U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

M.EIC

Natural Language Processing

Henrique Lopes Cardoso

FEUP / LIACC

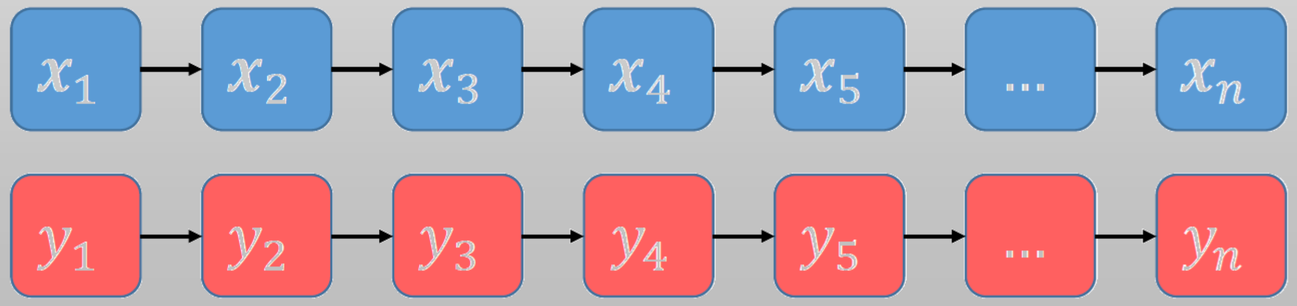
hlc@fe.up.pt

Sequence Labeling

POS-tagging, Named Entity Recognition, Hidden Markov Models, Conditional Random Fields

Sequence Labeling

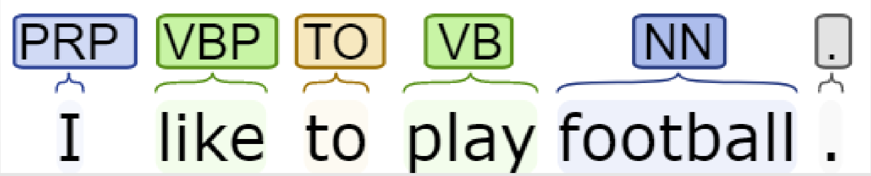
- Assign a label chosen from a small fixed set of labels to each element of a sequence
  - Input: sequence of n tokens (words) $\{x_1, x_2, \dots, x_n\}$
  - Output: sequence of n tags (labels) $\{y_1, y_2, \dots, y_n\}$, $y_i \in T = \{t_1, \dots, t_k\}$



Applications of Sequence Labeling

- Part-of-speech (POS) tagging: assign a morphosyntactic category to each word in a sentence
- Named Entity Recognition (NER): find and classify spans of text that correspond to concepts of interest (e.g. names, places, organizations, …) in some task domain
- Argument Component Detection: find claims and premises
- Code Switching: find segments of different languages in multilingual text

Parts of Speech (POS)



- The class or syntactic category of a word tells us about likely neighboring words and syntactic structure – a key aspect of parsing

- POS are useful features
  - for labeling named entities (people, organizations, …)
  - for coreference resolution
  - for speech recognition or synthesis (e.g. *CONtent* vs *conTENT*)

POS Classes (in languages such as English or Portuguese)

- Closed classes: prepositions, particles, determiners, conjunctions, pronouns, auxiliary verbs, numerals
  - Fixed membership
  - Usually function words
- Open classes: nouns, verbs, adjectives, adverbs, interjections
  - More elements are added all the time
  - Nouns: can occur with determiners, take possessives and be adjectivized
    - Proper nouns are usually capitalized
  - Verbs: refer to actions and processes, have inflections of tense, person and number
  - Adjectives: describe properties or qualities of nouns
  - Adverbs: modify something (often verbs)

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

The Universal Dependencies POS Tagset

	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

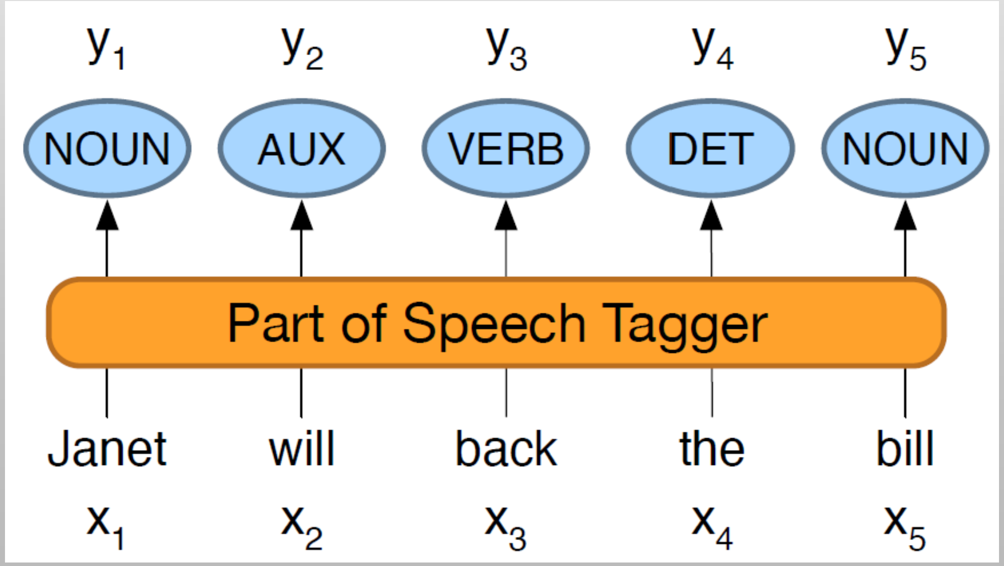
The Penn Treebank POS Tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"to"	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past partici- ple	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your, one's*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+,%, &*	WRB	wh-adverb	*how, where*

- The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
- There/EX are/VBP 70/CD children/NNS there/RB
- Preliminary/JJ findings/NNS were/VBD reported/VBN in/IN today/NN 's/POS New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

POS Tagging

- From a sequence of words to a sequence of tags
  - Assign a POS marker to each word



Ambiguity in POS Tagging

- Ambiguous words have more than one possible POS
  - ***book*** *a flight / buy a* ***book*** *hand me* ***that*** *book / I thought* ***that*** *you were happy*
  - *tanto* ***como*** *peixe* ***como*** *carne* *uma* ***liga*** *metálica / isto não* ***liga***
  - ***They can fish***: PRP MD VBP / PRP VBP NNS
  - ~15% of the vocabulary, ~60% of word tokens in running text (genre-dependent)

		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Baseline Algorithm for POS Tagging

- POS tagging aims at resolving these ambiguities

- Many words are easy to disambiguate, because their different tags aren't equally likely!

- Most Frequent Class Baseline: choose the most frequent tag for that word in the training corpus
  - Baseline accuracy: +90%
  - SOTA/human ceiling accuracy: ~97%

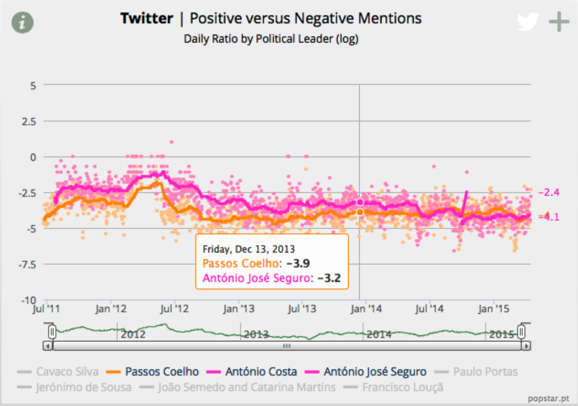
Named Entity Recognition (NER)

- Named entity: anything that can be referred to with a "proper name"
  - Person, location, organization, geo-political entity
- And also other kinds of things of interest:
  - Temporal expressions (dates, times)
  - Numerical expressions (prices)
  - …
- Application-specific types
  - Biomedical NLP: protein, DNA, RNA, cell line, cell type

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

Why NER?

- Entity monitoring / sentiment analysis
  - What entity is a consumer review about?
  - What are people saying about a particular entity?

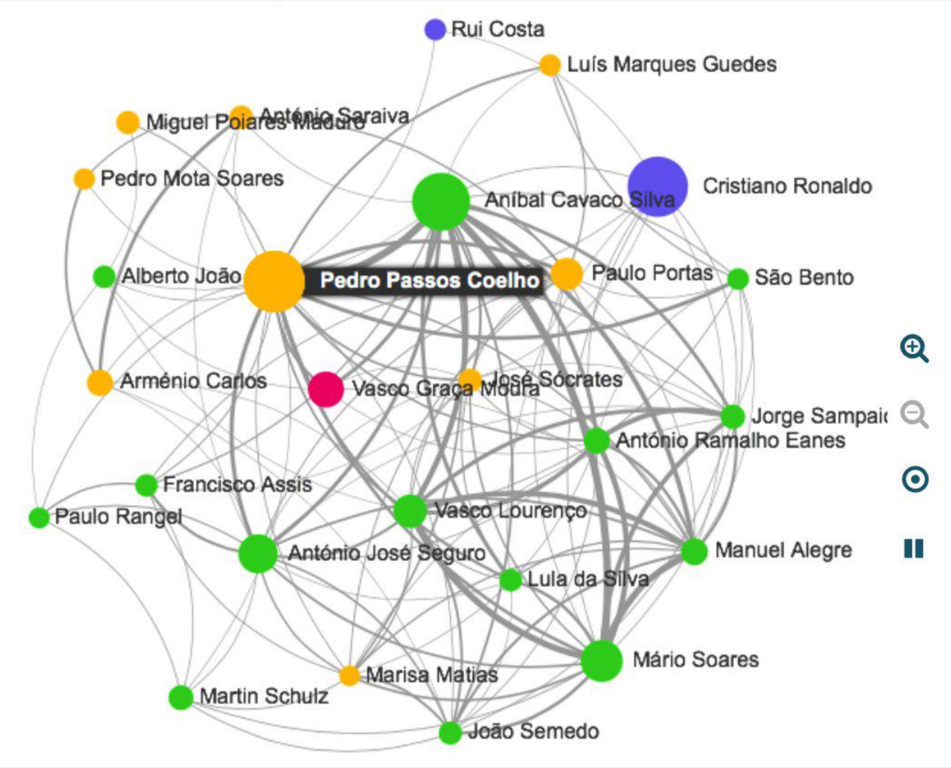


- Efficient search algorithms and question answering
  - Which documents mention the targeted entity?

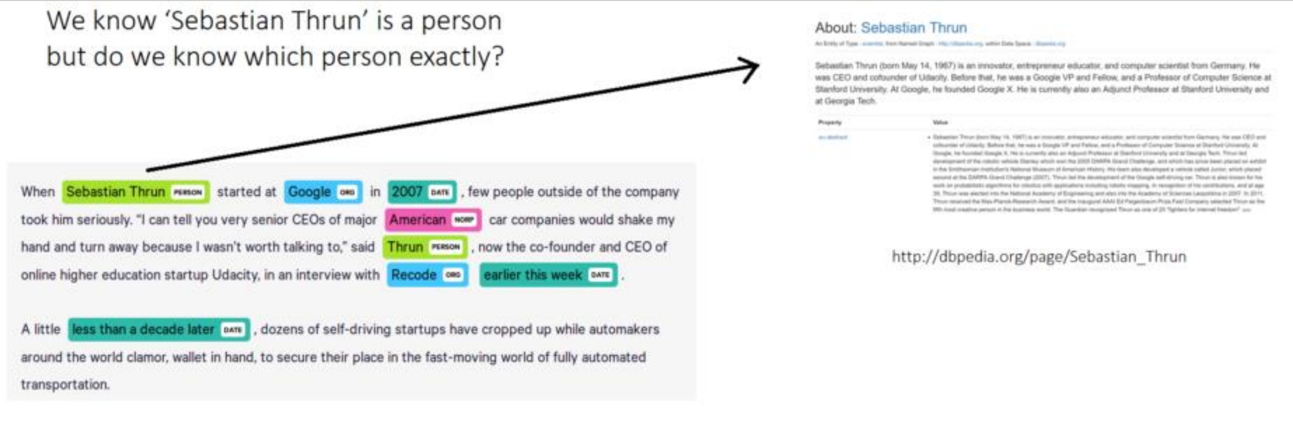
- Content aggregation / recommendation

Why NER?

- Entity relations through co-occurrence graphs



- Linking: how do we link text to information in structured knowledge sources (e.g., Wikipedia)?
  - Named Entity Linking, Wikification



Why NER?

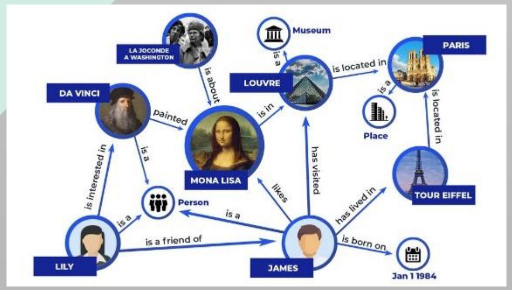
- Information extraction: how can we build semantic representations from the relationships between entities (names, organizations, locations, dates, events, …) mentioned in the text?

Text → Named Entity Recognition and Disambiguation → Coreference Resolution → Relation Extraction → Knowledge Graph

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed at turpis vitae velit euismod aliquet. Pellentesque et arcu. Nullam venenatis gravida orci. Pellentesque et arcu. Nam pharetra. Vestibulum viverra varius enim.

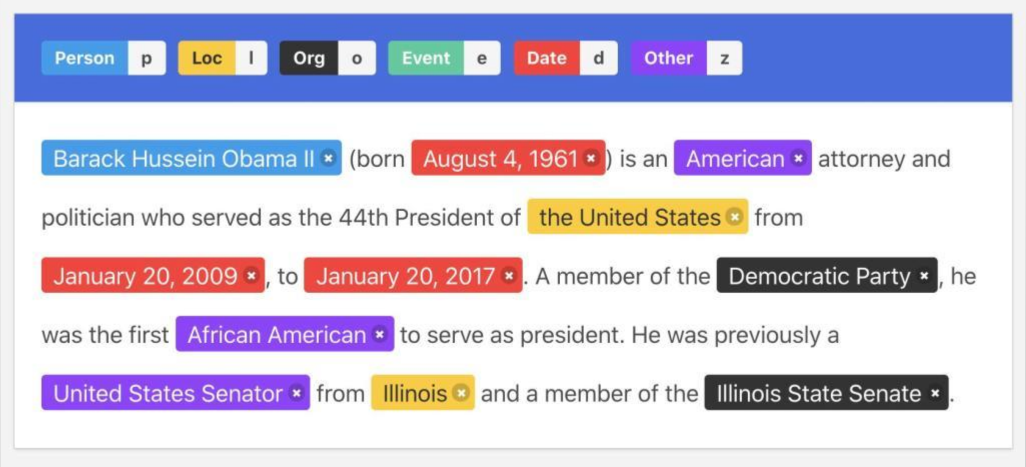
Nam laoreet dui sed magna. Nunc in turpis ac lacus eleifend sagittis. Pellentesque ac turpis. Aliquam justo lectus, iaculis a, auctor sed, congue in, nisl. Aenean luctus vulputate turpis. Mauris urna sem, suscipit vitae, dignissim id, ultrices sed, nunc.

Phasellus nisi metus, tempus sit amet, ultrices ac, porta nec, felis. Quisque malesuada nulla sed pede volutpat pulvinar. Sed non ipsum. Mauris et dolor. Pellentesque suscipit accumsan massa. In consectetuer, lorem eu lobortis egestas, velit odio



Ambiguity in NER

- Ambiguity challenges:
  - Segmentation: entity boundaries
    - *The Washington Post, Vila Nova de Gaia, Marcelo Rebelo de Sousa*
  - Type: several entities of different types with the same name
    - *JFK* can be a person, an airport, …



[$_{PER}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{ORG}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{LOC}$ Washington] for what may well be his last state visit.
In June, [$_{GPE}$ Washington] passed a primary seatbelt law.

NER Tagsets

- CoNLL 2003
  - LOC (location), ORG (organization), PER (person), MISC (miscellaneous)
- OntoNotes 5.0

PERSON	People, including fictional	LAW	Named documents made into laws
NORP	Nationalities or religious or political groups	LANGUAGE	Any named language
FACILITY	Buildings, airports, highways, bridges, etc.	DATE	Absolute or relative dates or periods
ORGANIZATION	Companies, agencies, institutions, etc.	TIME	Times smaller than a day
GPE	Countries, cities, states	PERCENT	Percentage (including “%”)
LOCATION	Non-GPE locations, mountain ranges, bodies of water	MONEY	Monetary values, including unit
PRODUCT	Vehicles, weapons, foods, etc. (Not services)	QUANTITY	Measurements, as of weight or distance
EVENT	Named hurricanes, battles, wars, sports events, etc.	ORDINAL	“first”, “second”
WORK OF ART	Titles of books, songs, etc.	CARDINAL	Numerals that do not fall under another type

NER Tagging

- Treat NER as a word-by-word sequence labeling task (just like POS)

- BIO (or IOB) encoding: Begin, In, Out
  - $2n+1$ tags, where n is the number of entity types

Marcelo	Rebelo	de	Sousa	is	going	to	Los	Angeles	in	California
B-PER	I-PER	I-PER	I-PER	O	O	O	B-LOC	I-LOC	O	B-LOC

- BILOU (or BIOES) encoding: Begin, In, Last/End, Out, Unit/Single

Marcelo	Rebelo	de	Sousa	is	going	to	Los	Angeles	in	California
B-PER	I-PER	I-PER	L-PER	O	O	O	B-LOC	L-LOC	O	U-LOC

POS vs NER Evaluation

- Part of Speech Tagging: accuracy
  - Unit is the word, as each word is assigned a tag individually

- Named Entity Recognition: recall, precision, F1
  - Unit is the entity
  - NER has a segmentation component
    - *Cristiano*$_{\text{B-PER}}$ *Ronaldo*$_{\text{I-PER}}$ *has*$_{\text{O}}$ *scored*$_{\text{O}}$ *again*$_{\text{O}}$ $\rightarrow$ labeling "Cristiano" as a person (but not "Ronaldo") is both a false positive (O) and a false negative for the entity (missing I-PER)
    - Mismatch between training and test conditions: entities as the unit of response, words as the unit of training
  - See also MUC metrics for partial matches

Hidden Markov Models (HMM)

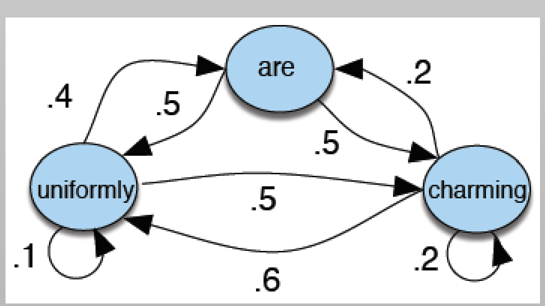
- HMM is a probabilistic sequence model
  - Given a sequence of units (words), compute a probability distribution over possible sequences of labels

- Markov chain: assigning probabilities to sequences of random variables
  - Markov assumption: to predict the future, we only care about the current state
    - $P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$

A set Q of n states

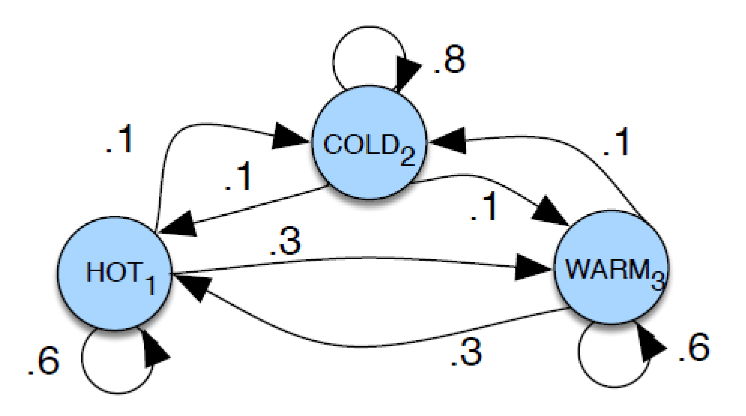
A transition probability matrix $A: Q \times Q \rightarrow [0,1]$, with $\sum_{j=1}^{n} a_{ij} = 1, \forall i$

An initial probability distribution π over states, with $\sum_{i=1}^{n} \pi_i = 1$



Markov Chain

- A Markov chain for weather:



- If $\pi = [0.1, 0.7, 0.2]$, what is the probability for each of the following sequences?
  - HOT → HOT → HOT → HOT
  - COLD → HOT → COLD → HOT
- What does this tell us about the weather?

Hidden Markov Models (HMM)

- The events we are interested in are hidden
  - We observe words, but would like to use information not observed directly in the input text (e.g., **tags**)
- HMM: take into account both observed (word) and hidden (tag) events

A set Q of n states

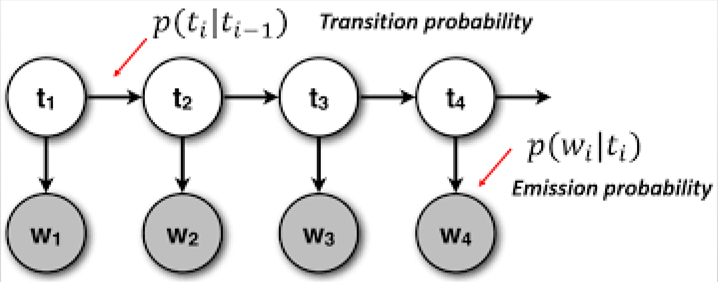
A transition probability matrix $A: Q \times Q \rightarrow [0,1]$, with $\sum_{j=1}^{n} a_{ij} = 1, \forall i$

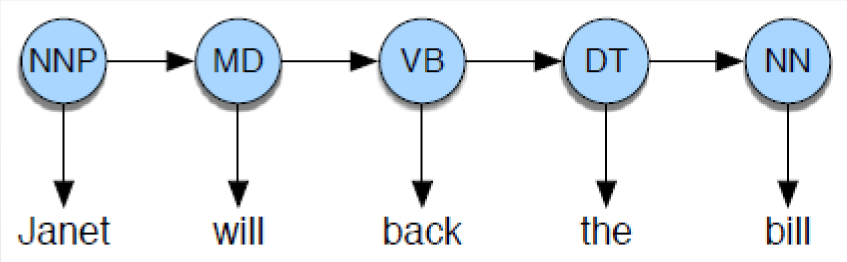
A sequence $O = o_1 o_2 \ldots o_T$ of observations drawn from a vocabulary V

A sequence $B = b_i(o_t)$ of observation likelihoods (emission probabilities) expressing the probability of observation o_t being generated from state q_i

An initial probability distribution π over states, with $\sum_{i=1}^{n} \pi_i = 1$

Hidden Markov Models (HMM)





- Markov assumption: $P(q_i|q_1 \dots q_{i-1}) = P(q_i|q_{i-1})$
- Output independence assumption: $P(o_i|q_1 \dots q_i \dots q_T, o_1 \dots o_T) = P(o_i|q_i)$

Computing Transition and Emission Probabilities

- Transition probabilities $P(t_i|t_{i-1})$

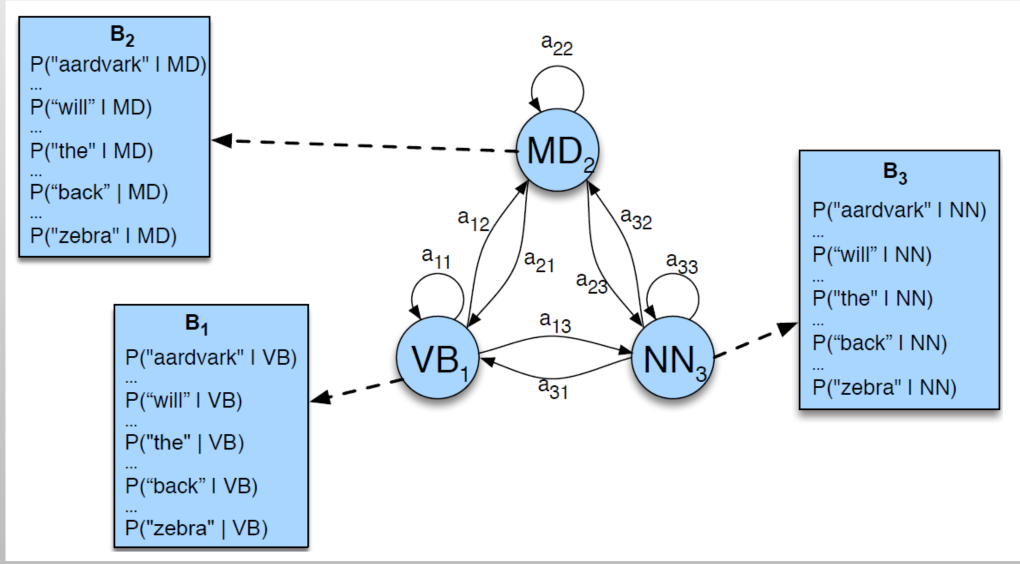
$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

  - Example: $P(VB|MD) = \frac{C(MD,VB)}{C(MD)} = \frac{10471}{13124} = .80$

- Emission probabilities $P(w_i|t_i)$

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

  - Example: $P(will|MD) = \frac{C(MD,will)}{C(MD)} = \frac{4046}{13124} = .31$



Decoding

- Decoding is the task of determining the sequence of hidden variables
  - Given transition (A) and emission (B) probabilities and a sequence of observations O, find the most probable sequence of states Q

- POS tagging: choose the most probable tag sequence given the observed word sequence

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

  - Bayes rule:

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

  - Dropping the denominator:

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

Decoding

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

- Output independence assumption: the probability of a word depends only on its own tag

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

- Markov assumption: the probability of a tag depends only on the previous tag (bigrams)

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

- Most probable tag sequence:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}}\ \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

The Viterbi Algorithm

initial probability for each state s

emission probability for the first word

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
    $viterbi[s,1] \leftarrow \boxed{\pi_s} * \boxed{b_s(o_1)}$
    $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $\boxed{viterbi[s,t] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)}$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \; viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

$$v_t(j) = \max_{i=1}^{N} \boxed{v_{t-1}(i)} \boxed{a_{ij}} \boxed{b_j(o_t)}$$

previous Viterbi path probability

transition probability

emission probability (observation likelihood)

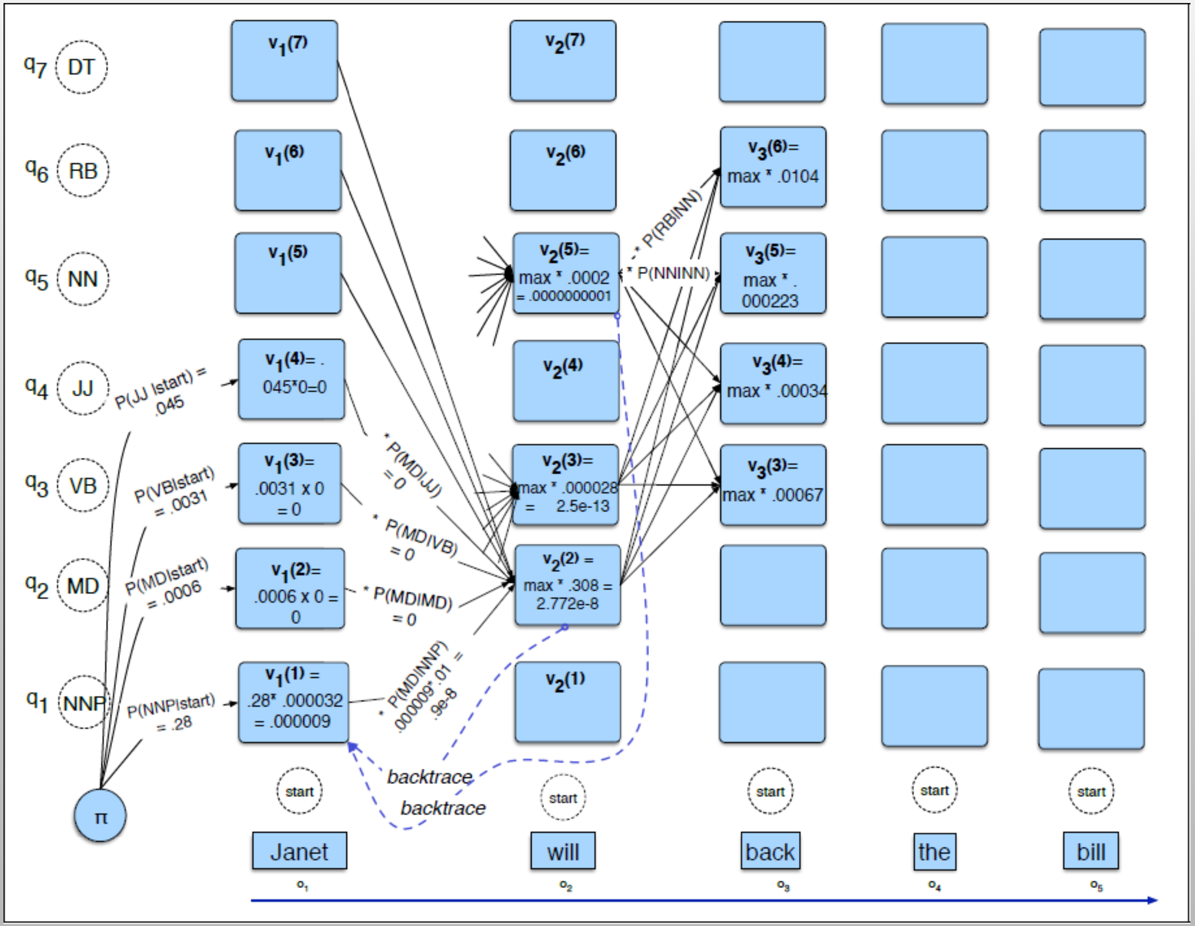
Viterbi Probability Matrix

- Transition probabilities:

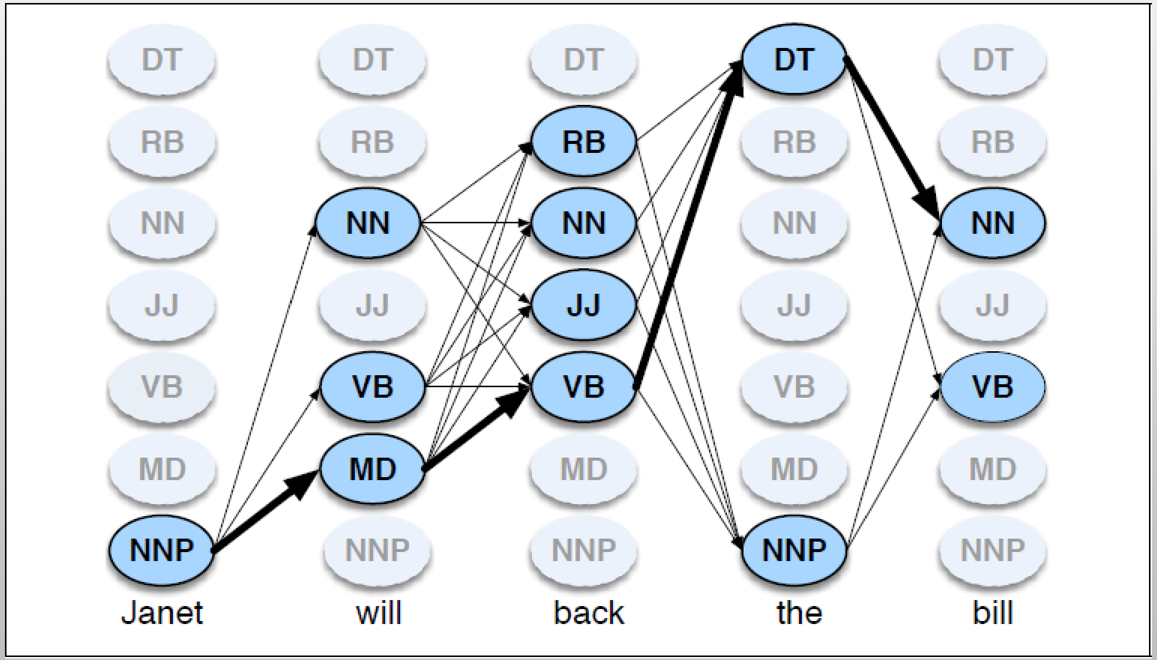
	NNP	MD	VB	JJ	NN	RB	DT
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	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

- Emission probabilities:

	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



Viterbi Path Matrix



HMM Extensions

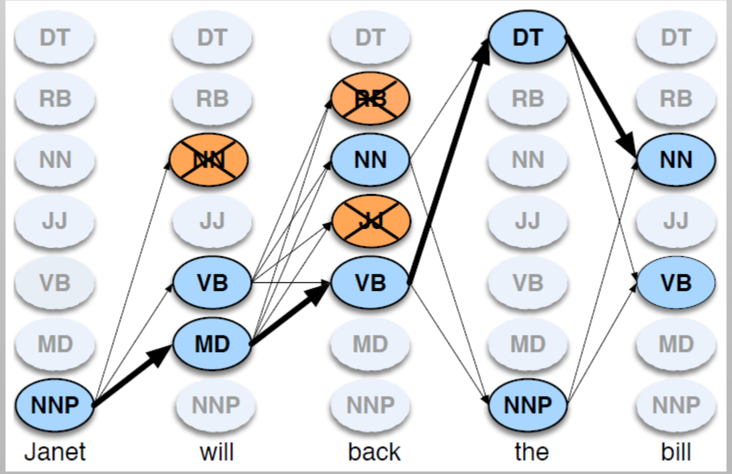
- Trigrams

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}, t_{i-2})$$

  - Requires changing the Viterbi algorithm (consider N^2 paths through cells in the previous two columns)
  - Trigram sparsity: use interpolation (as in language modeling)

- Beam search when the number of states is large:

  Viterbi is $O(N^2T)$ for trigram taggers



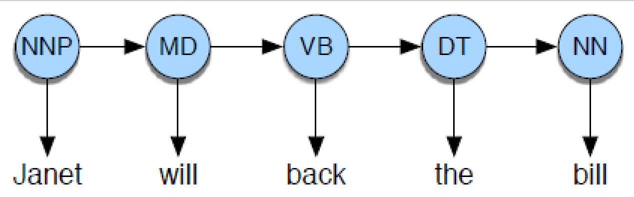
Maximum Entropy Markov Models (MEMM)

- Problem with HMMs: how do we deal with unknown words?
- Adding arbitrary features
  - Capitalization or morphology
  - Looking at the surrounding words
- MEMM is a discriminative model based on multinomial logistic regression (aka maximum entropy)
- The Markov part of MEMM allows us to deal with sequence labeling
  - Use the class assigned to the prior word as a feature and run the classifier on successive words
    (in practice, we'll be using much more than the class assigned to the prior word)

Maximum Entropy Markov Models (MEMM)

Hidden Markov Model (HMM)

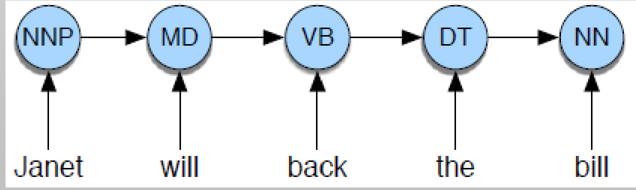
$$
\begin{aligned}
\hat{T} &= \underset{T}{\operatorname{argmax}}\, P(T|W) \\
&= \underset{T}{\operatorname{argmax}}\, P(W|T)P(T) \\
&= \underset{T}{\operatorname{argmax}} \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})
\end{aligned}
$$



Maximum Entropy Markov Model (MEMM)

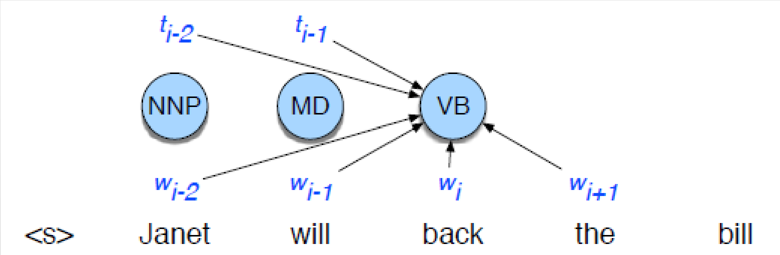
[McCallum et al., 2000]

$$
\begin{aligned}
\hat{T} &= \underset{T}{\operatorname{argmax}}\, P(T|W) \\
&= \underset{T}{\operatorname{argmax}} \prod_i P(t_i|w_i, t_{i-1})
\end{aligned}
$$



Feature Functions

- It's easy to incorporate a lot of features, based on the whole input sequence and prior states



- Probability of each local tag:

$$p(t_i|t_{i-1}, W) = \frac{e^{\sum_k \theta_k f_k(t_i, t_{i-1}, W)}}{\sum_{t' \in tagset} e^{\sum_k \theta_k f_k(t', t_{i-1}, W)}}$$

Feature Functions

- Example feature functions:

$$\mathbb{1}\{x_i = \textit{the},\ y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN},\ x_{i+1} = \textit{Street},\ y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB},\ y_{i-1} = \text{AUX}\}$$

- Feature templates:

$\langle y_i, x_i\rangle, \langle y_i, y_{i-1}\rangle, \langle y_i, x_{i-1}, x_{i+2}\rangle$ → *Janet$_{NNP}$ will$_{MD}$* ***back$_{VB}$*** *the$_{DT}$ bill$_{NN}$* →

f_{3743}: y_i = VB and x_i = back
f_{156}: y_i = VB and y_{i-1} = MD
f_{99732}: y_i = VB and x_{i-1} = will and x_{i+2} = bill

- Unknown words: express properties of the word's spelling or shape
  - w_i contains a particular prefix or suffix
  - w_i's word shape

well-dressed →

prefix(x_i) = w
prefix(x_i) = we
suffix(x_i) = ed
suffix(x_i) = d
word-shape(x_i) = xxxx-xxxxxxx
short-word-shape(x_i) = x-x

Features for NER

- Identity of w_i and neighboring words
- Embeddings for w_i and neighboring words
- POS of w_i and neighboring words
- Presence of w_i in a gazetteer
- w_i contains a particular prefix
- w_i contains a particular suffix
- Case-sensitive word shape of w_i and neighboring words
- Case-sensitive short word shape (including capitalization) of w_i and neighboring words

Decoding MEMMs

- Most likely sequence of tags:

$$
\begin{aligned}
\hat{T} &= \underset{T}{\operatorname{argmax}}\, P(T|W) \\
&= \underset{T}{\operatorname{argmax}} \prod_i P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \underset{T}{\operatorname{argmax}} \prod_i \frac{\exp\left(\sum_j \theta_j f_j(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in \text{tagset}} \exp\left(\sum_j \theta_j f_j(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}
$$

  - Each tag depends on:
    - words within w_{i-l}^{i+l} (including the current word)
    - the previous tags t_{i-k}^{i-1}

Decoding MEMMs

- Turning logistic regression into a sequence model:
  - Greedy: classify from left to right

function GREEDY SEQUENCE DECODING(words W, model P) **returns** tag sequence T

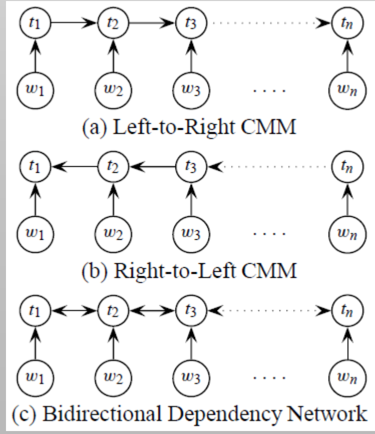
for $i = 1$ **to** *length(W)*
$$\hat{t}_i = \underset{t' \in T}{\operatorname{argmax}} \ P(t' \mid w_{i-l}^{i+l}, t_{i-k}^{i-1})$$

  - Or use Viterbi decoding (replacing transition and emission priors with the direct posterior):

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i)\, P(o_t|s_j) \longrightarrow v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i, o_t) \quad 1 \leq j \leq N, 1 < t \leq T$$

Bidirectionality

- MEMMs suffer from label bias
  - Labels having higher priors sometimes prevent the correct labeling sequence
    - *will to fight* → NN TO VB, but we can get a MD TO VB because $P(MD|will,\langle s\rangle) > P(NN|will,\langle s\rangle)$ and $P(TO|to,t_{will}) \approx 1$ for any t_{will}
- Multiple passes
  - Use POS features from left disambiguated words; use tags for all words (also from the right)
  - Left-to-right and right-to-left
    - Greedy decoding: choose the highest-scoring tags from both passes
    - Viterbi decoding: choose the higher scoring of the two sequences
  - Bidirectional version of MEMM: Stanford tagger (cyclic dependency network)



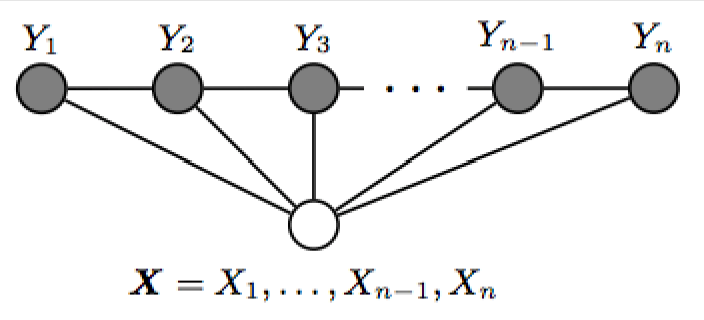
(a) Left-to-Right CMM
(b) Right-to-Left CMM
(c) Bidirectional Dependency Network

[Toutanova et al., 2003]

Conditional Random Fields (CRF)

- MEMM uses per-state exponential models for the conditional probabilities of next states given the current state

- CRF [Lafferty et al., 2001] uses a single exponential model to determine the joint probability of the entire sequence of labels, given the observation sequence



Conditional Random Fields (CRF)

- CRF normalizes probabilities over all tag sequences, which requires computing the sum over all possible labelings

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

  - Global features $F_k(X, Y)$: each is a property of the entire input and output sequences X and Y

Conditional Random Fields (CRF)

- Decomposing into a sum of local features for each position:

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

  - Current output token y_i, previous output token y_{i-1}, entire input string X and the current position i
  - Linear chain CRF: looking only to the current and previous state enables the use of Viterbi decoding

- Decoding

$$\hat{Y} = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} P(Y|X) = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)$$

Practical POS and NER

- Labeled data (examples)
  - Universal Dependencies (UD): morphosyntactic annotations for +100 languages
  - OntoNotes: corpora labeled for named entities in English, Chinese, and Arabic
  - BioRED: biomedical relation extraction dataset
- Rule-based methods
  - Commercial approaches to NER are often based on pragmatic combinations of lists and rules: regular expressions, dictionaries, semantic constraints
    1. Use high-precision rules to tag unambiguous entity mentions.
    2. Search for substring matches of the previously detected names.
    3. Use application-specific name lists to find likely domain-specific mentions.
    4. Apply supervised sequence labeling techniques that use tags from previous stages as additional features.

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

The Python Notebook

jupyter

sequence-labeling_.ipynb

- POS tagging in NLTK
- NER in NLTK: chunking through POS tags
- spaCy language processing pipelines

sequence-labeling-training_.ipynb

- Using an annotated corpus for NER
- Conditional Random Fields: sklearn-crfsuite

NLTK

https://www.nltk.org/

spaCy

https://spacy.io/