Named entity recognition with HMMs

Carlos Ramisch (carlos.ramisch@univ-amu.fr)

With the help of Benoit Favre & Alexis Nasr

Prédiction Structurée pour le Traitement Automatique des Langues
Master IAAA
Aix Marseille Université

Outline

Hidden Markov models (HMMs)

Named entity recognition (NER)

BIO encoding

Hidden Markov models $\neq$ deep learning

- Classical probabilistic model for sequence tagging
  - $\rightarrow$ Popular in the 80's and 90's for many tasks
- Parameters are probabilities, not arbitrary real numbers
  - $\rightarrow$ Generative story of dataset generation
- Learning relies on standard statistical estimation
  - $\rightarrow$ No loss function, no back-propagation

Hidden Markov models $\in$ machine learning

- Training: learn parameters from training data
  - $\rightarrow$ Direct probability estimation
- Inference: predict sequences on dev/test data
  - $\rightarrow$ Efficient inference: dynamic programming (Viterbi)
- Simple model to introduce the Viterbi algorithm
  - $\rightarrow$ Many applications in more recent/complex models
  - $\rightarrow$ Including deep learning models like RNNs!

Remember n-gram language models

- Probabilistic model for word sequence $w_1^n = w_1 w_2 \dots w_n$
  - $\longrightarrow$ Starting point: probability chain rule

$$P(w_1^n) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1 w_2) \times \dots \times P(w_n|w_1 w_2 \dots w_{n-1})$$

- Probability of next word w_i given previous words $w_1 w_2 \dots w_{i-1}$
  - $\longrightarrow$ Estimation becomes intractable as $i \rightarrow n$

Markov assumption

- Markov assumption: current word depends on limited history
  - → Word w_i depends only on its $k-1$ previous words $w_{i-k+1} \dots w_i$
  - → The value of k is called the model's order
- In particular, for $k=2$, we have a bigram model

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

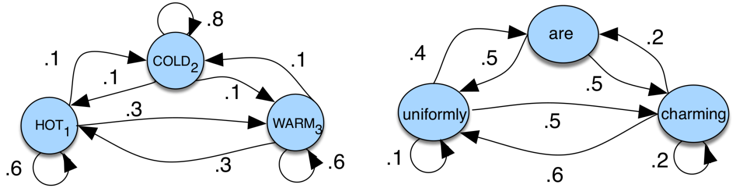
$$P(w_1^n) = P(w_1) \times \prod_{i=2}^{n} P(w_i | w_{i-1})$$

- N-gram language models can be used to:
  - → Rank sentences according to probability $P(w_1^n)$
  - → Generate sentences by sampling within $P(w_1^n)$

Graphical representation

- Markov chain: transitions between states
  - $\rightarrow$ Bigram language model = Markov chain over words
- Outgoing arrows form proper probability distribution

$$\sum_{i=1}^{|V_w|} P(w_i|w_j) \quad \forall w_j \in V_w$$



Source: Jurafsky & Martin

Markov chain: formalisation

Given a sequence of random variables $S_1 S_2 \dots S_n$ over time $k = 1 \dots n$

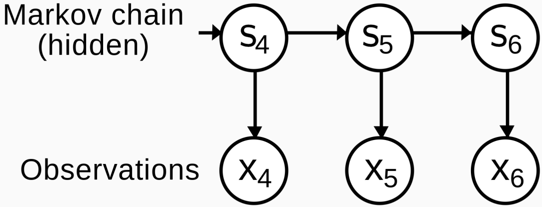
Markov chain

- $V_s = \{s_1 \dots s_N\}$: set of possible states for S_t
  $\rightarrow$ E.g. vocabulary of words V_w
- $T(s_i, s_j)$: transition matrix encoding $P(S_t = s_j | S_{t-1} = s_i)$
  $\rightarrow$ E.g. bigram probabilities $P(w_i | w_{i-1})$ estimated from corpus
- $\pi(s_i)$: initial state probabilities $P(S_1 = s_i)$
  $\rightarrow$ E.g. probability of starting sentence with w_i

Hidden Markov models (HMMs)

General idea: states are hidden, but they emit observable symbols

- Markov chain: states = observations = words (n-gram LM)
- Hidden Markov model: states $\neq$ observations
  - $\rightarrow$ State sequence $s_1 s_2 \ldots s_n$ is hidden
  - $\rightarrow$ At each time step k, state s_t emits observable x_t



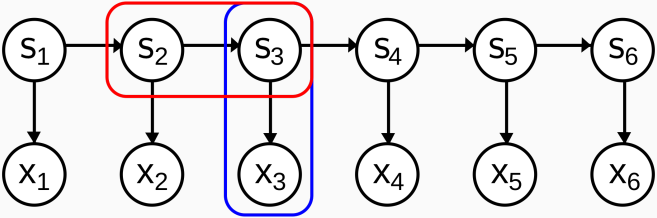
HMM independence assumptions

- Markov assumption: state S_k depends only on state S_{k-1}

$$P(S_k|S_1S_2\dots S_{k-1}) = P(S_k|S_{k-1})$$

- Output independence: observed X_k depends only on current state S_k

$$P(X_k|S_1S_2\dots S_nX_1X_2\dots X_n) = P(X_k|S_k)$$



HMM components

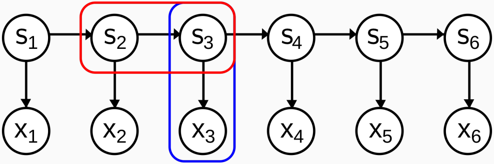
- $V_s = \{s_1 \dots s_N\}$ set of possible (hidden) states
- $V_x = \{x_1 \dots x_M\}$ set of possible (observable) symbols
- Transition matrix, dimension $N \times N$

$$T(s_i, s_j) = P(S_k = s_j | S_{k-1} = s_i)$$

- Emission matrix, dimension $N \times M$

$$E(s_i, x_j) = P(X_k = x_j | S_k = s_i)$$

- Initial state probabilities $\pi(s_i) = P(S_1 = s_i)$



Transition matrix T

$$
\begin{array}{cc|c|c|c|c|c|c|}
 & & \multicolumn{6}{c}{S_k} \\
 & & s_1 & s_2 & \dots & s_j & \dots & s_N \\
\hline
 & s_1 & & & & & & \\
\hline
 & s_2 & & & & & & \\
\hline
S_{k-1} & \vdots & & & & & & \\
\hline
 & s_i & & & & P(s_j|s_i) & & \\
\hline
 & \vdots & & & & & & \\
\hline
 & s_N & & & & & & \\
\hline
\end{array}
$$

Transition matrix $T(s_i, s_j)$

- **Attention!** Previous state s_i (row), next state s_j (column)
- Rows must sum to 1

Emission matrix E and initial vector π

$$
\begin{array}{cc|cccccc|}
 & & & & & X_k & & \\
 & & x_1 & x_2 & \dots & x_j & \dots & x_M \\
\hline
 & s_1 & & & & & & \\
 & s_2 & & & & & & \\
S_k & \vdots & & & & & & \\
 & s_i & & & & P(x_j|s_i) & & \\
 & \vdots & & & & & & \\
 & s_N & & & & & & \\
\hline
\end{array}
$$

Emission matrix $E(s_i, x_j)$

$$
\begin{array}{|cccccc|}
 & & & S_1 & & \\
s_1 & s_2 & \dots & s_i & \dots & s_N \\
\hline
 & & & P(S_1 = s_i) & & \\
\hline
\end{array}
$$

Initial vector $\pi(s_i)$

- Rows must sum to 1

Weather example (1)

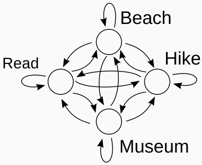
- Guess someone's next activity (hidden) given weather (observable)
  - $\rightarrow$ Weather $X_k \in \{\text{rainy, cloudy, sunny, windy}\}$
  - $\rightarrow$ Activity $S_k \in \{\text{beach, hike, museum, read}\}$

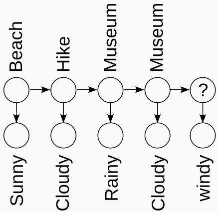
		S_{k-1}				X_k			
		beach	hike	mus.	read	rainy	cloudy	sunny	windy
S_k	beach	0.2	0.3	0.4	0.1	0.1	0.3	0.2	0.4
	hike	0.1	0.2	0.3	0.4	0.3	0.2	0.4	0.1
	mus.	0.4	0.1	0.2	0.3	0.2	0.4	0.1	0.3
	read	0.3	0.4	0.1	0.2	0.4	0.1	0.3	0.2
		Transition matrix T^T				Emission matrix E			

Weather example (2)

		S_{k-1}				X_k			
		beach	hike	mus.	read	rainy	cloudy	sunny	windy
S_k	beach	0.2	0.3	0.4	0.1	0.1	0.3	0.2	0.4
	hike	0.1	0.2	0.3	0.4	0.3	0.2	0.4	0.1
	mus.	0.4	0.1	0.2	0.3	0.2	0.4	0.1	0.3
	read	0.3	0.4	0.1	0.2	0.4	0.1	0.3	0.2

Transition matrix — Emission matrix

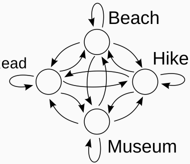


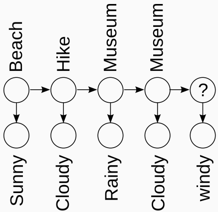


What is the most probable next activity?

Weather example (2)

	S_{k-1}				X_k			
	beach	hike	mus.	read	rainy	cloudy	sunny	windy
S_k beach	0.2	0.3	0.4	0.1	0.1	0.3	0.2	0.4
hike	0.1	0.2	0.3	0.4	0.3	0.2	0.4	0.1
mus.	0.4	0.1	0.2	0.3	0.2	0.4	0.1	0.3
read	0.3	0.4	0.1	0.2	0.4	0.1	0.3	0.2
	Transition matrix				Emission matrix			





What is the most probable next activity?

$$P(X_k = \text{windy} | S_k = \text{beach}) \times P(S_k = \text{beach} | S_{k-1} = \text{mus.}) \qquad = \mathbf{0.4} \times \mathbf{0.4}$$

$$P(X_k = \text{windy} | S_k = \text{mus.}) \times P(S_k = \text{mus.} | S_{k-1} = \text{mus.}) \qquad = 0.3 \times 0.2$$

$$P(X_k = \text{windy} | S_k = \text{hike}) \times P(S_k = \text{hike} | S_{k-1} = \text{mus.}) \qquad = 0.1 \times 0.3$$

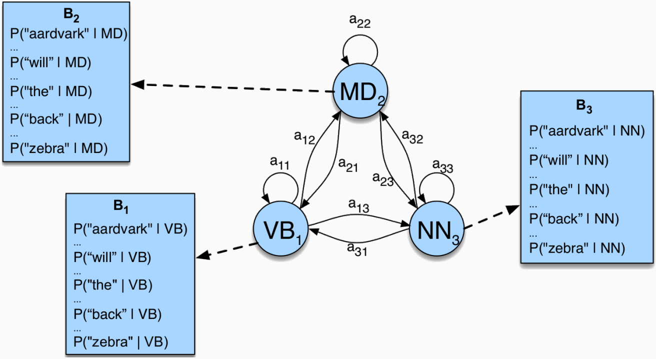
$$P(X_k = \text{windy} | S_k = \text{read}) \times P(S_k = \text{read} | S_{k-1} = \text{mus.}) \qquad = 0.2 \times 0.1$$

$\Longrightarrow$ Most probable next state $S_k = \text{beach}$

HMMs for words and tags

- States $S_1 S_2 \ldots S_n$ are (POS) tags $t_i \in V_t$
- Observables $X_1 X_2 \ldots X_n$ are words $w_j \in V_w$
  - $\rightarrow$ Words "generated" stochastically according to (hidden) POS tags
  - $\rightarrow$ At each timestep $k = 1 \ldots n$, go to next tag and generate a word
- Transition matrix $T(t_i, t_j) = P(S_k = t_j | S_{k-1} = t_i)$
- Emission matrix $E(t_i, w_j) = P(X_k = w_j | S_k = t_i)$
- Initial probabilities $\pi(t_i) = P(S_1 = t_i)$

Example: HMM POS tagger



Source: Jurafsky & Martin

Hidden Markov models: formalisation

Given random variables $S_1 S_2 \dots S_n$ and $X_1 X_2 \dots X_n$ over time $k = 1 \dots n$

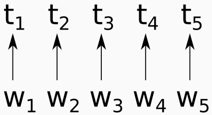
Hidden Markov model

- $V_s = \{s_1 \dots s_N\}$: set of possible states for S_t
  - $\rightarrow$ E.g. vocabulary of tags V_t
- $V_x = \{x_1 \dots x_M\}$: set of possible symbols for X_t
  - $\rightarrow$ E.g. vocabulary of words V_w
- $T(s_j, s_i)$: transition matrix encoding $P(S_k = s_i | S_{k-1} = s_j)$
  - $\rightarrow$ E.g. tag bigram probabilities $P(t_i | t_{i-1})$
- $E(s_i, x_j)$: emission matrix encoding $P(X_k = x_j | S_k = s_i)$
  - $\rightarrow$ E.g. word-tag probabilities $P(w_j | t_i)$
- $\pi(s_i)$: initial state probabilities $P(S_1 = s_i)$
  - $\rightarrow$ E.g. probability of starting with tag t_i

HMM questions

Given a word sequence $w_1^n = w_1 \dots w_n$ and an HMM

1. What is the probability $P(w_1^n)$ according to the HMM?
2. How can we estimate the HMM parameters E, T and π?
   → From an annotated corpus with aligned words and tags
3. What is the most probable (hidden) tag sequence $t_1^n = t_1 \dots t_n$?
   → Sequence tagging task (e.g. POS tagging)



Question 2: parameter estimation

- HMM parameters T, E and π are probabilities
- Idea: estimate from (training) corpus statistics
  - $\rightarrow$ Each word is annotated with its corresponding tag
- Probability $P(t_i) \approx$ proportion of occurrences $\frac{c(t_i)}{L}$
  - $\rightarrow$ $L =$ corpus length (total number of words)
  - $\rightarrow$ $c(t_i)$ number of occurrences of t_i

Example:

$$E(t_i, w_j) = P(X_k = w_j | S_k = t_i) = \frac{P(X_k = w_j, S_k = t_i)}{P(S_k = t_i)} = \frac{\frac{c(w_j, t_i)}{L}}{\frac{c(t_i)}{L}} = \frac{c(w_j, t_i)}{c(t_i)}$$

HMM training: from counts to probabilities

Step 1: obtain the following counts from annotated corpus

- $c(w_j, t_i)$ number of times w_j was tagged t_i
- $c(t_i, t_j)$ number of adjacent tag pairs $\langle t_i, t_j \rangle$
- $c(t_i)$ number of occurrences of tag t_i
- $c(\texttt{<s>}, t_i)$ number of sentence-initial occurrences of t_i
- S total number of sentences, i.e. $S = \sum_{t_i} c(\texttt{<s>}, t_i)$

Step 2: estimate parameters by maximum likelihood

$$E(t_i, w_j) = \frac{c(w_j, t_i)}{c(t_i)} \qquad T(t_i, t_j) = \frac{c(t_i, t_j)}{c(t_i)} \qquad \pi(t_i) = \frac{c(\texttt{<s>}, t_i)}{S}$$

HMM training: exercise

```
they/PRON fish/VERB
they/PRON can/AUX eat/VERB
they/PRON eat/VERB can/NOUN fish/NOUN
```

Given the annotated corpus, estimate $T(t_i, t_j)$, $E(t_i, w_j)$, and $\pi(t_i)$

HMM training: exercise

```
they/PRON fish/VERB
they/PRON can/AUX eat/VERB
they/PRON eat/VERB can/NOUN fish/NOUN
```

Given the annotated corpus, estimate $T(t_i, t_j)$, $E(t_i, w_j)$, and $\pi(t_i)$

	PRON	VERB	AUX	NOUN
PRON	0	2/3	1/3	0
VERB	0	0	0	1/1
AUX	0	1/1	0	0
NOUN	0	0	0	1/1

$T(t_i, t_j)$

	they	can	eat	fish
PRON	3/3	0	0	0
VERB	0	0	2/3	1/3
AUX	0	1/1	0	0
NOUN	0	1/2	0	1/2

$E(t_i, w_j)$

	PRON	VERB	AUX	NOUN
$\pi(t_j)$	3/3	0	0	0

Question 3: sequence tagging

- Given a word sequence $w_1^n = w_1 \dots w_n$ and a trained HMM
- What is the most probable tag sequence $\hat{t}_1^n = \hat{t}_1 \dots \hat{t}_n$, that is

$$\hat{t}_1^n = \operatorname{argmax} P(t_1^n | w_1^n)$$

- Among all possible tag sequences $t_1^n \in (V_t)^n$?

Sequence tagging in HMMs i

- First, apply Bayes rule

$$\hat{t}_1^n = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(t_1^n | w_1^n) = \underset{t_1 \dots t_n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

- argmax ranges over all possible tag sequences t_1^n, but w_1^n is constant

$$\hat{t}_1^n = \underset{t_1 \dots t_n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{\cancel{P(w_1^n)}} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n)$$

Sequence tagging in HMMs ii

- Use HMMs independence assumptions to simplify inference

$$\hat{t}_1^n = \operatorname{argmax} P(w_1^n|t_1^n)P(t_1^n) = \operatorname{argmax} \prod_{k=1}^{n} P(w_k|t_k)P(t_k|t_{k-1})$$

- These correspond to emission and transition probabilities of HMM![1]

$$\hat{t}_1^n = \operatorname{argmax} \prod_{k=1}^{n} E(t_k, w_k) \times T(t_{k-1}, t_k)$$

[1] Assuming $T(t_{k-1}, t_k) = \pi(t_1)$ for $k = 1$

Note on emission probabilities

- **Warning**: states emit observable symbols, not the opposite!
  - $\rightarrow$ States $=$ tags, observables $=$ words, so tags $\rightarrow$ words
- $E(t_i, w_j) = P(X_k = w_j | S_k = t_i) \neq \cancel{P(S_k = t_i | X_k = w_j)}$
  - $\rightarrow$ ~~Intepretation: probability to assign tag t_i to word w_j~~
  - $\rightarrow$ Intepretation: probability that word w_j was generated by tag t_i
- Bayes' rule's fault: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Naive HMM inference

1. Generate next possible tag sequence t_1^n
2. Calculate its probability

$$P(t_1^n|w_1^n) = \pi(t_1) \times E(t_1, w_1) \times \prod_{k=2}^{n} E(t_k, w_k) \times T(t_{k-1}, t_k)$$

3. If higher than previous tag sequence, record it (argmax)
4. Go to step 1 until all search space is explored

Naive inference: example

	PRON	VERB	AUX	NOUN
PRON	0	2/3	1/3	0
VERB	0	0	0	1
AUX	0	1	0	0
NOUN	0	0	0	1

$T(t_i, t_j)$

	they	can	eat	fish
PRON	1	0	0	0
VERB	0	0	2/3	1/3
AUX	0	1	0	0
NOUN	0	1/2	0	1/2

$E(t_i, w_j)$

	PRON	VERB	AUX	NOUN
$\pi(t_j)$	1	0	0	0

Most probable tag sequence for *"they can fish"* given HMM above?

Naive inference: example

	PRON	VERB	AUX	NOUN
PRON	0	2/3	1/3	0
VERB	0	0	0	1
AUX	0	1	0	0
NOUN	0	0	0	1

$T(t_i, t_j)$

	they	can	eat	fish
PRON	1	0	0	0
VERB	0	0	2/3	1/3
AUX	0	1	0	0
NOUN	0	1/2	0	1/2

$E(t_i, w_j)$

	PRON	VERB	AUX	NOUN
$\pi(t_j)$	1	0	0	0

Most probable tag sequence for *"they can fish"* given HMM above?

$\hat{t}_1^3 = \texttt{PRON AUX VERB}$ with probability:

$$\pi(\texttt{PRON}) \times E(\texttt{PRON,they}) \times T(\texttt{PRON,AUX}) \times E(\texttt{AUX,can}) \times T(\texttt{AUX,VERB}) \times E(\texttt{VERB,fish})$$
$$= 1 \times 1 \times \frac{1}{3} \times 1 \times 1 \times \frac{1}{3} = \frac{1}{9}$$

All other possible tag sequences have have zero probability

Tractability and redundancy

Naive HMM inference

1. Generate next possible tag sequence t_1^n
2. Calculate its probability
   $P(t_1^n|w_1^n) = \pi(t_1)E(t_1, w_1)\prod_{k=2}^{n} E(t_k, w_k) \times T(t_{k-1}, t_k)$
3. If higher than previous tag sequence, record it (argmax)
4. Go to step 1 until all search space is explored

- How many different tag sequences in the search space?

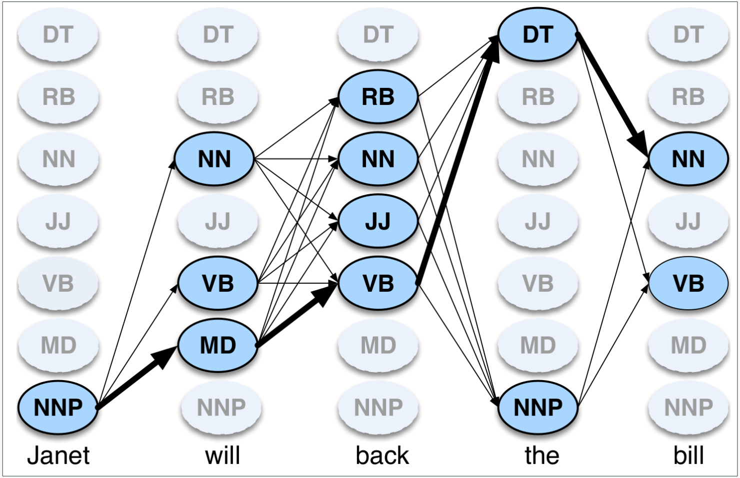
Tractability and redundancy

Naive HMM inference

1. Generate next possible tag sequence t_1^n
2. Calculate its probability
   $P(t_1^n|w_1^n) = \pi(t_1)E(t_1, w_1)\prod_{k=2}^{n} E(t_k, w_k) \times T(t_{k-1}, t_k)$
3. If higher than previous tag sequence, record it (argmax)
4. Go to step 1 until all search space is explored

- How many different tag sequences in the search space?
  $\rightarrow$ N^n sequences if tag vocabulary V_t is of size $|V_t| = N$
- For instance, if $N = 10$ tags and n=10 words:
  $\rightarrow$ $N^n = 10^{10} =$ 10 billion!
- Moreover, many calculations are redundant
  $\rightarrow$ Many candidate t_1^n share common subsequences

Lattice representation



Source: Jurafsky & Martin

Dynamic programming: principle

- Solve a problem by breaking it down into similar sub-problems
- Sub-problems are decomposed until we arrive at trivial sub-problems
- Initial problem's solution = combination sub-problems solutions
  - $\rightarrow$ Store sub-problems solutions to avoid redundant calculations
- Very common in NLP and machine learning
  - $\rightarrow$ Viterbi, Cocke Youger Kasami, gradient backpropagation, edit distance...

Dynamic programming example: Fibonacci

```
def fiboRec(n):
  if n <= 1:
    return n
  else:
    return fiboRec(n - 1) + fiboRec(n - 2)

def fiboDP(n):
  fib = [0] * (n + 1)
  fib[1] = 1
  for i in range(2, n + 1):
    fib[i] = fib[i - 1] + fib[i - 2]
  return fib[n]
```

Example: for $n = 20$, `fiboRec` $\rightarrow$ 21891 sums, `fiboDP` $\rightarrow$ 21 sums

Dynamic programming for HMM inference

- Idea: get $P(w_1 \dots w_k | t_1 \dots t_k)$ at incremental time steps $k = 1 \dots n$
  - $\rightarrow$ Keep track of most probable tag sequence up to now
- Initialisation: $P(w_1 | t_1)$ depends on $\pi(t_1)$ and $E(t_1, w_1)$
- Recurrence: $P(w_1 \dots w_{k+1} | t_1 \dots t_{k+1})$ depends only on:
  - $\rightarrow$ Previous step $P(w_1 \dots w_k | t_1 \dots t_k)$
  - $\rightarrow$ Emission probability $E(t_{k+1}, w_{k+1})$
  - $\rightarrow$ Transition probability $T(t_k, t_{k+1})$

Viterbi matrix $\delta(t_j, w_k)$

- Row indices: all possible tags $t_1 \dots t_N$
- Column indices: sentence positions $w_1 \dots w_n$

	w_1	w_2	$\dots$	w_k	$\dots$	w_n
t_1						
t_2						
$\vdots$						
t_j				$\max P(w_1 \dots w_k \vert S_k = t_j)$		
$\vdots$						
t_N						

Viterbi matrix $\delta(t_j, w_k)$

Recurrent definition

Initialisation:

$$\delta(t_j, w_1) = \pi(t_j) \times E(t_j, w_1) \qquad 1 \leq j \leq N$$

Recursion:

$$\delta(t_j, w_k) = \max_{1 \leq i \leq N} [\delta(t_i, w_{k-1}) \times T(t_i, t_j)] \times E(w_k, t_j) \qquad \begin{array}{l} 2 \leq k \leq n \\ 1 \leq j \leq N \end{array}$$

Viterbi: exercise

	PRON	VERB	AUX	NOUN
PRON	0	2/3	1/3	0
VERB	0	0	0	1
AUX	0	1	0	0
NOUN	0	0	0	1

$T(t_i, t_j)$

	they	can	eat	fish
PRON	1	0	0	0
VERB	0	0	2/3	1/3
AUX	0	1	0	0
NOUN	0	1/2	0	1/2

$E(t_i, w_j)$

	PRON	VERB	AUX	NOUN
$\pi(t_j)$	1	0	0	0

Build the Viterbi matrix $\delta(t_j, w_k)$ for *"they can fish"* given HMM above?

Viterbi: exercise

	PRON	VERB	AUX	NOUN
PRON	0	2/3	1/3	0
VERB	0	0	0	1
AUX	0	1	0	0
NOUN	0	0	0	1

$T(t_i, t_j)$

	they	can	eat	fish
PRON	1	0	0	0
VERB	0	0	2/3	1/3
AUX	0	1	0	0
NOUN	0	1/2	0	1/2

$E(t_i, w_j)$

	PRON	VERB	AUX	NOUN
$\pi(t_j)$	1	0	0	0

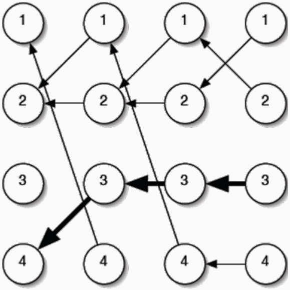
Build the Viterbi matrix $\delta(t_j, w_k)$ for *"they can fish"* given HMM above?

	they	can	fish
PRON	1	0	0
VERB	0	0	1/9
AUX	0	1/3	0
NOUN	0	0	0

Back pointers matrix $\psi(t_j, w_k)$

- Matrix $\delta(t_j, w_k)$ provides highest probability $(\max)$
- We want the tag sequence with highest probability (argmax)
- Solution: matrix $\psi(t_j, w_k)$ records argmax
  - $\rightarrow$ Pointer to a row in previous column

$$\psi(t_j, w_k) = \operatorname*{argmax}_{1 \leq i \leq N} [\delta(t_i, w_{k-1}) \times T(t_i, t_j)] \times E(t_j, w_k) \qquad \begin{array}{c} 2 \leq k \leq n \\ 1 \leq j \leq N \end{array}$$



Backtracking

- Most probable sequence $=$ backtrack on $\psi(t_j, w_k)$
  - $\rightarrow$ Get last tag $\hat{t_n}$ from last column of $\delta(t_j, w_n)$
  - $\rightarrow$ Get previous $\hat{t_{n-1}}$ from $\psi(\hat{t_n}, w_{n-1})$
  - $\rightarrow$ Stop when obtaining $\hat{t_1}$ from $\psi(\hat{t_2}, w_2)$

$$\hat{t}_n = \underset{1 \leq i \leq N}{\operatorname{argmax}} \, \delta(t_i, w_n)$$

$$\hat{t}_{k-1} = \psi(\hat{t}_k, w_{k-1}) \qquad n > k \geq 2$$

Viterbi pseudo-code

```
def viterbi(sent, E, T, pi):
  for j in 1..N: # initialize pi * E values in column 1
    delta[j,1] = E[j, sent[1]] * pi[j]
  for k in 2..n :
    for j in 1..N: # For all possible tags of word k
      for i in 1..N : # For all possible previous best seq
        prob = delta[i, k-1] * T[i, j] * E[j, sent[k]]
        if prob > delta[j,k] :
          delta[j,k] = prob
          psi[j,k] = i
  # Backtrack
  result = [argmax(delta[j, n])]
  for k in n-1..2:
    result = psi[result[0], k] + result
  return result
```

Viterbi: exercise

Create $\delta(t_j, w_k), psi(t_j, w_k)$ and infer $\hat{t_1^n}$ for $w_1^n below$

$w_1^n =$ *Janet will back the bill*

Emission matrix E

	Janet	**will**	**back**	**the**	**bill**
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Initial π and transition matrix T

	NNP	**MD**	**VB**	**JJ**	**NN**	**RB**	**DT**
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Add-1 smoothing for OOV pairs

- Some elements in E and T may be zero (OOV pairs)
  - $\rightarrow$ Catastrophic: zeroes whole Viterbi path!
- Solution: Add-one smoothing
  - $\rightarrow$ Add one to numerator (zeroes becomes ones)
  - $\rightarrow$ Adjust denominator to obtain proper probabilities

$$E(t_i, w_j) = \frac{c(w_j, t_i)}{c(t_i)} = \frac{c(w_j, t_i)}{\sum_{w_j \in V_w} c(w_j, t_i)}$$

$$\begin{aligned} E_{\text{add-1}}(t_i, w_j) &= \frac{c(w_j, t_i) + 1}{\sum_{w_j \in V_w} [c(w_j, t_i) + 1]} \\ &= \frac{c(w_j, t_i) + 1}{\sum_{w_j \in V_w} c(w_j, t_i) + \sum_{w_j \in V_w} 1} \\ &= \frac{c(w_j, t_i) + 1}{c(t_i) + |V_w|} \end{aligned}$$

Laplace smoothing

- Instead of add 1, add α (typically $0.001 \leq \alpha < 1$)
  - $\rightarrow$ Possibly different values for each parameter
- Careful: different denominator vocabularies V_w and V_t!

$$E_{\text{Laplace}}(t_i, w_j) = \frac{c(w_j, t_i) + \alpha}{c(t_i) + |V_w|\alpha}$$

$$T_{\text{Laplace}}(t_i, t_j) = \frac{c(t_i, t_j) + \alpha}{c(t_i) + |V_t|\alpha}$$

$$\pi_{\text{Laplace}}(t_i) = \frac{c(\texttt{<s>}, t_i) + \alpha}{S + |V_t|\alpha}$$

Smaller and smaller

- We multiply probabilities to get $\delta(t_j, w_n)$
  $\rightarrow$ Probabilities are values between 0 and 1
- This product gets very very small fast

$$\text{As } n \rightarrow \infty, \prod_{i=1}^{n} p_i \rightarrow 0 \text{ when } 0 \leq p_i < 1$$

- Computers' float point precision is limited

Smaller and smaller

- We multiply probabilities to get $\delta(t_j, w_n)$
  $\rightarrow$ Probabilities are values between 0 and 1
- This product gets very very small fast

$$\text{As } n \rightarrow \infty, \prod_{i=1}^{n} p_i \rightarrow 0 \text{ when } 0 \leq p_i < 1$$

- Computers' float point precision is limited

Underflow!

Preventing underflow

- Remember $\hat{t_1^n} = \mathrm{argmax}_{t_1 \dots t_n} P(t_1^n | w_1^n)$
- Trick: transform probability $P(t_1^n | w_1^n)$ into log-probability
  - $\rightarrow$ log p ranges from $-\infty$ to 0 for $0 \leq p \leq 1$
  - $\rightarrow$ Products become sums: $\log(a \times b) = \log(a) + \log(b)$
  - $\rightarrow$ argmax invariant to log (monotonic)
- Negative log-probability to deal with positive numbers
  - $\rightarrow$ argmin instead of argmax
- Bonus: sums are (slightly) faster than products

Preventing underflow: gory details

$$\begin{aligned}
\hat{t_1^n} &= \operatorname*{argmax}_{t_1 \ldots t_n} P(t_1^n | w_1^n) \\
&= \operatorname*{argmin}_{t_1 \ldots t_n} - \log P(t_1^n | w_1^n) \\
&= \operatorname*{argmin}_{t_1 \ldots t_n} - \log \left(\pi(t_1) \times E(t_1, w_1) \times \prod_{k=2}^{n} E(t_k, w_k) \times T(t_{k-1}, t_k) \right) \\
&= \operatorname*{argmin}_{t_1 \ldots t_n} - \log \pi(t_1) - \log E(t_1, w_1) - \left(\sum_{k=2}^{n} \log E(t_k, w_k) + \log T(t_{k-1}, t_k) \right)
\end{aligned}$$

All together

Parameter estimation:

$$E(t_i, w_j) = \log\left[c(t_i) + |V_w|\alpha\right] - \log\left[c(w_j, t_i) + \alpha\right]$$
$$T(t_i, t_j) = \log\left[c(t_i) + |V_t|\alpha\right] - \log\left[c(t_i, t_j) + \alpha\right]$$
$$\pi(t_i) = \log(S + |V_t|\alpha) - \log\left[c(\texttt{<s>}, t_i) + \alpha\right]$$

Viterbi:

$$\delta(t_j, w_1) = \pi(t_j) + E(t_j, w_1) \qquad 1 \leq j \leq N$$
$$\delta(t_j, w_k) = \min_{1 \leq i \leq N} \left[\delta(t_i, w_{k-1}) + T(t_i, t_j)\right] + E(w_k, t_j) \qquad 2 \leq k \leq n$$
$$\psi(t_j, w_k) = \operatorname*{argmin}_{1 \leq i \leq N} \left[\delta(t_i, w_{k-1}) + T(t_i, t_j)\right] + E(w_k, t_j) \qquad 1 \leq j \leq N$$

Viterbi for RNNs

- Idea: replace emission matrix E by RNN + linear
- $P(t_i|w_1^n) = softmax(\text{RNN}(w_1^n) \times U)$, with $U \in \mathbb{R}^{d_h \times |V_t|}$

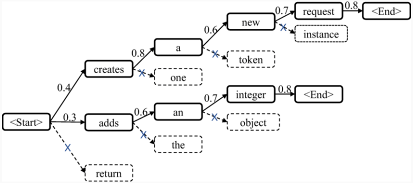
  $\rightarrow$ Probability distribution conditioned on whole w_1^n
- RNN model learned independently using cross-entropy
- Transition and initial probabilities estimated separately
- Advantages: powerful neural models, no invalid tag sequence

Viterbi complexity

- For N different tags, and a n-words sentence
- Time complexity: $O(N^2 \times n)$
- Space complexity: $O(N \times n)$
- Much better than naive search: $O(N^n)$
- May still be prohibitive for large N

Beam search

- Reduce the cost of Viterbi by using beam search
  - $\rightarrow$ Sub-optimal solution
- At a given time k, fill in column $\delta(t_j, w_k)$ for all states t_j
- Keep only the K most probable elements of column w_k
  - $\rightarrow$ Discard other rows in that column
- At step $k+1$, only surviving states are considered to get $\delta(t_j, w_{k+1})$
- Time complexity: $O(N \times K \times n)$, where K is a parameter



Advanced topics

- K-best solutions
  - $\rightarrow$ Tansform Viterbi $N \times n$ matrix into $N \times n \times K$ tensor
  - $\rightarrow$ Predict all K best solutions (not only *the* best)
- Training HMMs from non-annotated corpus
  - $\rightarrow$ Baum-Welch algorithm (expectation maximisation)
- Conditional random fields (CRFs)
  - $\rightarrow$ Discriminant generalisation of HMMs
- Deep CRFs
  - $\rightarrow$ Combine CRF with neural networks for sequence prediction

Outline

Hidden Markov models (HMMs)

Named entity recognition (NER)

BIO encoding

Segmentation task

Task definition

- **Input** ($\mathbf{x}$): a sequence of n inputs (words) w_1 to w_n
- **Output** ($\mathbf{y}$): one category tag t_i per (continuous) segment

$$\mathrm{w}_1 \; \overbrace{\mathrm{w}_2 \; \mathrm{w}_3}^{\mathrm{t}_3} \; \mathrm{w}_4 \; \overbrace{\mathrm{w}_5}^{\mathrm{t}_8}$$

- Some segments may have no tag
- Segments can be composed of 1 or more words
  - $\longrightarrow$ Segments are continuous (no gaps)

Named entity recognition (NER)

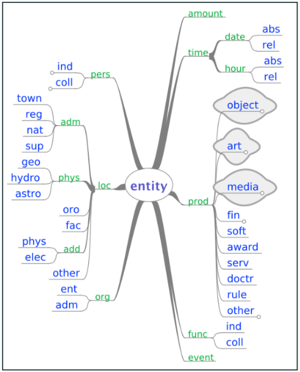
- Identify contiguous segments denoting entities
  - Person, location, organisation, date...
- NER is an important first step in downstream tasks
  - → Information extraction, sentiment analysis, event extraction, ...

> Citing high fuel prices, [$_{ORG}$ **United Airlines**] said [$_{TIME}$ **Friday**] it has increased fares by [$_{MONEY}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{ORG}$ **American Airlines**], a unit of [$_{ORG}$ **AMR Corp.**], immediately matched the move, spokesman [$_{PER}$ **Tim Wagner**] said. [$_{ORG}$ **United**], a unit of [$_{ORG}$ **UAL Corp.**], said the increase took effect [$_{TIME}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{LOC}$ **Chicago**] to [$_{LOC}$ **Dallas**] and [$_{LOC}$ **Denver**] to [$_{LOC}$ **San Francisco**].

Source: Jurafsky & Martin

Named entity tagsets: examples

Type	Tag	Sample Categories
People	PER	people, characters
Organization	ORG	companies, sports teams
Location	LOC	regions, mountains, seas
Geo-Political Entity	GPE	countries, states



CoNLL 2003 (English, German)

- Simple: 4 NE categories
- Flat: no nesting

Quaero (French)

- Complex: many categories
- Hierarchical: nested NEs

Source: Jurafsky & Martin; Dupont et al. 2017

The Sequoia / PARSEME-FR tagset

- `PERS` – person, e.g. *Jul*, *Amy Winehouse*, *Jacques Chirac*
- `LOC` – location, e.g. *La Belle de Mai*, *Chili*, *Ajaccio*
- `ORG` – organisation, e.g. *Olympique de Marseille*, *ONU*, *AP-HM*
- `PROD` – production (cultural, artistic. . .), e.g. *Wikipédia*, *Le Canard Enchaîné*, *Accords de Schengen*, *Le Petit Prince*
- `EVE` – named event, e.g. *JO de Paris 2024*, *guerre d'Indochine*, *Coupe d'Europe*

Source: `https://gitlab.lis-lab.fr/PARSEME-FR/PARSEME-FR-public/`

The Sequoia / PARSEME-FR format

FORM	parseme:ne
Le	1:PROD
Petit	1
Prince	1
de	*
Saint-Exupéry	2:PERS
est	*
entré	*
à	*
l'	*
École	3:ORG
Jules-Romains	3

Entities

- *Le Petit Prince* → `PROD`
- *Saint-Exupéry* → `PERS`
- *École Jules-Romains* → `ORG`

Conventions

- * → not entity token
- NEs: `INDEX` or `INDEX:CAT`
- Same index = same NE
- First token → `:CAT`

Named entity annotation: exercise

Dans son intervention, M. Soyer a fait l'historique de l'école maternelle, qui existe à Vignot depuis 1972. Elle occupait depuis cette date, un bâtiment préfabriqué dans le parc Verneau.

Au mois de mars dernier, les enfants de CE2 de Mme Philippe avaient montré leur travail réalisé d'après "Les contes du chat perché" de Marcel Aimé.

La section judo de l'OFP a organisé ce week-end le premier tour de la compétition appelée "les Petits Tigres", dans la salle du COSEC.

Named entity annotation: exercise

Dans son intervention, M. Soyer $_{(PERS)}$ a fait l'historique de l'école maternelle, qui existe à Vignot $_{(LOC)}$ depuis 1972. Elle occupait depuis cette date, un bâtiment préfabriqué dans le parc Verneau$_{(PERS)}$ $_{(LOC)}$.

Au mois de mars dernier, les enfants de CE2 de Mme Philippe $_{(PERS)}$ avaient montré leur travail réalisé d'après "Les contes du chat perché" $_{(PROD)}$ de Marcel Aimé $_{(PERS)}$.

La section judo de l'OFP $_{(ORG)}$ a organisé ce week-end le premier tour de la compétition appelée "les Petits Tigres" $_{(EVE)}$, dans la salle du COSEC $_{(ORG)}$.

NE processing: challenges

- Span definition: *[Le Havre]* but *les [États Unis]*
  - → Arbitrary but consistent annotation guidelines
- Variability: *États Unis*, *États Unis d'Amérique*, *USA*, *US*, *Amérique*
  - → Character-level models, variant dictionaries
- Nesting: $[_{ORG}$ *Comité d'intérêt du* $[_{LOC}$ *quartier* $[_{PERS}$ *Saint Marcel*$]]]$
  - → Rarely addressed in practice
  - → Requires more complex models, e.g. NE list generation by trained LM
- Ambiguity: *Saint Laurent* can be a river (LOC), a brand (PROD), or a person (PERS)
  - → Statistical models, enough training data, external knowledge bases

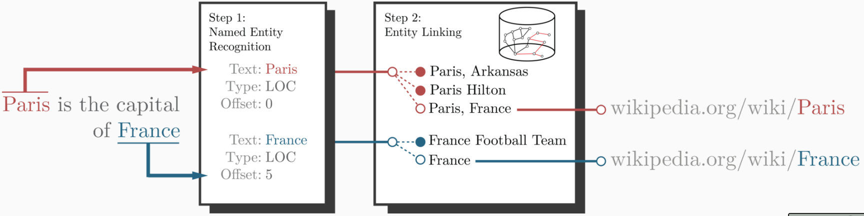
Named entity linking

Category ambiguity (metonymy)

- *Blair arrived in [LOC Washington] for his last state visit.*
- *[ORG Washington] went up 2 games to 1 in the four-game series.*
- *[PER Washington] was born into slavery on the farm of James Burroughs.*

Referential ambiguity (homonymy)

- *[LOC Paris] is the county seat for the northern district of Logan County.*
- *[LOC Paris] is represented in the Texas Senate by senator Bryan Hughes.*



Outline

Hidden Markov models (HMMs)

Named entity recognition (NER)

BIO encoding

Segmentation as tagging

General idea: transform the segmentation problem into a tagging problem

→ Tags represent segment boundaries (+ categories)

→ Use known models for sequence tagging, e.g. RNNs

BIO tagging scheme

- BIO encoding – proposed by Ramshaw & Marcus (1995)
  - B: begin a segment (optional)
  - I: inside a segment
  - O: outside any segment
- Append the NE category to B and I tags

Words	IO Label	BIO Label
Jane	I-PER	B-PER
Villanueva	I-PER	I-PER
of	O	O
United	I-ORG	B-ORG
Airlines	I-ORG	I-ORG
Holding	I-ORG	I-ORG
discussed	O	O
the	O	O
Chicago	I-LOC	B-LOC
route	O	O
.	O	O

- *Note*: some BIO sequences are *invalid*, e.g. `O I` or `B-LOC I-ORG`

BIO tagging: exercise

FORM	BIO
Le	?
Petit	?
Prince	?
de	?
Saint-Exupéry	?
est	?
entré	?
à	?
l'	?
École	?
Jules-Romains	?

BIO tagging: exercise

FORM	BIO
Le	B-PROD
Petit	I-PROD
Prince	I-PROD
de	O
Saint-Exupéry	B-PERS
est	O
entré	O
à	O
l'	O
École	B-ORG
Jules-Romains	I-ORG

BIO tagging: quiz

- How many different tags in IO with $m = 5$ NE categories?

BIO tagging: quiz

- How many different tags in IO with $m = 5$ NE categories?

  $\rightarrow$ $|V_t| = m + 1 = 6$ different tags

- How many different tags in BIO with $m = 5$ NE categories?

BIO tagging: quiz

- How many different tags in IO with $m = 5$ NE categories?
  $\rightarrow$ $|V_t| = m + 1 = 6$ different tags
- How many different tags in BIO with $m = 5$ NE categories?
  $\rightarrow$ $|V_t| = 2m + 1 = 11$ different tags
- What are the advantages of BIO over IO?

BIO tagging: quiz

- How many different tags in IO with $m = 5$ NE categories?

  $\rightarrow$ $|V_t| = m + 1 = 6$ different tags
- How many different tags in BIO with $m = 5$ NE categories?

  $\rightarrow$ $|V_t| = 2m + 1 = 11$ different tags
- What are the advantages of BIO over IO?

  $\rightarrow$ Distinguish consecutive entities, e.g. *trajet* [$_B$*Paris]* [$_B$*Marseille]*
- What are the advantages of IO over BIO?

BIO tagging: quiz

- How many different tags in IO with $m = 5$ NE categories?

  $\rightarrow$ $|V_t| = m + 1 = 6$ different tags

- How many different tags in BIO with $m = 5$ NE categories?

  $\rightarrow$ $|V_t| = 2m + 1 = 11$ different tags

- What are the advantages of BIO over IO?

  $\rightarrow$ Distinguish consecutive entities, e.g. *trajet* [$_B$*Paris*] [$_B$*Marseille*]

- What are the advantages of IO over BIO?

  $\rightarrow$ Less tags, consistent e.g. [$_B$ Aéroport] [$_I$ de] [$_I$ Roissy] vs. [$_B$ Roissy]

Convert PARSEME-FR format → BIO

```
from conllulib import CoNLLUReader
test="""# global.columns = ID FORM parseme:ne
1     Le     1:PROD
2     Petit     1
3     Prince     1
4     de     *
5     Saint-Exupéry     2:PERS
6     est     *
7     entré     *
8     à     *
9     l'     *
10     École     3:ORG
11     Jules-Romains     3"""
for sent in CoNLLUReader.readConlluStr(test):
  print(CoNLLUReader.to_bio(sent))
```

Convert PARSEME-FR format → BIO

```
from conllulib import CoNLLUReader
test="""# global.columns = ID FORM parseme:ne
1     Le     1:PROD
2     Petit     1
3     Prince     1
4     de     *
5     Saint-Exupéry     2:PERS
6     est     *
7     entré     *
8     à     *
9     l'     *
10     École     3:ORG
11     Jules-Romains     3"""
for sent in CoNLLUReader.readConlluStr(test):
  print(CoNLLUReader.to_bio(sent))
#['B-PROD', 'I-PROD', 'I-PROD', 'O', 'B-PERS', 'O', 'O',
# 'O', 'O', 'B-ORG', 'I-ORG']
```

Convert BIO → PARSEME-FR format

```
s1 = ["B-PERS", "I-PERS", "I-PERS", "O", "B-LOC", "I-LOC"]
s2 = ["I-PERS", "B-PERS", "I-PERS", "O", "I-LOC"]
print(CoNLLUReader.from_bio(s1, bio_style='bio'))
```

Convert BIO → PARSEME-FR format

```
s1 = ["B-PERS", "I-PERS", "I-PERS", "O", "B-LOC", "I-LOC"]
s2 = ["I-PERS", "B-PERS", "I-PERS", "O", "I-LOC"]
print(CoNLLUReader.from_bio(s1, bio_style='bio'))
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s1, bio_style='io'))
```

Convert BIO → PARSEME-FR format

```
s1 = ["B-PERS", "I-PERS", "I-PERS", "O", "B-LOC", "I-LOC"]
s2 = ["I-PERS", "B-PERS", "I-PERS", "O", "I-LOC"]
print(CoNLLUReader.from_bio(s1, bio_style='bio'))
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s1, bio_style='io'))
# WARNING: Got B tag in 'io' bio_style: interpreted as I
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s2, bio_style='bio'))
```

Convert BIO → PARSEME-FR format

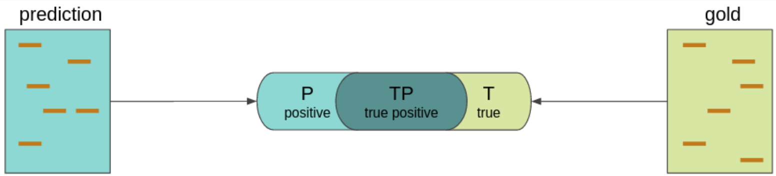
```
s1 = ["B-PERS", "I-PERS", "I-PERS", "O", "B-LOC", "I-LOC"]
s2 = ["I-PERS", "B-PERS", "I-PERS", "O", "I-LOC"]
print(CoNLLUReader.from_bio(s1, bio_style='bio'))
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s1, bio_style='io'))
# WARNING: Got B tag in 'io' bio_style: interpreted as I
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s2, bio_style='bio'))
# WARNING: Invalid I-initial tag I-PERS converted to B
# WARNING: Invalid I-initial tag I-LOC converted to B
# ['1:PERS', '2:PERS', '2', '*', '3:LOC']
print(CoNLLUReader.from_bio(s2, bio_style='io'))
```

Convert BIO → PARSEME-FR format

```
s1 = ["B-PERS", "I-PERS", "I-PERS", "O", "B-LOC", "I-LOC"]
s2 = ["I-PERS", "B-PERS", "I-PERS", "O", "I-LOC"]
print(CoNLLUReader.from_bio(s1, bio_style='bio'))
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s1, bio_style='io'))
# WARNING: Got B tag in 'io' bio_style: interpreted as I
# ['1:PERS', '1', '1', '*', '2:LOC', '2']
print(CoNLLUReader.from_bio(s2, bio_style='bio'))
# WARNING: Invalid I-initial tag I-PERS converted to B
# WARNING: Invalid I-initial tag I-LOC converted to B
# ['1:PERS', '2:PERS', '2', '*', '3:LOC']
print(CoNLLUReader.from_bio(s2, bio_style='io'))
# WARNING: Got B tag in 'io' bio_style: interpreted as I
# ['1:PERS', '1', '1', '*', '2:LOC']
```

NER evaluation

- Accuracy of BIO tags is not a good metric!
  - `O` is much more frequent than `B` and `I`...
  - ...but predicting `O` is less important than `B` and `I`
- Entities' precision, recall and F-score:
  - Exact match: full entity span
  - Partial/fuzzy match: tokens belonging to entities
  - Category: considered or ignored



NER evaluation: exercise

Input →	***Mary***	***Allen***	*buys*	***Big***	***Pasta***	***Inc.***	*from*	*former*	*owner*	***Jim***	***Smith***
Gold →	B-PERS	I-PERS	O	B-ORG	I-ORG	I-ORG	O	O	O	B-PERS	I-PERS
Pred. →	B-PERS	I-PERS	O	B-ORG	I-ORG	O	O	B-ORG	I-ORG	O	B-PERS

- Precision and recall (exact match) ignoring NE categories?
- Precision and recall (fuzzy match) ignoring NE categories?
- Precision and recall (exact match) per NE category?
- BIO tag accuracy?

NER evaluation: exercise

Input →	*Mary*	*Allen*	*buys*	*Big*	*Pasta*	*Inc.*	*from*	*former*	*owner*	*Jim*	*Smith*
Gold →	B-PERS	I-PERS	O	B-ORG	I-ORG	I-ORG	O	O	O	B-PERS	I-PERS
Pred. →	B-PERS	I-PERS	O	B-ORG	I-ORG	O	O	B-ORG	I-ORG	O	B-PERS

- Precision and recall (exact match) ignoring NE categories?
  → P=4, T=3, TP=1, Prec=1/4, Rec=1/3, F=2/7≈0.28
- Precision and recall (fuzzy match) ignoring NE categories?
- Precision and recall (exact match) per NE category?
- BIO tag accuracy?

NER evaluation: exercise

Input →	*Mary*	*Allen*	*buys*	*Big*	*Pasta*	*Inc.*	*from*	*former*	*owner*	*Jim*	*Smith*
Gold →	B-PERS	I-PERS	O	B-ORG	I-ORG	I-ORG	O	O	O	B-PERS	I-PERS
Pred. →	B-PERS	I-PERS	O	B-ORG	I-ORG	O	O	B-ORG	I-ORG	O	B-PERS

- Precision and recall (exact match) ignoring NE categories?
  - → P=4, T=3, TP=1, Prec=1/4, Rec=1/3, F=2/7≈0.28
- Precision and recall (fuzzy match) ignoring NE categories?
  - → P=7, T=7, TP=5, Prec=5/7, Rec=5/7, F=5/7≈0.71
- Precision and recall (exact match) per NE category?
- BIO tag accuracy?

NER evaluation: exercise

Input →	*Mary*	*Allen*	*buys*	*Big*	*Pasta*	*Inc.*	*from*	*former*	*owner*	*Jim*	*Smith*
Gold →	B-PERS	I-PERS	O	B-ORG	I-ORG	I-ORG	O	O	O	B-PERS	I-PERS
Pred. →	B-PERS	I-PERS	O	B-ORG	I-ORG	O	O	B-ORG	I-ORG	O	B-PERS

- Precision and recall (exact match) ignoring NE categories?
  → P=4, T=3, TP=1, Prec=1/4, Rec=1/3, F=2/7≈0.28
- Precision and recall (fuzzy match) ignoring NE categories?
  → P=7, T=7, TP=5, Prec=5/7, Rec=5/7, F=5/7≈0.71
- Precision and recall (exact match) per NE category?
  → PERS: P=2, T=2, TP=1, Prec=Rec=F=1/2=0.5
  → ORG: P=2, T=1, TP=0, Prec=Rec=F=0
- BIO tag accuracy?

NER evaluation: exercise

Input →	***Mary***	***Allen***	*buys*	***Big***	***Pasta***	***Inc.***	*from*	*former*	*owner*	***Jim***	***Smith***
Gold →	B-PERS	I-PERS	O	B-ORG	I-ORG	I-ORG	O	O	O	B-PERS	I-PERS
Pred. →	B-PERS	I-PERS	O	B-ORG	I-ORG	O	O	B-ORG	I-ORG	O	B-PERS

- Precision and recall (exact match) ignoring NE categories?
  - → P=4, T=3, TP=1, Prec=1/4, Rec=1/3, F=2/7≈0.28
- Precision and recall (fuzzy match) ignoring NE categories?
  - → P=7, T=7, TP=5, Prec=5/7, Rec=5/7, F=5/7≈0.71
- Precision and recall (exact match) per NE category?
  - → PERS: P=2, T=2, TP=1, Prec=Rec=F=1/2=0.5
  - → ORG: P=2, T=1, TP=0, Prec=Rec=F=0
- BIO tag accuracy?
  - → TP=6, P=T=11, Acc=6/11≈0.55

Provided code to evaluate NER

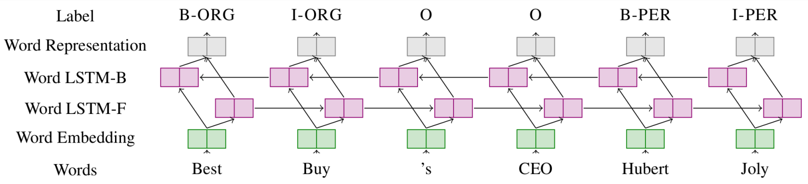
```
./accuracy.py --pred pred/sequoia.pred \
              --gold sequoia.test \
              --tagcolumn parseme:ne

Accuracy on all parseme:ne: 80.99 ( 7773/ 9598) # <- IGNORE!

Precision, recall, and F-score on parseme:ne:
Exact-EVE   : P= 13.79 ( 12/   87) R= 60.00 ( 12/ 20) F= 22.43
Exact-LOC   : P= 33.70 ( 31/   92) R= 40.79 ( 31/ 76) F= 36.90
Exact-ORG   : P= 47.06 ( 48/  102) R= 39.34 ( 48/122) F= 42.86
Exact-PERS  : P= 14.93 ( 80/  536) R= 65.57 ( 80/122) F= 24.32
Exact-PROD  : P= 50.70 ( 36/   71) R= 54.55 ( 36/ 66) F= 52.55
Exact-nocat: P= 26.35 (234/  888) R= 57.64 (234/406) F= 36.17
Fuzzy-nocat: P= 38.11 (686/1800) R= 85.75 (686/800) F= 52.77
macro-avg   : P= 32.09             R= 57.66            F= 41.23
```

RNNs for named entity recognition

- Segmentation problem can be solved with tagging model
- (Bidirectional) RNN predicts each word's BIO tag from hidden state
  - $\longrightarrow$ Similar to POS tagging, different output vocabulary V_t



RNNs for NER: class imbalance

- Cross-entropy may be problematic for sporadic annotations
  - $\rightarrow$ Continuous version of accuracy - bad metric
  - $\rightarrow$ Take little risk: predict only `O` tags – most frequent
- Solution 1: different weights $\mathbf{w}$ per tag in loss

$$L(y, \widehat{\mathbf{y}}) = -\mathbf{w}[y] \log \frac{\exp(\widehat{\mathbf{y}}[y])}{\sum_{c=1}^{C} \exp(\widehat{\mathbf{y}}[c])}$$

  - $\rightarrow$ $\widehat{\mathbf{y}}$=logits vector, $\mathbf{w}$=weights vector, y=gold tag index
- Solution 2: task-specific loss, e.g. derivable F-score

RNNs for NER: invalid sequences

- Greedy prediction – choose argmax tag for each word
  - → Can predict invalid tag sequences!

$$\hat{t}_i = \operatorname*{argmax}_{t \in V_t} P(t|w_1 \ldots w_n)$$

- Solution 1: post-processing heuristics
  - → Greedy prediction + rules to "fix" output
- Solution 2: adapt the Viterbi algorithm
  - → Requires estimating tag transition probabilities
- Solution 3: Conditional random fields (CRFs)
  - → Loss includes tag sequence features
  - → Invalid sequences less probable but still possible

BIOES tagging

- B: begin, I: inside, O: outside
- E: end of a segment
- S: single-word segment (B and E at the same time)

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

BIO for tokenisation

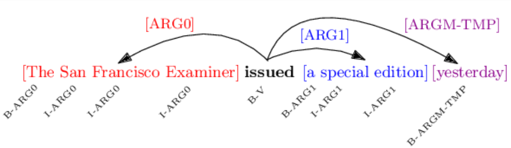
- Regexp usually enough for languages using whitespace
  - Cannot deal with Chinese, compounds, etc.
- Character model: predict 1 to start a word, 0 to continue
- More complex encoding: character-based BIOES
  - begin, inside, outside (X), end, single

```
On considère qu'environ 50 000 Allemands
BEXBIIIIIIIEXBIEBIIIIIEXBIIIIEXBIIIIIIIEX
du Wartheland ont péri pendant la période.
BEXBIIIIIIIIEXBIEXBIIEXBIIIIIEXBEXBIIIIIES
```

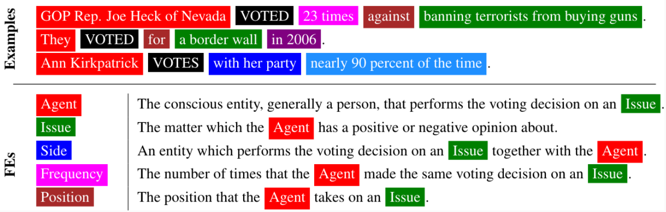
Source: `https://aclanthology.org/Q18-1030/`

BIO for semantic parsing

- Semantic role labelling (Propbank)



- Frame semantic parsing (FrameNet)



Source: `https://idir.uta.edu/src/claimframe.html`

BIO for multiword expressions

- Multiword expressions: word combinations acting as units
  - → ***take*** *a* ***shower***, ***dead end***, ***make ends meet***, ***give*** *it* ***up***. . .
- Multiword expressions can be discontinuous
  - → New tag `G` for gaps

Sentence	Jean	prend	de	longues	douches	.
BIO	O	B	G	G	I	O
IO+cat	O	I-LVC	G	G	I-LVC	O
BIO+cat	O	B-LVC	G	G	I-LVC	O

Thanks!

That's all for today

Carlos Ramisch (carlos.ramisch@univ-amu.fr)

With the help of Benoit Favre & Alexis Nasr

Prédiction Structurée pour le Traitement Automatique des Langues
Master IAAA
Aix Marseille Université

Sources i

- Benoit Favre's PSTALN website – https://pageperso.lis-lab.fr/benoit.favre/pstaln/
- Alexis Nasr's slides from TLNL and ML course – master 2 IAAA
- Dan Jurafsky & James H. Martin, Speech and Language Processing 3 (Online edition Aug 2024) – https://web.stanford.edu/~jurafsky/slp3/
- Shao et al. 2018 – https://aclanthology.org/Q18-1030/
- Universal Dependencies – https://universaldependencies.org/
- PARSEME-FR projec documentation – https://gitlab.lis-lab.fr/PARSEME-FR/PARSEME-FR-public/

- Discussions with Bruno Guillaume, Marie Candito, Benoit Favre, Alexis Nasr, Frédéric Béchet
- Feedback from participants of previous course editions

Sources ii

- Slides illustrated with the help of: Google images, `imgupscaler.com`, Canva
- Slides written with the help of: ChatGPT, Google Bard, DeepL, Linguee, Overleaf

Backup slides