a sequence of n items $\boldsymbol{x} = (x_1, \ldots, x_n)$, assign to each of them one of K tags:

$$\boldsymbol{y} = (y_1, \ldots, y_n) \quad \text{where each} \quad y_i \in \{1, \ldots, K\}.$$

Example 1: Part-of-speech (POS) tagging in NLP

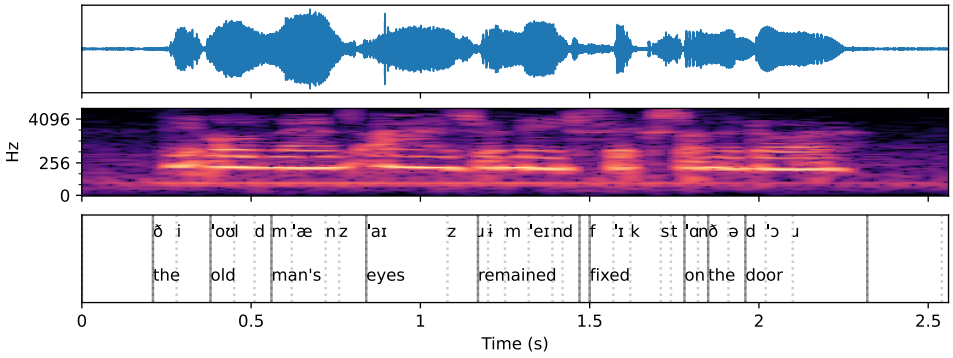
	the	old	man	the	boat
$\boldsymbol{y}_a$	det	adj	noun	det	noun
$\boldsymbol{y}_b$	det	noun	verb	det	noun

Sequence Tagging

Given a sequence of n items $\boldsymbol{x} = (x_1, \ldots, x_n)$, assign to each of them one of K tags:

$$\boldsymbol{y} = (y_1, \ldots, y_n) \quad \text{where each} \quad y_i \in \{1, \ldots, K\}.$$

Example 2: Frame-level phone(me) classification (may be part of speech recognition)

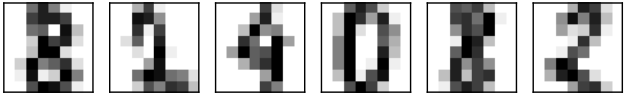


Sequence Tagging

Given a sequence of n items $\boldsymbol{x} = (x_1, \ldots, x_n)$, assign to each of them one of K tags:

$$\boldsymbol{y} = (y_1, \ldots, y_n) \quad \text{where each} \quad y_i \in \{1, \ldots, K\}.$$

Example 3: Optical character recognition



Characterizing The Output Space

Given a sequence of n items $\boldsymbol{x} = (x_1, \ldots, x_n)$, assign to each of them one of K tags:

$$\boldsymbol{y} = (y_1, \ldots, y_n) \quad \text{where each} \quad y_i \in \{1, \ldots, K\}.$$

Input $\boldsymbol{x} = (x_1, \ldots, x_n)$, e.g., a sequence of words.

Output $\boldsymbol{y} = (y_1, \ldots, y_n)$, e.g., a sequence of part-of-speech tags.

For each data point (sentence), $|\boldsymbol{y}| = |\boldsymbol{x}|$; different data points have different lengths.

Characterizing The Output Space

Given a sequence of n items $\boldsymbol{x} = (x_1, \ldots, x_n)$, assign to each of them one of K tags:

$$\boldsymbol{y} = (y_1, \ldots, y_n) \quad \text{where each} \quad y_i \in \{1, \ldots, K\}.$$

Input $\boldsymbol{x} = (x_1, \ldots, x_n)$, e.g., a sequence of words.

Output $\boldsymbol{y} = (y_1, \ldots, y_n)$, e.g., a sequence of part-of-speech tags.

For each data point (sentence), $|\boldsymbol{y}| = |\boldsymbol{x}|$; different data points have different lengths.

For fixed length n, some possible outputs:

- $(1, 1, \ldots, 1, 1) \in \mathcal{Y}$
- $(1, 1, \ldots, 1, 2) \in \mathcal{Y}$
- $(K, K, \ldots, K, K) \in \mathcal{Y}$

How many in terms of n?

Part-Of-Speech Tags

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by, under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Figure 8.1 The 17 parts of speech in the Universal Dependencies tagset (Nivre et al., 2016a). Features can be added to make finer-grained distinctions (with properties like number, case, definiteness, and so on).

Lecture 9

Sequence Tagging

Part 2: Different Scoring Models

Machine Learning for Structured Data
Vlad Niculae · LTL, UvA · https://vene.ro/mlsd

Outline:

Designing A Simple Scorer

Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

$\boldsymbol{A}$ is a matrix of scores,
e.g., computed by a NN encoder.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Designing A Simple Scorer

Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take
$\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

$\boldsymbol{A}$ is a matrix of scores,
e.g., computed by a NN encoder.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

	the	old	man	the	boat
$\boldsymbol{y}_a$	det	adj	noun	det	noun
$\boldsymbol{y}_b$	det	noun	verb	det	noun

Designing A Simple Scorer

Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take
$\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

$\boldsymbol{A}$ is a matrix of scores,
e.g., computed by a NN encoder.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

	the	old	man	the	boat
$\boldsymbol{y}_a$	det	adj	noun	det	noun
$\boldsymbol{y}_b$	det	noun	verb	det	noun

$\text{score}(\boldsymbol{y}_a) =$

Designing A Simple Scorer

Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take
$\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

$\boldsymbol{A}$ is a matrix of scores,
e.g., computed by a NN encoder.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

	the	old	man	the	boat
$\boldsymbol{y}_a$	det	adj	noun	det	noun
$\boldsymbol{y}_b$	det	noun	verb	det	noun

$\text{score}(\boldsymbol{y}_a) = 21$

Designing A Simple Scorer

Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take
$\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

$\boldsymbol{A}$ is a matrix of scores,
e.g., computed by a NN encoder.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

	the	old	man	the	boat
$\boldsymbol{y}_a$	det	adj	noun	det	noun
$\boldsymbol{y}_b$	det	noun	verb	det	noun

$\text{score}(\boldsymbol{y}_a) = 21$

$\text{score}(\boldsymbol{y}_b) =$

Designing A Simple Scorer

Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take
$\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

$\boldsymbol{A}$ is a matrix of scores,
e.g., computed by a NN encoder.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

	the	old	man	the	boat
$\boldsymbol{y}_a$	det	adj	noun	det	noun
$\boldsymbol{y}_b$	det	noun	verb	det	noun

$\text{score}(\boldsymbol{y}_a) = 21$

$\text{score}(\boldsymbol{y}_b) = 17$

Designing A Simple Scorer

A first attempt:
separate classifier for each position.

1. embed and encode $\boldsymbol{x}$, eg, with a CNN.

$$(x_1, \ldots, x_n) \to (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$$

2. For each position j, apply a classification head with K outputs. E.g.,

$$\boldsymbol{a}_j = \boldsymbol{W}^\top \boldsymbol{z}_j + \boldsymbol{b}$$

   Think of $\boldsymbol{A}$ as a matrix with n rows and K columns, where $a_{j,c}$ is the score of assigning tag c at position j.

3. Writing $\boldsymbol{y} = (y_1, \ldots, y_n)$, take $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$.

```
words = [21, 79, 14]  # indices
emb = Embedding(vocab_sz, dim)
clf = Linear(dim, n_tags)

# optionally add RNN, CNN, whatever

Z = emb(words)  # (3 × dim)
A = clf(Z)      # (3 × n_tags)

# computing the score of a given tag sequence:
y = [2, 0, 2]

y_score = sum(A[i, yi]
              for i, yi in enumerate(y))

# or, if you want to be fancy/fast:
y_score = A[torch.arange(len(y)), y].sum()
```

Finding The Best sequence

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\max_{\boldsymbol{y} \in \mathcal{Y}} \text{score}(\boldsymbol{y})$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Finding The Best sequence

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\max_{\boldsymbol{y} \in \mathcal{Y}} \text{score}(\boldsymbol{y})$$

$$= \max_{y_1 \in [K], \ldots, y_n \in [K]} \text{score}([y_1, \ldots, y_n])$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Finding The Best sequence

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\begin{aligned}
&\max_{\boldsymbol{y} \in \mathcal{Y}} \text{score}(\boldsymbol{y}) \\
&= \max_{y_1 \in [K], \ldots, y_n \in [K]} \text{score}\left([y_1, \ldots, y_n]\right) \\
&= \max_{y_1 \in [K], \ldots, y_n \in [K]} \sum_j a_{j,y_j}
\end{aligned}$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Finding The Best sequence

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$
\begin{aligned}
&\max_{\boldsymbol{y} \in \mathcal{Y}} \text{score}(\boldsymbol{y}) \\
&= \max_{y_1 \in [K], \ldots, y_n \in [K]} \text{score}([y_1, \ldots, y_n]) \\
&= \max_{y_1 \in [K], \ldots, y_n \in [K]} \sum_j a_{j,y_j} \\
&= \sum_j \max_{y_j \in [K]} a_{j,y_j}
\end{aligned}
$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Finding The Best sequence

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$
\begin{aligned}
&\max_{\boldsymbol{y} \in \mathcal{Y}} \text{score}(\boldsymbol{y}) \\
&= \max_{y_1 \in [K], \ldots, y_n \in [K]} \text{score}([y_1, \ldots, y_n]) \\
&= \max_{y_1 \in [K], \ldots, y_n \in [K]} \sum_j a_{j,y_j} \\
&= \sum_j \max_{y_j \in [K]} a_{j,y_j}
\end{aligned}
$$

So, $\arg\max_{\boldsymbol{y}} \text{score}(\boldsymbol{y})$ is made up of the tags selected independently at each position.

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Normalizing Constant (log-sum-exp)

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\text{score}(\boldsymbol{y})\right)$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Normalizing Constant (log-sum-exp)

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\text{score}(\boldsymbol{y})\right)$$

$$= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \exp \sum_{j=1}^{n} a_{j,y_j}$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Normalizing Constant (log-sum-exp)

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\text{score}(\boldsymbol{y})\right)$$

$$= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \exp \sum_{j=1}^{n} a_{j,y_j}$$

$$= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \prod_{j=1}^{n} \exp a_{j,y_j}$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Normalizing Constant (log-sum-exp)

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$\log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\text{score}(\boldsymbol{y})\right)$$

$$= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \exp \sum_{j=1}^{n} a_{j,y_j}$$

$$= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \prod_{j=1}^{n} \exp a_{j,y_j}$$

$$= \log \prod_{j=1}^{n} \sum_{y_j=1}^{K} \exp a_{j,y_j}$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Normalizing Constant (log-sum-exp)

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$
\begin{aligned}
&\log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\text{score}(\boldsymbol{y})\right) \\
&= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \exp \sum_{j=1}^{n} a_{j,y_j} \\
&= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \prod_{j=1}^{n} \exp a_{j,y_j} \\
&= \log \prod_{j=1}^{n} \sum_{y_j=1}^{K} \exp a_{j,y_j} \\
&= \sum_{j=1}^{n} \log \sum_{y_j=1}^{K} \exp a_{j,y_j}
\end{aligned}
$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Normalizing Constant (log-sum-exp)

With our $\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$, can we compute:

$$
\begin{aligned}
&\log \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left(\text{score}(\boldsymbol{y})\right) \\
&= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \exp \sum_{j=1}^{n} a_{j,y_j} \\
&= \log \sum_{y_1=1}^{K} \cdots \sum_{y_n=1}^{K} \prod_{j=1}^{n} \exp a_{j,y_j} \\
&= \log \prod_{j=1}^{n} \sum_{y_j=1}^{K} \exp a_{j,y_j} \\
&= \sum_{j=1}^{n} \log \sum_{y_j=1}^{K} \exp a_{j,y_j}
\end{aligned}
$$

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

Probabilistic interpretation: independence

$$
\begin{aligned}
\log \Pr(\boldsymbol{y}) &= \text{score}(\boldsymbol{y}) - \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp \text{score}(\boldsymbol{y}') \\
&= \sum_j \underbrace{\left(a_{j,y_j} - \log \sum_{k \in [K]} \exp a_{j,k} \right)}_{\log \Pr(y_j)}
\end{aligned}
$$

Fully-Local vs. Fully-Global

For sequence tagging, the separable (fully-local) score

$$\text{score}(\boldsymbol{y}) = \sum_j a_{j,y_j}$$

amounts to applying a probabilistic classifier to each of the n positions separately! (any "magic" comes from the feature representation / neural net encoder.)

Can we design a richer $\text{score}(\boldsymbol{y})$ taking into account the sequential structure of $\boldsymbol{y}$?

Fully-Local vs. Fully-Global

Entirely global model: like classification, where *each possible sequence* is a class.

$\boldsymbol{y}$	score($\boldsymbol{y}$)
det det det det det	−1000
det det det det noun	−940
det det det det verb	−800
...	
det noun verb det noun	400
...	
verb verb verb verb verb	−1100

As expressive as possible: score is any function of the sequence.

Fully-Local vs. Fully-Global

Entirely global model: like classification, where *each possible sequence* is a class.

$\boldsymbol{y}$	score($\boldsymbol{y}$)
det det det det det	−1000
det det det det noun	−940
det det det det verb	−800
...	
det noun verb det noun	400
...	
verb verb verb verb verb	−1100

As expressive as possible: score is any function of the sequence.

But completely intractable: $O(K^n)$ time and space.

Fully-Local vs. Fully-Global

Entirely global model: like classification, where *each possible sequence* is a class.

$\boldsymbol{y}$	score($\boldsymbol{y}$)
det det det det det	−1000
det det det det noun	−940
det det det det verb	−800
...	
det noun verb det noun	400
...	
verb verb verb verb verb	−1100

As expressive as possible: score is any function of the sequence.

But completely intractable: $O(K^n)$ time and space.

Structure output prediction is about the space in between these two extremes.

Scoring Transitions Between Tags

A rich scorer that takes into account the sequential nature of $\boldsymbol{y}$ while still allowing efficient computation:

scoring transitions between adjacent tags

$$\text{score}(\boldsymbol{y}) = \sum_{j=1}^{n} a_{j,y_j} + \sum_{j=2}^{n} t_{y_{j-1},y_j}$$

For example, $\text{score}(\texttt{[NOUN, DET, VERB]}) = a_{1,\text{NOUN}} + a_{2,\text{DET}} + a_{3,\text{VERB}} + t_{\text{NOUN,DET}} + t_{\text{DET,VERB}}$

Sequence Modeling With Transition Scores

$$\text{score}(\boldsymbol{y}) = \sum_{j=1}^{n} a_{j,y_j} + \sum_{j=2}^{n} t_{y_{j-1},y_j}$$

The tag scores $\boldsymbol{A} \in \mathbb{R}^{n \times K}$ can be computed as before (e.g., with a convnet.)

The transition scores $\boldsymbol{T} \in \mathbb{R}^{K \times K}$:

- could be a learned parameter. (size does not depend on n)
- could be predicted by the neural net as a function of $\boldsymbol{x}$.

Unlike in the separable case, with transition scores, we no longer get n parallel classifiers: the different tags impact one another. (This makes the model more expressive and more interesting.)

Lecture 9

Sequence Tagging

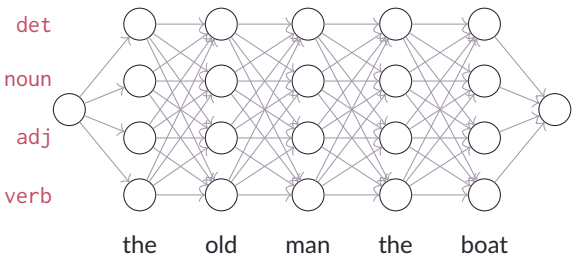
Part 3: Sequence Tagging Algorithms

Machine Learning for Structured Data
Vlad Niculae · LTL, UvA · https://vene.ro/mlsd

Outline:

Sequence Tagging As A DAG

$$\text{score}(\boldsymbol{y}) = \sum_{j=1}^{n} a_{j,y_j} + \sum_{j=2}^{n} t_{y_{j-1},y_j}$$



$G = (V, E, w)$ where:

$$V = \{(j, c) : j \in [n], c \in [K]\} \cup \{s, t\}$$

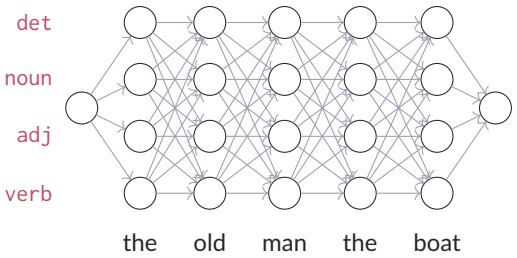
$$E = \{(j-1, c') \to (j, c) : j \in [2, n], c, c' \in [K]\} \cup \{s \to (1, c) : c \in [K]\} \cup \{(n, c) \to t : c \in [K]\}$$

$$w\big((j-1, c') \to (j, c)\big) = a_{j,c} + t_{c',c}$$
$$w(s \to (1, c)) = a_{1,c}$$
$$w((n, c) \to t) = 0$$

$|V| \in \Theta(nK); \quad |E| \in \Theta(nK^2)$

Topological ordering?

Viterbi For Sequence Tagging



General Viterbi (reminder sketch)

initialize $m_1 \leftarrow 0$
for $i = 2, \ldots, n$ **do**
$m_i \leftarrow \max_{j \in P_i} \left(m_j + w(ji)\right)$
$\pi_i \leftarrow \arg\max_{j \in P_i} \left(m_j + w(ji)\right)$
follow backpointers to get best path

Viterbi for sequence tagging

input: Unary scores $\boldsymbol{A}$ ($n \times K$ array)
Transition scores $\boldsymbol{T}$ ($K \times K$ array)

Forward: compute scores recursively
$m_{1c} = a_{1c}$ for all $c \in [K]$
for $j = 2$ to n **do**
for $c = 1$ to K **do**
$m_{j,c} \leftarrow \max_{c' \in [K]} \left(m_{j-1,c'} + a_{j,c} + t_{c',c}\right)$
$\pi_{j,c} \leftarrow \arg\max_{c' \in [K]} \left(m_{j-1,c'} + a_{j,c} + t_{c',c}\right)$
$f^\star = \max_{c' \in [K]} m_{n,c'}$

Backward: follow backpointers
$y_n = \arg\max_{c'} m_n(c')$
for $j = n - 1$ **down to** 1 **do**
$y_j = \pi_{j+1,y_{j+1}}$

output: $f^\star$ and $\boldsymbol{y}^\star = [y_1, \ldots, y_n]$

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M}$ =

	det	noun	adj	verb
the				
old				
man				
the				
boat				

unary and transition scores:

$\boldsymbol{A}$ =

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T}$ =

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old				
man				
the				
boat				

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

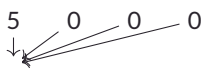
Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old				
man				
the				
boat				



unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1			
man				
the				
boat				

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

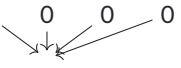
Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1			
man				
the				
boat				



unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1	9		
man				
the				
boat				

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1	9	10	4
man				
the				
boat				

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1	9	10	4
man	8	15	11	12
the	18	13	14	17
boat	18	26	20	17

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1	9	10	4
man	8	15	11	12
the	18	13	14	17
boat	18	26	20	17

To find the best tag sequence $\boldsymbol{y}^\star$, keep track of the path.

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Viterbi For Sequence Tagging: Example

$m_{j,c}$ is stored as a matrix $\boldsymbol{M}$, same shape as $\boldsymbol{A}$.

Apply $m_{1,c} = a_{1,c}$ to get the first row: (copied from $\boldsymbol{A}$)

Then iteratively: $m_{j,c} = \max_{c' \in [K]} m_{j-1,c'} + a_{j,c} + t_{c',c}$

At the end, take the maximum over the last row.

$\boldsymbol{M} =$

	det	noun	adj	verb
the	5	0	0	0
old	1	9	10	4
man	8	15	11	12
the	18	13	14	17
boat	18	26	20	17

To find the best tag sequence $\boldsymbol{y}^\star$, keep track of the path.

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

The Two Main Recurrences Of Sequence Tagging:

(Dynamic programming applied to the sequence tagging DAG)

$$m_{j,c} = \max_{c' \in [K]} \left(m_{j-1,c'} + a_{jc} + t_{c'c} \right),$$
$$q_{j,c} = \log \sum_{c' \in [K]} \exp \left(q_{j-1,c'} + a_{jc} + t_{c'c} \right).$$

The Forward Algorithm

Forward algorithm for sequence tagging

input: Unary scores $\boldsymbol{A}$ ($n \times K$ array)
Transition scores $\boldsymbol{T}$ ($K \times K$ array)

Forward: compute scores recursively

$q_{1,c} = a_{1,c}$ for all $c \in [K]$

for $j = 2$ **to** n **do**

  for $c = 1$ **to** K **do**

    $q_{j,c} = \log \sum_{c' \in [K]} \exp\left(q_{j-1,c'} + a_{j,c} + t_{c',c}\right)$

return $\log Z = \log \sum_{c' \in [K]} \exp\left(q_{n,c'}\right)$

	the	old	man	the	boat	
y_a	det	adj	noun	det	noun	$\text{score}(y_a) = 25$
y_b	det	noun	verb	det	noun	$\text{score}(y_b) = 26$
y_c	noun	noun	noun	noun	noun	$\text{score}(y_c) = 1$

Applying the Forward algorithm yields

$Q =$

	det	noun	adj	verb
the	5.00	0.00	0.00	0.00
old	1.73	9.00	10.00	4.19
man	8.18	15.01	11.05	12.70
the	18.88	13.92	14.37	17.03
boat	18.08	26.88	20.90	18.38

unary and transition scores:

$A =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$T =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

	the	old	man	the	boat	
y_a	det	adj	noun	det	noun	$\text{score}(y_a) = 25$
y_b	det	noun	verb	det	noun	$\text{score}(y_b) = 26$
y_c	noun	noun	noun	noun	noun	$\text{score}(y_c) = 1$

Applying the Forward algorithm yields

$Q =$

	det	noun	adj	verb
the	5.00	0.00	0.00	0.00
old	1.73	9.00	10.00	4.19
man	8.18	15.01	11.05	12.70
the	18.88	13.92	14.37	17.03
boat	18.08	26.88	20.90	18.38

$\log Z \approx 26.885$

unary and transition scores:

$A =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$T =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

	the	old	man	the	boat	
$\boldsymbol{y}_a$	det	adj	noun	det	noun	$\text{score}(\boldsymbol{y}_a) = 25$
$\boldsymbol{y}_b$	det	noun	verb	det	noun	$\text{score}(\boldsymbol{y}_b) = 26$
$\boldsymbol{y}_c$	noun	noun	noun	noun	noun	$\text{score}(\boldsymbol{y}_c) = 1$

Applying the Forward algorithm yields

$\boldsymbol{Q} =$

	det	noun	adj	verb
the	5.00	0.00	0.00	0.00
old	1.73	9.00	10.00	4.19
man	8.18	15.01	11.05	12.70
the	18.88	13.92	14.37	17.03
boat	18.08	26.88	20.90	18.38

$$\log Z \approx 26.885$$

$$\log P(\boldsymbol{y}_a) = \text{score}(\boldsymbol{y}_a) - \log Z = 25 - 26.885 = -1.885$$

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

	the	old	man	the	boat	
$\boldsymbol{y}_a$	det	adj	noun	det	noun	$\text{score}(\boldsymbol{y}_a) = 25$
$\boldsymbol{y}_b$	det	noun	verb	det	noun	$\text{score}(\boldsymbol{y}_b) = 26$
$\boldsymbol{y}_c$	noun	noun	noun	noun	noun	$\text{score}(\boldsymbol{y}_c) = 1$

Applying the Forward algorithm yields

$\boldsymbol{Q} =$

	det	noun	adj	verb
the	5.00	0.00	0.00	0.00
old	1.73	9.00	10.00	4.19
man	8.18	15.01	11.05	12.70
the	18.88	13.92	14.37	17.03
boat	18.08	26.88	20.90	18.38

$$\log Z \approx 26.885$$

$$\log P(\boldsymbol{y}_a) = \text{score}(\boldsymbol{y}_a) - \log Z = 25 - 26.885 = -1.885$$

$$\log P(\boldsymbol{y}_b) = \text{score}(\boldsymbol{y}_b) - \log Z = 26 - 26.885 = -0.885$$

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

	the	old	man	the	boat	
$\boldsymbol{y}_a$	det	adj	noun	det	noun	$\text{score}(\boldsymbol{y}_a) = 25$
$\boldsymbol{y}_b$	det	noun	verb	det	noun	$\text{score}(\boldsymbol{y}_b) = 26$
$\boldsymbol{y}_c$	noun	noun	noun	noun	noun	$\text{score}(\boldsymbol{y}_c) = 1$

Applying the Forward algorithm yields

$\boldsymbol{Q} =$

	det	noun	adj	verb
the	5.00	0.00	0.00	0.00
old	1.73	9.00	10.00	4.19
man	8.18	15.01	11.05	12.70
the	18.88	13.92	14.37	17.03
boat	18.08	26.88	20.90	18.38

$$\log Z \approx 26.885$$

$$\log P(\boldsymbol{y}_a) = \text{score}(\boldsymbol{y}_a) - \log Z = 25 - 26.885 = -1.885$$

$$\log P(\boldsymbol{y}_b) = \text{score}(\boldsymbol{y}_b) - \log Z = 26 - 26.885 = -0.885$$

$$\log P(\boldsymbol{y}_c) = \text{score}(\boldsymbol{y}_c) - \log Z = 1 - 26.885 = -25.885$$

unary and transition scores:

$\boldsymbol{A} =$

	det	noun	adj	verb
the	5	0	0	0
old	0	1	3	0
man	0	3	0	1
the	5	0	0	0
boat	0	5	0	0

$\boldsymbol{T} =$

	det	noun	adj	verb
det	−4	3	2	−1
noun	−3	−2	−1	2
adj	−2	2	1	1
verb	1	−1	0	0

Putting It All Together

At this point, we have all the ingredients needed to train a probabilistic sequence tagger with transition scores!

1. Receiving an input sequence $\boldsymbol{x}$, the model returns unary and transition scores $\boldsymbol{A}$ and $\boldsymbol{T}$.
2. If we're at test time:
   run Viterbi to get predicted sequence; compute accuracies etc.
3. If training time:
   run Forward algorithm to compute the training objective
   $-\log P(\boldsymbol{y} \mid \boldsymbol{x}) = -\operatorname{score}(\boldsymbol{y}) + \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp \operatorname{score}(\boldsymbol{y}')$.

Evaluation

(Jurafsky and Martin, 2024, Sec. 4.7)

Well, what would we do in the unstructured case?

Notation: Iverson Bracket

$$[\![p]\!] = \begin{cases} 1, & p \text{ is true,} \\ 0, & \text{otherwise.} \end{cases}$$

- Accuracy:
  What fraction of test cases are correctly classified?

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} [\![y^{(i)} = \widehat{y}^{(i)}]\!]$$

Structured evaluation: POS tagging

For sequential data, accuracy already becomes more complicated:

sequence-level?

$$\text{Acc}_{\text{seq}} = \frac{\sum_{i=1}^{N} [\![\boldsymbol{y}^{(i)} = \hat{\boldsymbol{y}}^{(i)}]\!]}{N}$$

or (micro-averaged) tag accuracy? (writing $n^{(i)} = |\boldsymbol{y}^{(i)}|$):

$$\text{Acc}_{\text{tag}} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{n^{(i)}} [\![y_j^{(i)} = \hat{y}_j^{(i)}]\!]}{\sum_{i=1}^{N} n^{(i)}}$$

(could also imagine a macro-averaged version, but it's not meaningful here)

Example:

true:	PRO	VERB	NUM	NOUN	ADV
pred:	PRO	VERB	NUM	NOUN	PRO
words:	there	are	70	children	there

true:	INTJ
pred:	X
words:	eeeeek

$$\text{Acc}_{\text{seq}} = \frac{0}{2} = 0$$

$$\text{Acc}_{\text{tag}} = \frac{4}{6} = 0.667$$

Historical Notes And References

- This probabilistic model is often known as a Linear-Chain Conditional Random Field and due to Lafferty et al. (2001). (Historically, Linear-Chain CRFs didn't use neural net scorers, but the math doesn't change. I prefer to teach it in a more general way.)
- On POS tagging: (Jurafsky and Martin, 2024) Speech and Language Processing [link]
- Structured prediction: (Daumé III, 2012), A Course In Machine Learning, ch. 17 [link]

References I

- Daumé III, Hal (2012). *A Course in Machine Learning*.
- Jurafsky, Daniel and James H. Martin (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released August 20, 2024.
- Lafferty, John, Andrew McCallum, Fernando Pereira, et al. (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *Icml*. Vol. 1. 2. Williamstown, MA, p. 3.