IN4080 – 2022 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Lecture 6, 29 Sept.

Today

- N-gram language models
  - (hangover from last week)
- POS-tagging
- HMM-tagging
- Discriminative tagging
- Named-entity recognition (NER)
- Evaluation of NER

Probabilistic Language Models

- Goal: Ascribe probabilities to word sequences.
  - $P(w_1, w_2, w_3, \ldots, w_n)$
- Related: the probability of the next word
  - $P(w_n \mid w_1, w_2, w_3, \ldots, w_{n-1})$
- A model which does either is called a Language Model, LM

Markov assumption

- A word depends only on the immediate preceding word
- $P(w_1, w_2, w_3, \ldots, w_n) \approx$
- $P(w_1) \times P(w_2 | w_1) \times P(w_3 | w_2) \times \cdots \times P(w_n | w_{n-1}) =$
- $\prod_i^n P(w_i | w_{i-1})$

- P("its water is so transparent") ≈

  P(its) × P(water | its) × P(is | water) × P(so | is) × P(transparent | so)

- This is called a bigram model

General *n*-gram models

- A word depends only on the k many immediately preceding words
- $P(w_1, w_2, w_3, \ldots, w_n) \approx$
- $\prod_i^n P(w_i | w_{i-k}, w_{i+1-k}, \ldots, w_{i-1}) = \prod_i^n P\left(w_i | w_{i-k}^{i-1}\right)$

- This is called a
  - unigram model – no preceding words
  - trigram model – two preceding words
  - *k*-gram model – *k*-1 preceding words

Markov assumption:
A word depends only on the immediate preceding word

Generating with n-grams

- Goal: Generate a sequence of words
- Unigram:
  - Choose the first word according to how probable it is
  - Choose the second word according to how probable it is, etc.
  - = the generative model for multinomial NB text classification
- Bigram
  - Pick a word *k* according to $\hat{P}(w_i|w_{i-1})$
- *k*-gram
  - Pick a word w_i according to its probability given the $k-1$ preceding words $\hat{P}\left(w_i|\, w_{i-k}^{i-1}\right)$

Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
	–Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
	–What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.
	–This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;
	–It cannot be but so.

Unknown words

- There might be words that is never observed during training.
- Use a special symbol for unseen words during application, e.g. UNK
- Set aside a probability for seeing a new word
  - This may be estimated from a held-out corpus
- Adjust
  - the probabilities for the other words in a unigram model accordingly
  - the conditional probabilities of the *k-gram model*

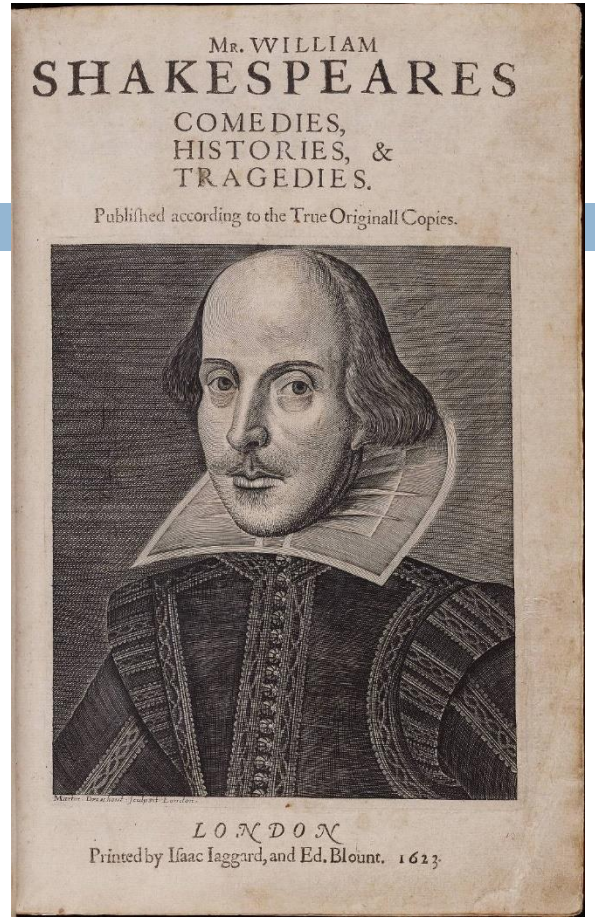
Smoothing, Laplace, Lidstone

(The words in the vocabulary, words we have seen)

- Since we might not have seen all possibilities in training data, we might use Lidstone or, more generally, Laplace smoothing
- $\hat{P}(w_i|w_{i-1}) = \frac{count(w_{i-1},w_i)+k}{count(w_{i-1})+k\,|V|}$
  - where $|V|$ is the size of the vocabulary V.

But:

- Shakespeare produced
  - N = 884,647 word tokens
  - V = 29,066 word types
- Bigrams:
  - Possibilities:
    - $V^2 = 844{,}000{,}000$
  - Shakespeare,
    - bigram tokens: 884,647
    - bigram types: 300,000



- Add-k smoothing is not appropriate for *n*-grams

Smoothing n-grams

Backoff

- If you have good evidence, use the trigram model,
- If not, use the bigram model,
- or even the unigram model

Interpolation

- Combine the models

Use either of this. According to J&M interpolation works better

Interpolation

- Simple interpolation:

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

- The λ-s can be tuned on a held out corpus
- A more elaborate model will condition the λ-s on the context
  - (Brings in elements of backoff)

Evaluation of n-gram models

- Extrinsic evaluation:
  - To compare two LMs, see how well they are doing in an application, e.g. translation, speech recognition
- Intrinsic evaluation:
  - Use a held out-corpus and measure $P(w_1, w_2, w_3, \dots, w_n)^{\frac{1}{n}}$
    - The n-root compensate for different lengths
  - $\prod_i^n P\left(w_i \middle| w_{i-k}^{i-1}\right)^{\frac{1}{n}}$ for a k-gram model
  - It is normal to use the inverse of this, called the perplexity
  - $PP(w_1^n) = \frac{1}{P(w_1, w_2, w_3, \dots, w_n)^{\frac{1}{n}}} = P(w_1, w_2, w_3, \dots, w_n)^{-\frac{1}{n}}$

Properties of LMs

- The best smoothing is achieved with Kneser-Ney smoothing
- Short-comings of all n-gram models
  - The smoothing is not optimal
  - The context are restricted to a limited number of preceding words.

A practical advice: Use logarithms when working with n-grams

Today

- N-gram language models
  - (hangover from last week)
- POS-tagging
- HMM-tagging
- Discriminative tagging
- Named-entity recognition (NER)
- Evaluation of NER

Tagged text and tagging


```
[('They', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('saw', 'NN'), ('.', '.')]
[('They', 'PRP'), ('like', 'VBP'), ('to', 'TO'), ('saw', 'VB'), ('.', '.')]
[('They', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('log', 'NN')]
```

- In tagged text each token is assigned a "part of speech" (POS) tag
- A tagger is a program which automatically ascribes tags to words in text
- From the context we are (most often) able to determine the tag.
  - But some sentences are genuinely ambiguous and hence so are the tags.

Various POS tag sets

- A tagged text is tagged according to a fixed small set of tags.
- There are various such tag sets.
- Brown tagset:
  - Original: 87 tags
  - Versions with extended tags <original>-<more>
    - Comes with the Brown corpus in NLTK
- Penn treebank tags: 35+9 punctuation tags
- Universal POS Tagset, 12 tags,

Brown vs. Penn: Nouns

NN	Noun, sing. or mass	*llama*
NNS	Noun, plural	*llamas*
NNP	Proper noun, singular	*IBM*
NNPS	Proper noun, plural	*Carolinas*

Penn treebank

NN	(common) singular or mass noun	time, world, work, school, family, door
NN$	possessive singular common noun	father's, year's, city's, earth's
NNS	plural common noun	years, people, things, children, problems
NNS$	possessive plural noun	children's, artist's parent's years'
NP	singular proper noun	Kennedy, England, Rachel, Congress
NP$	possessive singular proper noun	Plato's Faulkner's Viola's
NPS	plural proper noun	Americans Democrats Belgians Chinese Sox
NPS$	possessive plural proper noun	Yankees', Gershwins' Earthmen's
NR	adverbial noun	home, west, tomorrow, Friday, North,
NR$	possessive adverbial noun	today's, yesterday's, Sunday's, South's
NRS	plural adverbial noun	Sundays Fridays

Brown, original

Different tagsets - example

			Brown	Penn treebank ('wsj')	Universal
	he	she	PPS	PRP	PRON
I			PPSS	PRP	PRON
me	him	her	PPO	PRP	PRON
my	his	her	PP$	PRP$	DET
mine	his	hers	PP$$	?	PRON

Ambiguity rate

		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%**)

Figure 8.2 Tag ambiguity for word types in Brown and WSJ, using Treebank-3 (45-tag) tagging. Punctuation were treated as words, and words were kept in their original case.

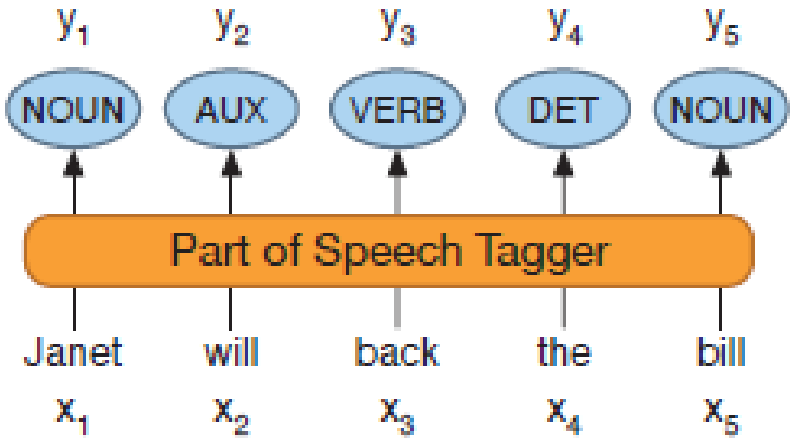
How ambiguous are tags (J&M, 2.ed)

		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well, beat*)	32	
	7 tags	2	(*still, down*)	6	(*well, set, round, open, fit, down*)
	8 tags			4	(*'s, half, back, a*)
	9 tags			3	(*that, more, in*)

BUT: Not directly comparable because of different tokenization

Tagging as Sequence Classification

- Classification (earlier):
  - a well-defined set of observations, O
  - a given set of classes, $S=\{s_1, s_2, \ldots, s_k\}$
  - Goal: a classifier, γ, a mapping from O to S



- Sequence classification:
  - Goal: a classifier, γ, a mapping from sequences of elements from O to sequences of elements from S:
  - $\gamma(o_1, o_2, \ldots o_n) = (s_{k1}, s_{k2}, \ldots s_{kn})$

Baseline tagger

- In all classification tasks establish a baseline classifier.
- Compare the performance of other classifiers you make to the baseline.
- For tagging, a natural baseline is the Most Frequent Class Baseline:
  - Assign each word the tag to which is occurred most frequent in the training set
  - For words unseen in the training set, assign the most frequent tag in the training set.

Today

- N-gram language models
  - (hangover from last week)
- POS-tagging
- HMM-tagging
- Discriminative tagging
- Named-entity recognition (NER)
- Evaluation of NER

Hidden Markov Model (HMM) tagger

Extension of language model

- Two layers:
  - Observed: the sequence of words
  - Hidden: the tags/classes where each word is assigned a class

Extension of Naive Bayes

- NB assigns a class to each observation
- An HMM is a sequence classifier:
  It assigns a sequence of classes to a sequence of words

HMM is a probabilistic tagger

Notation:
$t_1^n = t_1, t_2, \ldots t_n$

- The goal is to decide: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$
- Using Bayes theorem: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$
- This simplifies to: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n)$

  because the denominator is the same for all tag sequences

Simplifying assumption 1

- For the tag sequence, we apply the chain rule
  - $P(t_1^n) = P(t_1)P(t_2|t_1)P(t_3|t_1t_2) \dots P\left(t_i|t_1^{i-1}\right) \dots P(t_n|t_1^{n-1})$
- We then assume the Markov (chain) assumption
  - $P(t_1^n) = P(t_1)P(t_2|t_1)P(t_3|t_2) \dots P(t_i|t_{i-1}) \dots P(t_n|t_{n-1})$

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

  - Assuming a special start tag t_0 and $P(t_1) = P(t_1|t_0)$

Simplifying assumption 2

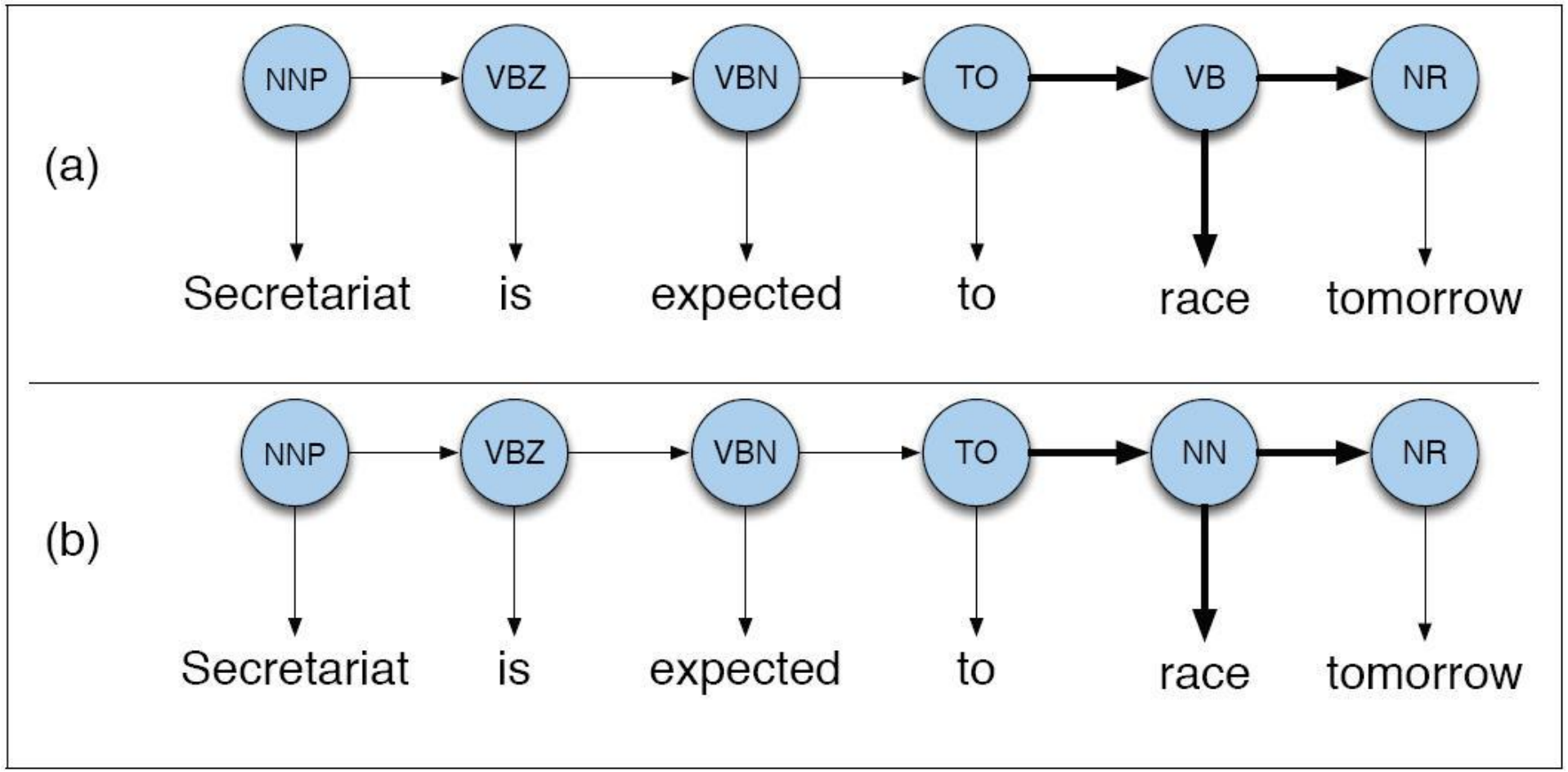
- Applying the chain rule

$$P(w_1^n|t_1^n) = \prod_{i=1}^{n} P\left(w_i|w_1^{i-1}t_1^n\right)$$

  i.e., a word depends on all the tags and on all the preceding words

- We make the simplifying assumption: $P\left(w_i|w_1^{i-1}t_1^n\right) \approx P(w_i|t_i)$
  - i.e., a word depends only on the immediate tag, and hence

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


(a)
NNP
VBZ
VBN
TO
VB
NR
Secretariat
is
expected
to
race
tomorrow
(b)
NNP
VBZ
VBN
TO
NN
NR
Secretariat
is
expected
to
race
tomorrow

Training

- From a tagged training corpus, we can estimate the probabilities with Maximum Likelihood (as in Language Models and Naïve Bayes:)
  - $\hat{P}(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$
  - $\hat{P}(w_i|t_i) = \frac{C(w_i,t_i)}{C(t_i)}$

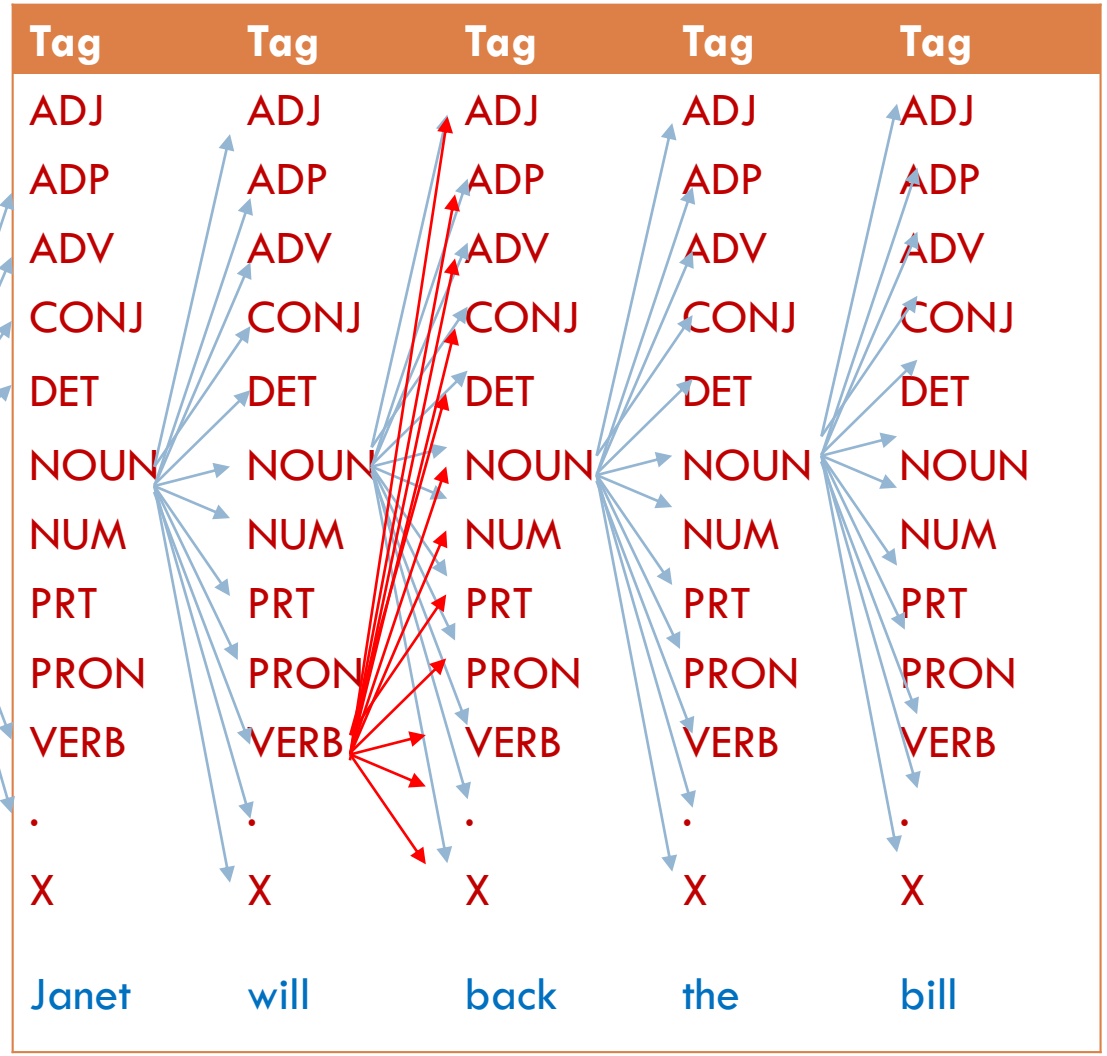
Putting it all together

- From a trained model, it is straightforward to calculate the probability of a sentence with a tag sequence
  - $P(w_1^n, t_1^n) = P(t_1^n)P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \prod_{i=1}^{n} P(w_i|t_i)$

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

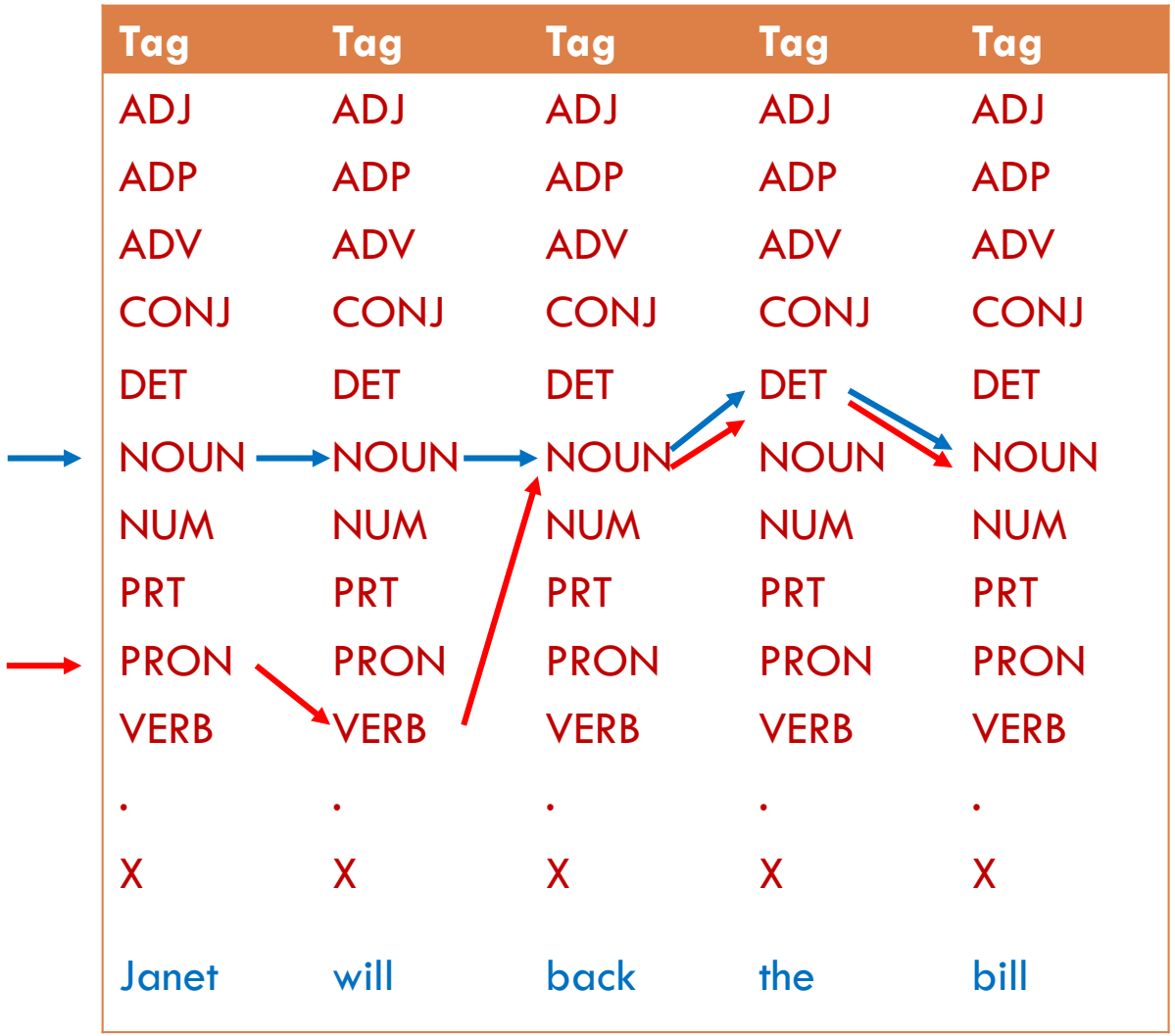
- To find the best tag sequence, we could – in principle – calculate this for all possible tag sequences and choose the one with highest score
  - $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(w_1^n|t_1^n)P(t_1^n)$
- Impossible in practice – There are too many

Possible tag sequences



- The number of possible tag sequences =
- The number of paths through the trellis =
- m^n
  - m is the number of tags in the set
  - n is the number of tokens in the sentence
  - Here: $12^5 \approx 250{,}000$.

Viterbi algorithm (dynamic programming)



- Walk through the word sequence
- For each word keep track of
  - all the possible tag sequences up to this word and the probability of each sequence
- If two paths are equal from a point on, then
- The one scoring best at this point will also score best at the end
- Discard the other one

Viterbi algorithm

- A nice example of dynamic programming
- Skip the details:
  - Viterbi is covered in IN2110
  - We will use preprogrammed tools in this course – not implement ourselves
  - HMM is not state of the art taggers

HMM trigram tagger

- Take two preceding tags into consideration
- $P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}, t_{i-2})$
-

$$P(w_1^n, t_1^n) = \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}, t_{i-2})$$

- Add two initial special states and one special end state

Challenges for the HMM-tagger

- Even with Viterbi, it is expensive:
  - For the trigram, the size of the trellis: $(n+2) \times m^3$
    - n words in the sequence
    - m tags in the model
  - Example, 6 words
    - 12 tags: 15,552
    - With 45 tags: 820,125
    - With 87 tags: 5,926,527
- We have probably not seen all tag trigrams during training:
  - We must use back-off or interpolation to lower n-grams
- Words not observed during training:
  - How can we include e.g. morphological features?
    - e.g., *-ly* ➔ Adv

Today

- N-gram language models
  - (hangover from last week)
- POS-tagging
- HMM-tagging
- Discriminative tagging
- Named-entity recognition (NER)
- Evaluation of NER

Discriminative tagging

Notation:
$t_1^n = t_1, t_2, \ldots t_n$

- The goal of tagging is to decide: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$
- HMM is generative.
  - It estimates $P(w_1^n | t_1^n) P(t_1^n) = P(w_1^n, t_1^n)$
- As for text classification, we could instead use a discriminative procedure and try to estimate the tag sequence directly
- $P(t_1^n | w_1^n) = P(t_1 | w_1^n) P(t_2 | t_1, w_1^n) \ldots P\left(t_i | t_1^{i-1}, w_1^n\right) \ldots = \prod_{i=1}^n P\left(t_i | t_1^{i-1}, w_1^n\right)$

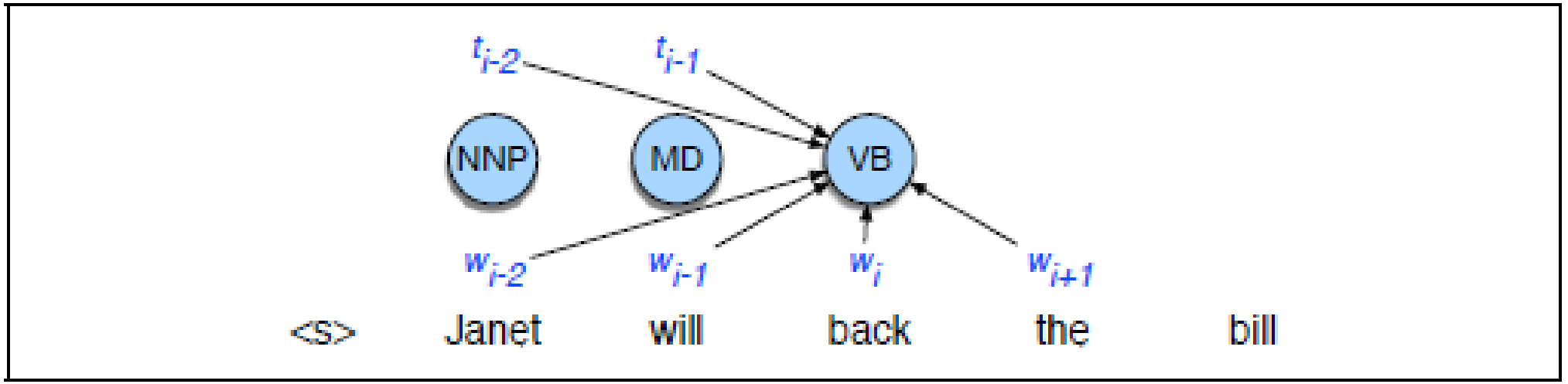


Figure 8.13 An MEMM for part-of-speech tagging showing the ability to condition on more features.

- Apply the chain rules for probabilities
  - $P(t_1^n|w_1^n) = P(t_1|w_1^n)P(t_2|t_1, w_1^n) \dots P\left(t_i|t_1^{i-1}, w_1^n\right) \dots = \prod_{i=1}^{n} P\left(t_i|t_1^{i-1}, w_1^n\right)$
  - $\underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n|w_1^n) = \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i|t_1^{i-1}, w_1^n\right)$
- Simplifying assumptions:
  - $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i|\, t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$

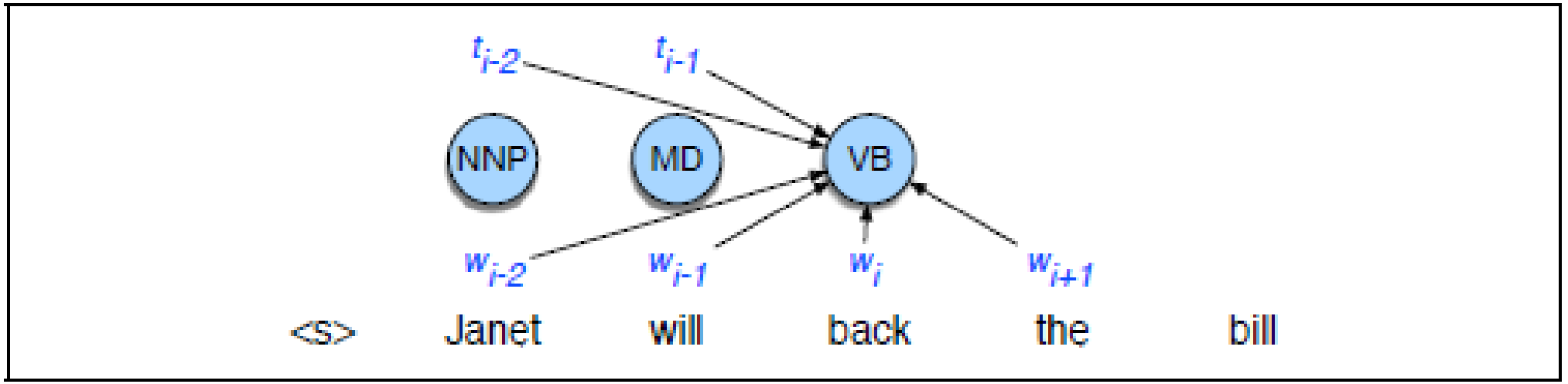


Figure 8.13 An MEMM for part-of-speech tagging showing the ability to condition on more features.

- Simplifying assumptions:
  - $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i | \, t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$
  - The tag t_i only depends on
    - k preceding tags, typically k=1 or k=2
    - the words in a window around w_i of size m

Feature extraction

- $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i | t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$

- We use a template to extract features from preceding tag(s) and neighboring words.
- The actual number of features may be large.
- Observe that properties may be combined into one feature

$t_i = \text{VB}$ and $w_{i-2} = \text{Janet}$
$t_i = \text{VB}$ and $w_{i-1} = \text{will}$
$t_i = \text{VB}$ and $w_i = \text{back}$
$t_i = \text{VB}$ and $w_{i+1} = \text{the}$
$t_i = \text{VB}$ and $w_{i+2} = \text{bill}$
$t_i = \text{VB}$ and $t_{i-1} = \text{MD}$
$t_i = \text{VB}$ and $t_{i-1} = \text{MD}$ and $t_{i-2} = \text{NNP}$
$t_i = \text{VB}$ and $w_i = \text{back}$ and $w_{i+1} = \text{the}$

Remarks

- $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i | \, t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$

- The extracted features corresponds to J&M's "small features", $f_k(y_{i-1}, y_i, X, i)$
- J&M includes the tag into the feature:
  - There are alternative ways of presenting this
  - We do not have to include the class

$t_i = \text{VB and } w_{i-2} = \text{Janet}$
$t_i = \text{VB and } w_{i-1} = \text{will}$
$t_i = \text{VB and } w_i = \text{back}$
$t_i = \text{VB and } w_{i+1} = \text{the}$
$t_i = \text{VB and } w_{i+2} = \text{bill}$
$t_i = \text{VB and } t_{i-1} = \text{MD}$
$t_i = \text{VB and } t_{i-1} = \text{MD and } t_{i-2} = \text{NNP}$
$t_i = \text{VB and } w_i = \text{back and } w_{i+1} = \text{the}$

Features (for unknown words)

- We may include features which inspect properties of the word

w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i contains a number
w_i contains an upper-case letter
w_i contains a hyphen
w_i is all upper case
w_i's word shape
w_i's short word shape
w_i is upper case and has a digit and a dash (like *CFC-12*)
w_i is upper case and followed within 3 words by Co., Inc., etc.

Decoding

- Goal: $\underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) = \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i|t_1^{i-1}, w_1^n\right)$
- Simplest alternative: Greedy sequence decoding:
  - Choose the best tag for the first word in the sentence $\underset{t_1}{\operatorname{argmax}} P(t_1|w_1^n)$
  - Then choose the best tag for the second word in the sentence, given the choice for the first word, $\underset{t_2}{\operatorname{argmax}} P(t_2|t_1, w_1^n)$
  - And so on, tagging one word at a time, $\underset{t_i}{\operatorname{argmax}} P(t_i|t_{i-1}, w_1^n)$

    until we have finished the sentence.

Training a greedy classifier

- The training examples are extracted from a tagged corpus
- For each word occurrence in the corpus, one training example:
  - The class is the correct tag
  - The features are extracted from the context
- One can then
  - train any multi-class ML-algorithm, e.g., multinomial logistic regression
  - apply greedy tagging on untagged text.

Shortcomings of greedy decoding

- Early decisions, considers only one tag at a time
  - If $\hat{t}_1 = \underset{t_1}{\operatorname{argmax}}\, P(t_1|w_1^n)$ and $\hat{t}_2 = \underset{t_1}{\operatorname{argmax}}\, P(\hat{t}_1|w_1^n)$ then

    $\underset{t_1 t_2}{\operatorname{argmax}}\, P(t_1 t_2|w_1^n)$ does not have to equal $\hat{t}_1 \hat{t}_2$

- Compare to HMM which considers whole tag sequences and choose the most probable sequence.

Maximum Entropy Markov Models (MEMM)

- If the model uses a limited history,
  - $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i |\, t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$

    one may use a form of Viterbi decoding
- We may then find $\underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i |\, t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$
- This should make better result

However

- The greedy sequence decoding does surprisingly well
- And equally surprising: using preceding tags as features does not improve the tagger that much compared to not including them.
- See mandatory assignment 2

Conditional Random Fields

- Even if we use Viterbi decoding and find the most probable overall tag sequence, we so far trained or model on the greedy task.
- What we ideally should have done was to also train the model on the task of predicting the optimal whole sequence
- Conditional Random Fields (CRFs) is a generalization compared to MEMM:
  - Makes it possible to optimize training for whole tag sequences
  - Slow in training
  - Considered the best tool for sequence labelling until a few years ago
- Currently, neural networks ("deep learning") are considered the best tool

	Generative	Discriminative
Classification	Naive Bayes	Logistic regression
Sequence labeling	HMM	CRF

Today

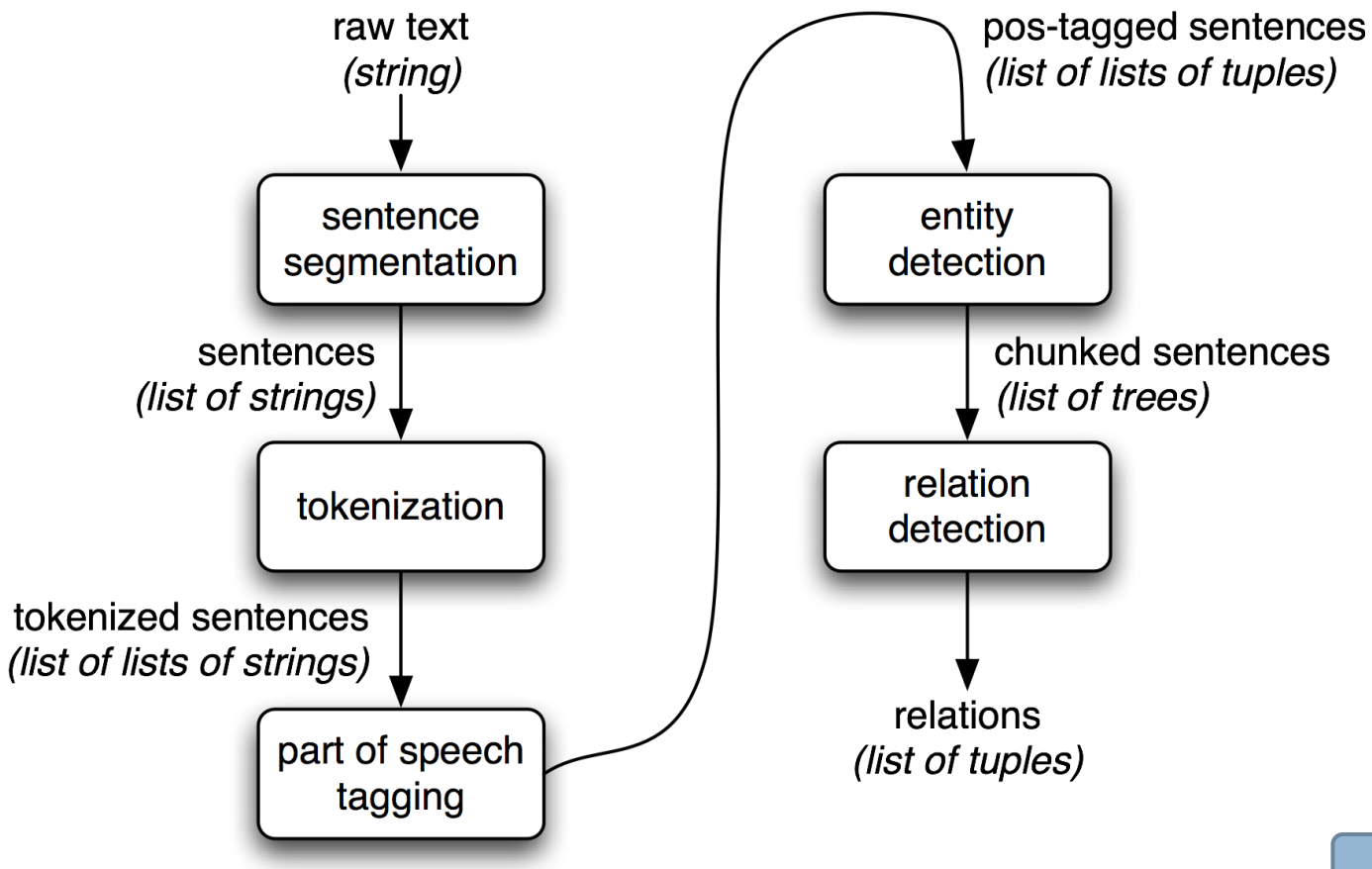
- N-gram language models
  - (hangover from last week)
- POS-tagging
- HMM-tagging
- Discriminative tagging
- Named-entity recognition (NER)
- Evaluation of NER

IE basics

Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. (Wikipedia)

- Bottom-Up approach
- Start with unrestricted texts, and do the best you can
- The approach was in particular developed by the Message Understanding Conferences (MUC) in the 1990s
- Select a particular domain and task

A typical pipeline



From NLTK

Some example systems

- Stanford core nlp (Java): http://corenlp.run/
- Stanza (Python): https://stanfordnlp.github.io/stanza/
  - with a wrapper for Stanford Core NLP

- SpaCy (Python): https://spacy.io/docs/api/

More systems

- OpenNLP (Java): https://opennlp.apache.org/docs/
- GATE (Java): https://gate.ac.uk/
  - https://cloud.gate.ac.uk/shopfront
- UDPipe: http://ufal.mff.cuni.cz/udpipe
  - Online demo: http://lindat.mff.cuni.cz/services/udpipe/
- Collection of tools for NER:
  - https://www.clarin.eu/resource-families/tools-named-entity-recognition

Named entities

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].

- Named entity:
  - Anything you can refer to by a proper name
- NE Recognition
  - Find the phrases
  - Classify them

Types of NE

Type	Tag	Sample Categories
People	PER	Individuals, fictional characters, small groups
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams
Location	LOC	Physical extents, mountains, lakes, seas
Geo-Political Entity	GPE	Countries, states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicles	VEH	Planes, trains, and automobiles

- The set of types vary between different systems
- Which classes are useful depend on application:
  - The first 4 above are most common
  - The last two are more for particular applications
  - Others: TIME, MONEY,

Ambiguities

Name	Possible Categories
Washington	Person, Location, Political Entity, Organization, Facility
Downing St.	Location, Organization
IRA	Person, Organization, Monetary Instrument
Louis Vuitton	Person, Organization, Commercial Product

[$_{PERS}$ 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.
The [$_{FAC}$ Washington] had proved to be a leaky ship, every passage I made...

BIO Labels (IOB)

- B-PER:
  - First word in this PER-NE
- I-NP:
  - Part of PER-NE
- O:
  - Not part of any NE

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

- Can code where something begins and ends without altering the word sequence
- Applying "CONNL-format"
  - one word per line
  - info in columns
  - we may add more columns, e.g. for POS-tag

Alternatives

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

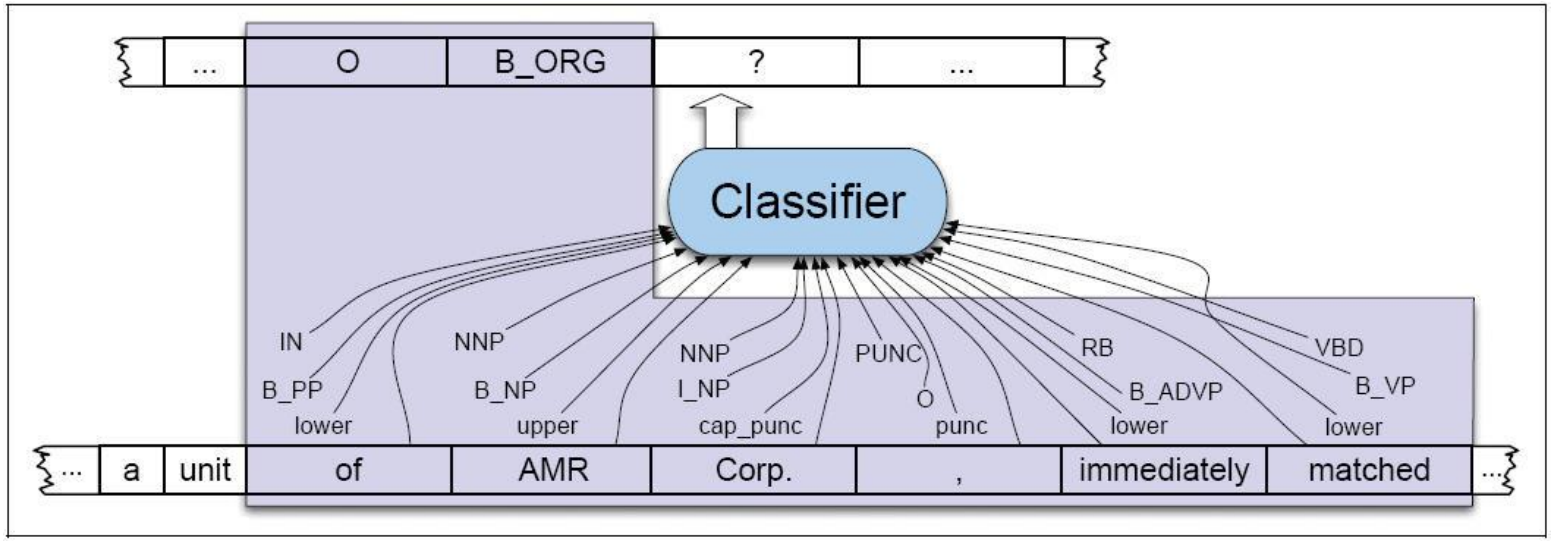
Figure 8.7 NER as a sequence model, showing IO, BIO, and BIOES taggings.

Named Entity Recognition

Methods (alternatives):

- Hand-written rules
- Regular expressions
  - NLTK demonstrates this for NP-chunking
- Supervised machine learning
  - Feature-based discriminative sequence labelling
    - similarly to (CRF) POS-tagging
  - Neural sequence labelling

Feature-based NE sequence labeling



- Similar to tagging and chunking
- You will need features from several layers
- Features may include
  - Words, POS-tags, Chunk-tags, Graphical prop.
  - and more (See J&M, 3.ed)

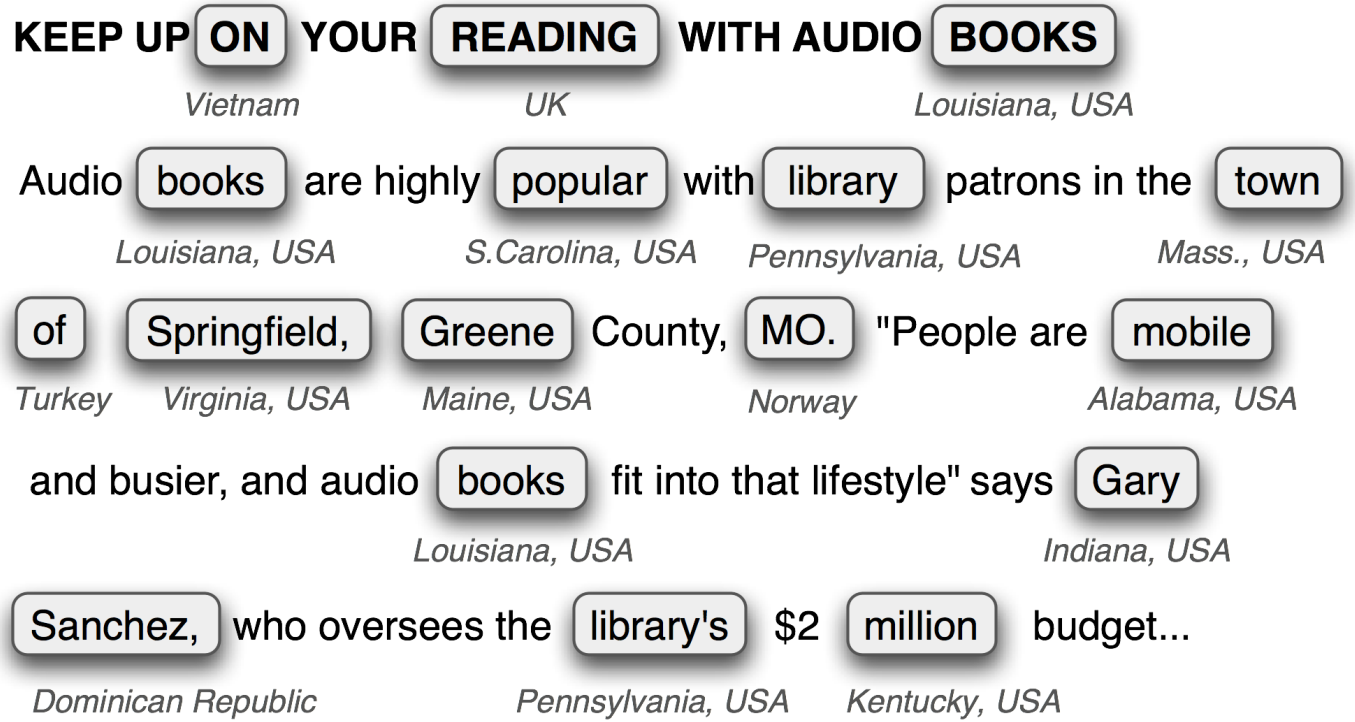
Features

identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
gazetteer features

Figure 8.15 Typical features for a feature-based NER system.

Gazetteer

- Useful: List of names, e.g.
  - Gazetteer: list of geographical names
- But does not remove all ambiguities
  - cf. example



Today

- N-gram language models
  - (hangover from last week)
- POS-tagging
- HMM-tagging
- Discriminative tagging
- Named-entity recognition (NER)
- Evaluation of NER

Evaluating NE Recognition

- Have we found the correct named entities?
  - The correct beginning and end of the named entity?
  - The right label?
- We might evaluate the BIO-tags, but that is not what we are looking for
- Observe that since the number of predicted NEs may be different from the number of gold NEs, we should use Precision and Recall.

Evaluation measures

		Is in C	
		Yes	NO
Classifier	Yes	tp	fp
	No	fn	tn

- Accuracy: (tp+tn)/N
- Precision: tp/(tp+fp)
- Recall: tp/(tp+fn)

- F-score combines P and R
- $F_1 = \frac{2PR}{P+R} \left(= \frac{1}{\frac{\frac{1}{R}+\frac{1}{P}}{2}} \right)$
- F_1 called "harmonic mean"
- General form
  - $F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}}$
  - for some $0 < \alpha < 1$

Confusion matrix

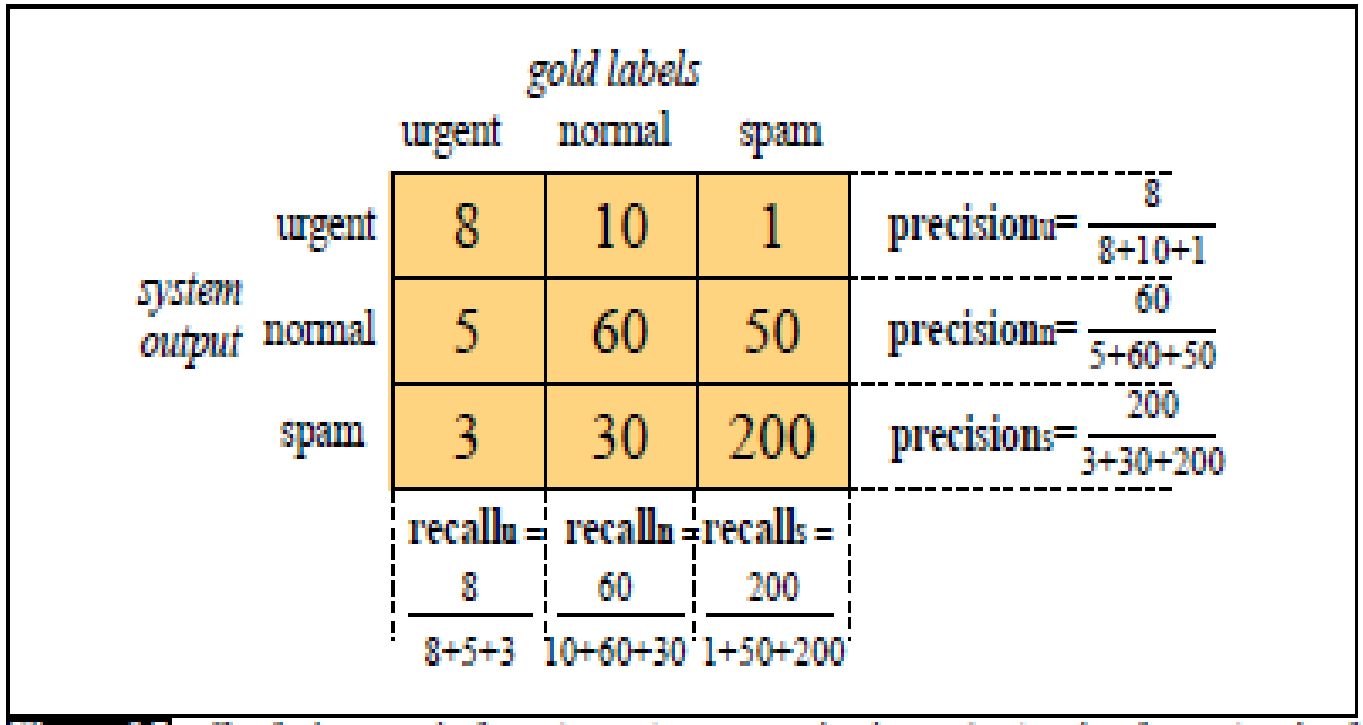


Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

- Precision, recall and f-score can be calculated for each class against the rest
- Examples:
  - For each tag for a POS-tagger
  - For each entity type for NE Recognizer

Tag accuracy

In	IN	O	O
addition	NN	B-NP	B-NP
to	TO	O	O
his	PRP$	B-NP	B-NP
previous	JJ	I-NP	I-NP
real-estate	NN	I-NP	I-NP
investment	NN	I-NP	I-NP
and	CC	I-NP	I-NP
asset-management	NN	I-NP	I-NP
duties	NNS	I-NP	I-NP
,	,	O	O
Mr.	NNP	B-NP	B-NP
Meador	NNP	I-NP	I-NP
takes	VBZ	O	O
responsibility	NN	B-NP	B-NP
for	IN	O	O
development	NN	B-NP	B-NP
and	CC	O	I-NP
property	NN	B-NP	I-NP
management	NN	I-NP	I-NP
.	.	O	O

- 2 out of 21 tags are incorrect
- Tag-accuracy: 19/21

In	IN	O	O
addition	NN	B-NP	B-NP
to	TO	O	O
his	PRP$	B-NP	B-NP
previous	JJ	I-NP	I-NP
real-estate	NN	I-NP	I-NP
investment	NN	I-NP	I-NP
and	CC	I-NP	I-NP
asset-management	NN	I-NP	I-NP
duties	NNS	I-NP	I-NP
,	,	O	O
Mr.	NNP	B-NP	B-NP
Meador	NNP	I-NP	I-NP
takes	VBZ	O	O
responsibility	NN	B-NP	B-NP
for	IN	O	O
development	NN	B-NP	B-NP
and	CC	O	I-NP
property	NN	B-NP	I-NP
management	NN	I-NP	I-NP
.	.	O	O

Counting chunks

- Left column: Gold
- Right column: Predicted

Precision: ?

Recall: ?