BITS Pilani

Pilani | Dubai | Goa | Hyderabad

S1-25_DSECLZG530/SSCLZG599

Natural Language Processing

(Lecture #6 –Parts of Speech)

- *The slides presented here are obtained from the authors of the books and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified a few slides to suit the requirements of the course.*

Text Books

	Author(s), Title, Edition, Publishing House
T1	Speech and Language processing: An introduction to Natural Language Processing, Computational Linguistics and speech Recognition by Daniel Jurafsky and James H. Martin[3rd edition]
T2	Foundations of statistical Natural language processing by Christopher D.Manning and Hinrich schutze

innovate achieve lead

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

POS Basics

Parts of Speech

- From the earliest linguistic traditions (Yaska and Panini 5th C. BCE, Aristotle 4th C. BCE), the idea that words can be classified into grammatical categories

  part of speech, word classes, POS, POS tags

- 8 parts of speech attributed to Dionysius Thrax of Alexandria (c. 1st C. BCE):

  noun, verb, pronoun, preposition, adverb, conjunction, participle, article

- These categories are relevant for NLP today.

POS examples

- N noun *chair, bandwidth, pacing*
- V verb *study, debate, munch*
- ADJ adjective *purple, tall, ridiculous*
- ADV adverb *unfortunately, slowly*
- P preposition *of, by, to*
- PRO pronoun *I, me, mine*
- DET determiner *the, a, that, those*

Two classes of words: Open vs. Closed

Closed class words

- Relatively fixed membership
- Usually **function** words: short, frequent words with grammatical function
  - determiners: ***a, an, the***
  - pronouns: ***she, he, I***
  - prepositions: ***on, under, over, near, by, …***

Open class words

- Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
  - Plus interjections: **oh, ouch, uh-huh, yes, hello**
- New nouns and verbs like *iPhone* or *to fax*

"Universal Dependencies" 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

Nivre et al. 2016

Major open class - Nouns

Four major open classes occur in the languages of the world: nouns (including proper nouns), verbs, adjectives, and adverbs, as well as the smaller open class of interjections.

- English has all five, although not every language does.

Nouns are words for people, places, or things, but include others as well. Common nouns include

- concrete terms like cat and mango,
- abstractions like algorithm and beauty, and
- verb-like terms like pacing as in His pacing to and fro became quite annoying.

Nouns in English can occur with determiners (a goat, its bandwidth), take possessives (IBM's annual revenue), and may occur in the plural (goats, abaci).

Major open class - Nouns

Many languages, including English, divide common nouns into count nouns and mass nouns.

- Count nouns can occur in the singular and plural (goat/goats, relationship/relationships) and can be counted (one goat, two goats).
- Mass nouns are used when something is conceptualized as a homogeneous group. So snow, salt, and communism are not counted
- Proper nouns, like Regina, Colorado, and IBM, are names of specific persons or entities

Major open class - Verbs

Verbs refer to actions and processes, including main verbs like draw, provide, and go.

English verbs have

- inflections (non-third-person-singular (eat),
- third-person-singular (eats),
- progressive (eating),
- past participle (eaten)

Most human languages have the categories of noun and verb

- Few languages, such as Riau Indonesian and Tongan, don't make this distinction

Major open class - Adjectives

Adjectives often describe properties or qualities of nouns, like

- color (white, black),
- age (old, young), and
- value (good, bad) etc.

There are few languages without adjectives. In Korean, the words corresponding to English adjectives act as a subclass of verbs

- what is in English an adjective “beautiful” acts in Korean like a verb meaning “to be beautiful”

Major open class - Adverbs

Adverbs are a hodge-podge. All the italicized words in this example are adverbs

Actually, I ran *home extremely quickly yesterday*

Adverbs generally modify something (often verbs, hence the name "adverb", but also other adverbs and entire verb phrases).

- Directional adverbs or locative adverbs (home, here, downhill) specify the direction or location of some action;
- Degree adverbs (extremely, very, somewhat) specify the extent of some action, process, or property;
- Manner adverbs (slowly, slinkily, delicately) describe the manner of some action or process; and
- Temporal adverbs describe the time that some action or event took place (yesterday, Monday).

Part-of-Speech Tagging

Assigning a part-of-speech to each word in a text.

Words often have more than one POS.

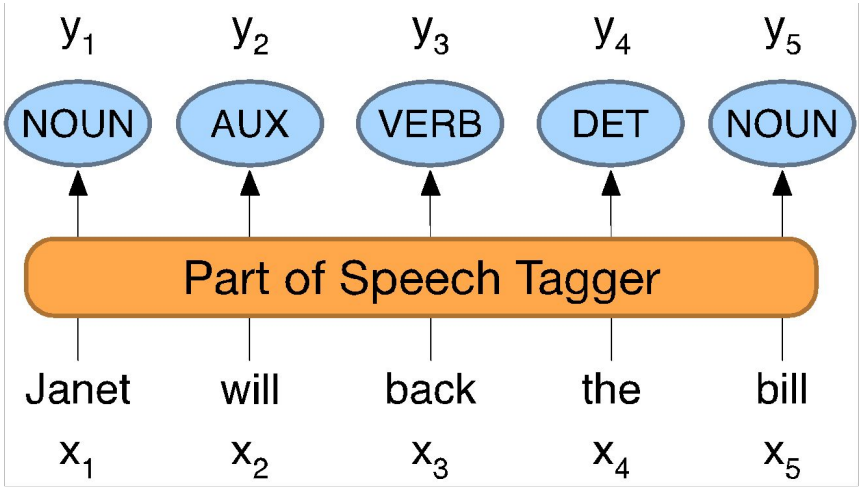
book:

VERB: (***Book*** *that flight*)

NOUN: (*Hand me that* ***book***).

Part-of-Speech Tagging

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



Treebank

A treebank is a parsed text corpus that annotates syntactic or semantic sentence structure.

- Treebanks are often created on top of a corpus that has already been annotated with part-of-speech tags.
- Treebanks are sometimes enhanced with semantic or other linguistic information

Treebanks have been used to engineer state-of-the-art NLP systems such as

- part-of-speech taggers,
- parsers,
- semantic analyzers and
- machine translation system

Penn Treebank

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*

Other tags are #, $, ., ,, :, (,), ", '.

Sample "Tagged" English sentences

There/PRO were/VERB 70/NUM children/NOUN there/ADV ./PUNC

Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN 's/PART New/PROPN England/PROPN Journal/PROPN of/ADP Medicine/PROPN

Tagged according to both the UD and Penn tagsets

There/PRO/EX are/VERB/VBP 70/NUM/CD children/NOUN/NNS there/ADV/RB ./PUNC/.

Preliminary/ADJ/JJ findings/NOUN/NNS were/AUX/VBD reported/VERB/VBN in/ADP/IN today/NOUN/NN 's/PART/POS New/PROPN/NNP England/PROPN/NNP Journal/PROPN/NNP of/ADP/IN Medicine/PROPN/NNP

Why Part of Speech Tagging?

- Can be useful for other NLP tasks
  - Parsing: POS tagging can improve syntactic parsing
  - MT: reordering of adjectives and nouns (say from Spanish to English)
  - Sentiment or affective tasks: may want to distinguish adjectives or other POS
  - Text-to-speech (how do we pronounce “lead” or "object"?)

- Or linguistic or language-analytic computational tasks
  - Need to control for POS when studying linguistic change like creation of new words, or meaning shift
  - Or control for POS in measuring meaning similarity or difference

How difficult is POS tagging in English?

Roughly 15% of word types are ambiguous

Hence 85% of word types are unambiguous

Janet is always PROPN, *hesitantly* is always ADV

But those 15% tend to be very common.

So ~60% of word tokens are ambiguous

E.g., *back*

earnings growth took a back/ADJ seat
a small building in the back/NOUN
a clear majority of senators back/VERB the bill
enable the country to buy back/PART debt
I was twenty-one back/ADV then

POS tagging performance in English

How many tags are correct? (Tag accuracy)

- About 97%
  - Hasn't changed in the last 10+ years
  - HMMs, CRFs, BERT perform similarly .
  - Human accuracy about the same

But baseline is 92%!

- Baseline is performance of stupidest possible method
  - "Most frequent class baseline" is an important baseline for many tasks
    - Tag every word with its most frequent tag
    - (and tag unknown words as nouns)
- Partly easy because
  - Many words are unambiguous

Sources of information for POS tagging

Janet will back the bill

AUX/NOUN/VERB? NOUN/VERB?

Prior probabilities of word/tag

- "will" is usually an AUX

Identity of neighboring words

- "the" means the next word is probably not a verb

Morphology and word shape:

- Prefixes unable: un- → ADJ
- Suffixes importantly: -ly → ADJ
- Capitalization Janet: CAP → PROPN

Named Entity Recognition (NER)

Named Entities

Named entity, in its core usage, means anything that can be referred to with a proper name. Most common 4 tags:

- PER (Person): "Marie Curie"
- LOC (Location): "New York City"
- ORG (Organization): "Stanford University"
- GPE (Geo-Political Entity): "Boulder, Colorado"

– Often multi-word phrases

– But the term is also extended to things that aren't entities:
  - dates, times, prices

Named Entity tagging

The task of named entity recognition (NER):

find spans of text that constitute proper names

tag the type of the entity.

NER output

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**].

Why NER?

Sentiment analysis: consumer's sentiment toward a particular company or person?

Question Answering: answer questions about an entity?

Information Extraction: Extracting facts about entities from text.

Why NER is hard

1) Segmentation
   - In POS tagging, no segmentation problem since each word gets one tag.
   - In NER we have to find and segment the entities!
2) Type ambiguity

[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.

BIO Tagging

How can we turn this structured problem into a sequence problem like POS tagging, with one label per word?

[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago] route.

BIO Tagging

[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago] route.

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

B: token that *begins* a span

I: tokens *inside* a span

O: tokens outside of any span

Now we have one tag per token!!!

BIO Tagging variants: IO and BIOES

[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago] route.

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

Standard algorithms for NER

Supervised Machine Learning given a human-labeled training set of text annotated with tags

Hidden Markov Models

Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)

Neural sequence models (RNNs or Transformers)

Large Language Models (like BERT), finetuned

innovate achieve lead

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Rule Based Tagging

Rule Based Tagging

- The Brill tagger is an inductive method for part-of-speech tagging.
- It was described and invented by Eric Brill in his 1993 PhD thesis.
- It can be summarized as an error-driven transformation-based tagger

A Simple Rule-Based Part of Speech Tagger, Eric Brill Department of Computer Science, University of Pennsylvania

Brill tagger

- The tagger works by automatically recognizing and remedying its weaknesses, thereby incrementally improving its performance.
- The tagger initially tags by assigning each word its most likely tag, estimated by examining a large tagged corpus, without regard to context.
- Tagger uses 2 initial procedures
- One procedure is provided with information that words that were not in the training corpus and are capitalized tend to be proper nouns
- To tag words not seen in the training corpus by assigning such words the tag most common for words ending in the same three letters. For example, blahblahous would be tagged as an adjective

A Simple Rule-Based Part of Speech Tagger, Eric Brill Department of Computer Science, University of Pennsylvania

Brill tagger

- The initial tagger was trained on 90% of the corpus (the training corpus). 5% was held back to be used for the patch acquisition procedure (the patch corpus) and 5% for testing.
- Once the initial tagger is trained, it is used to tag the patch corpus. A list of tagging errors is compiled by comparing the output of the tagger to the correct tagging of the patch corpus.
- This list consists of triples < tag_a, tag_b, number >, indicating the number of times the tagger mistagged a word with tag_a when it should have been tagged with tag_b in the patch corpus.
- Next, for each error triple, it is determined which instantiation of a template from the prespecified set of patch templates results in the greatest error reduction.

A Simple Rule-Based Part of Speech Tagger, Eric Brill Department of Computer Science, University of Pennsylvania

Brill tagger

- The first ten patches found by the system are listed below.

(1) TO IN NEXT-TAG AT
(2) VBN VBD PREV-WORD-IS-CAP YES
(3) VBD VBN PREV-1-OR-2-OR-3-TAG HVD
(4) VB NN PREV-1-OR-2-TAG AT
(5) NN VB PREV-TAG TO
(6) TO IN NEXT-WORD-IS-CAP YES
(7) NN VB PREV-TAG MD
(8) PPS PPO NEXT-TAG.
(9) VBN VBD PREV-TAG PPS
(10) NP NN CURRENT-WORD-IS-CAP NO

AT = article, HVD = had,
IN = preposition,
MD = modal,
NN = sing. noun,
NP = proper noun,
PPS = 3rd sing. nora. pronoun,
PPO = obj. personal pronoun,
TO = infinitive to,
VB = verb,
VBN = past part. verb,
VBD = past verb

A Simple Rule-Based Part of Speech Tagger, Eric Brill Department of Computer Science, University of Pennsylvania

innovate achieve lead

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Markov Model

Markov chain

A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set.

(These sets can be words, or tags, or symbols representing anything, for example the weather.)

A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state.

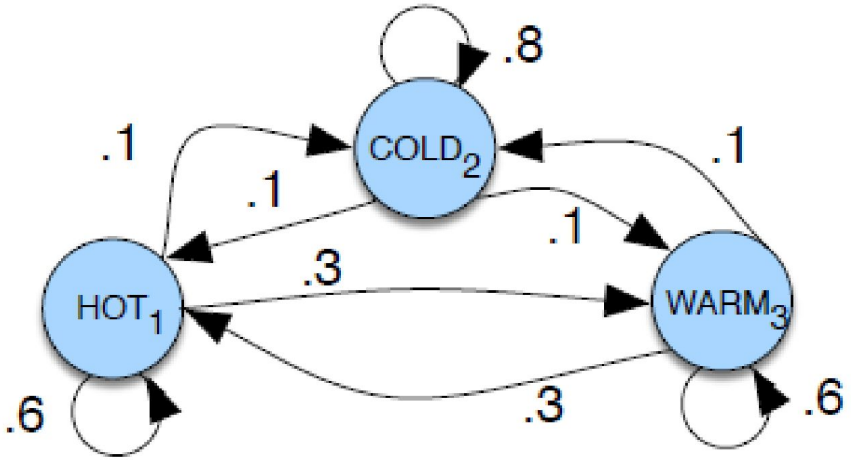
- All the states before the current state have no impact on the future except via the current state

Markov chain

Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

A Markov chain for weather (N=3)



The vocabulary consists of HOT, COLD, and WARM. The states are represented as nodes in the graph, and the transitions, with their probabilities, as edges. The transitions are probabilities: the values of arcs leaving a given state must sum to 1

A start distribution π is required; setting π = [0:1; 0:7; 0:2] would mean a probability 0.7 of starting in state 2 (cold), probability 0.1 of starting in state 1 (hot), etc.

A Markov chain for weather

Given π = [0:1; 0:7; 0:2] and transition probabilities as per previous slide, compute the probability of each of the following sequences:

a. hot hot hot hot

b. cold hot cold hot

a. P = 0.1*0.6**3

b. P = 0.7*0.1**3

innovate achieve lead

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Hidden Markov Model

Hidden Markov Model

A Markov chain is useful when we need to compute a probability for a sequence of observable events.

However, the events we are interested in may be hidden.

- We do not directly observe part-of-speech tags in text.
- Many ML tasks require that we infer the tags from the word sequence.
- There are many words in English that can be take multiple POS tags

Hidden Markov Model

- A Markov chain is useful when we need to compute a probability for a sequence of observable events.
  - We don't observe part-of-speech tags in a text. Rather, we see words and must infer the tags from the word sequence.
  - We call the tags hidden because they are not observed

A hidden Markov model (HMM) includes both

- observed events model (like words that we see in the input) and
- hidden events (like part-of-speech tags) that we think of as causal factors

in our probabilistic model.

Sequence Labeling Algorithm

Hidden Markov Model is a sequence labeling algorithm.

A sequence labeler is a model whose job is to assign a label to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels of the same length.

An HMM is a probabilistic sequence model: given a sequence of units (words, letters, morphemes, sentences, whatever), it computes a probability distribution over possible sequences of labels and chooses the best label sequence

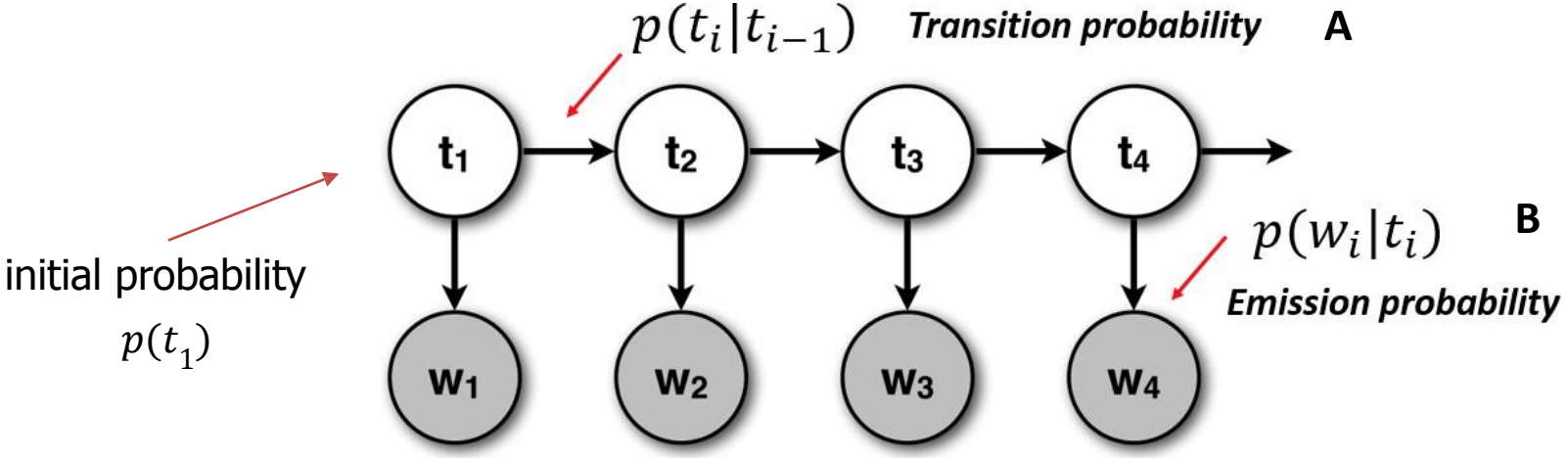
Hidden Markov Model

An HMM is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t (drawn from a vocabulary $V = v_1, v_2, ..., v_V$) being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Prediction in generative model

- Inference: What is the most likely sequence of tags for the given sequence of words **w**



- What are the latent states that most likely generate the sequence of word **w**

Hidden Markov Model

A first-order hidden Markov model instantiates two simplifying assumptions

- Probability of a particular state depends only on the previous state:

$$\textbf{Markov Assumption:} \quad P(q_i|q_1,...,q_{i-1}) = P(q_i|q_{i-1})$$

- Probability of an output observation oi depends only on the state that produced the observation qi and not on any other states or any other observations:

$$\textbf{Output Independence:} \quad P(o_i|q_1,\ldots q_i,\ldots,q_T,o_1,\ldots,o_i,\ldots,o_T) = P(o_i|q_i)$$

The components of an HMM tagger

An HMM has two components, the A and B probabilities

A matrix contains the tag transition probabilities which represent the probability of a tag occurring given the previous tag

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

In the WSJ corpus, for example, MD occurs 13124 times of which it is followed by VB 10471 times

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

The components of an HMM tagger

The B represent emission probabilities, given a tag (say MD), that it will be associated with a given word (say will). The MLE of the emission probability is

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

In the WSJ corpus, of the 13124 occurrences of MD, it is associated with 'will' 4046 times

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

The Emission Probability

This is the Bayesian modeling the likelihood term is not answering "which is the most likely tag for the word will?"

That would be the posterior P(MD|will).

Instead, P(will|MD) answers the question "If we were going to generate a MD, how likely is it that this word (modal verb) would be will?"

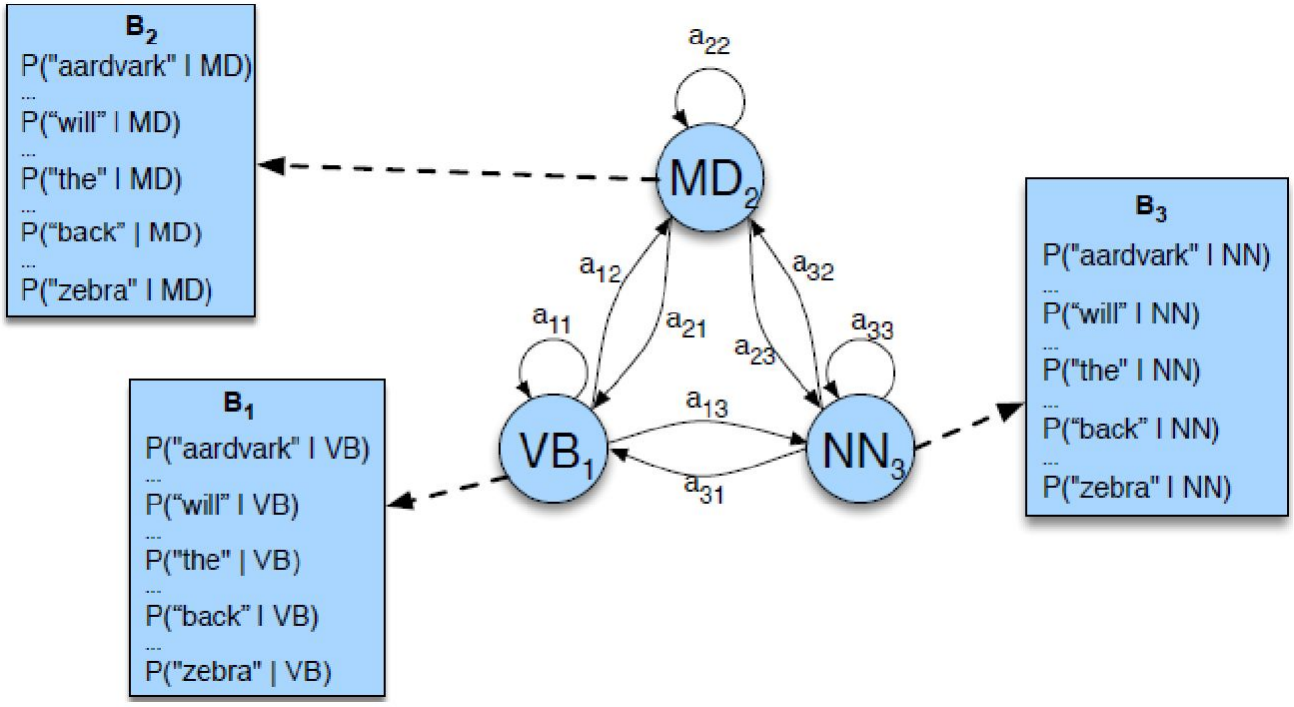
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

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

HMM representation

The A transition probabilities are used to compute the prior probability.

The B observation likelihoods that are associated with each state, give likelihood for each possible observation word



HMM Part-of-Speech Tagging

The goal of HMM decoding is to choose

the tag sequence $t_1 \ldots t_n$

that is most probable given the

observation sequence of n words $w_1 \ldots w_n$.

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n)$$

Using Bayes' rule we instead compute

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} \frac{P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)}$$

HMM Part-of-Speech Tagging

We can drop the denominator without impacting the argmax:

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)$$

HMM Part-of-Speech Tagging

HMM taggers make two further simplifying assumptions.

The probability of a word appearing depends only on its own tag and is independent of neighboring words and tags:

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

The second assumption (the bigram assumption), is that the probability of a tag is dependent only on the previous tag, rather than the entire tag sequence;

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

HMM Part-of-Speech Tagging

Plugging the simplifying assumptions from previous slide, we get the following equation for the most probable tag sequence from a bigram tagger

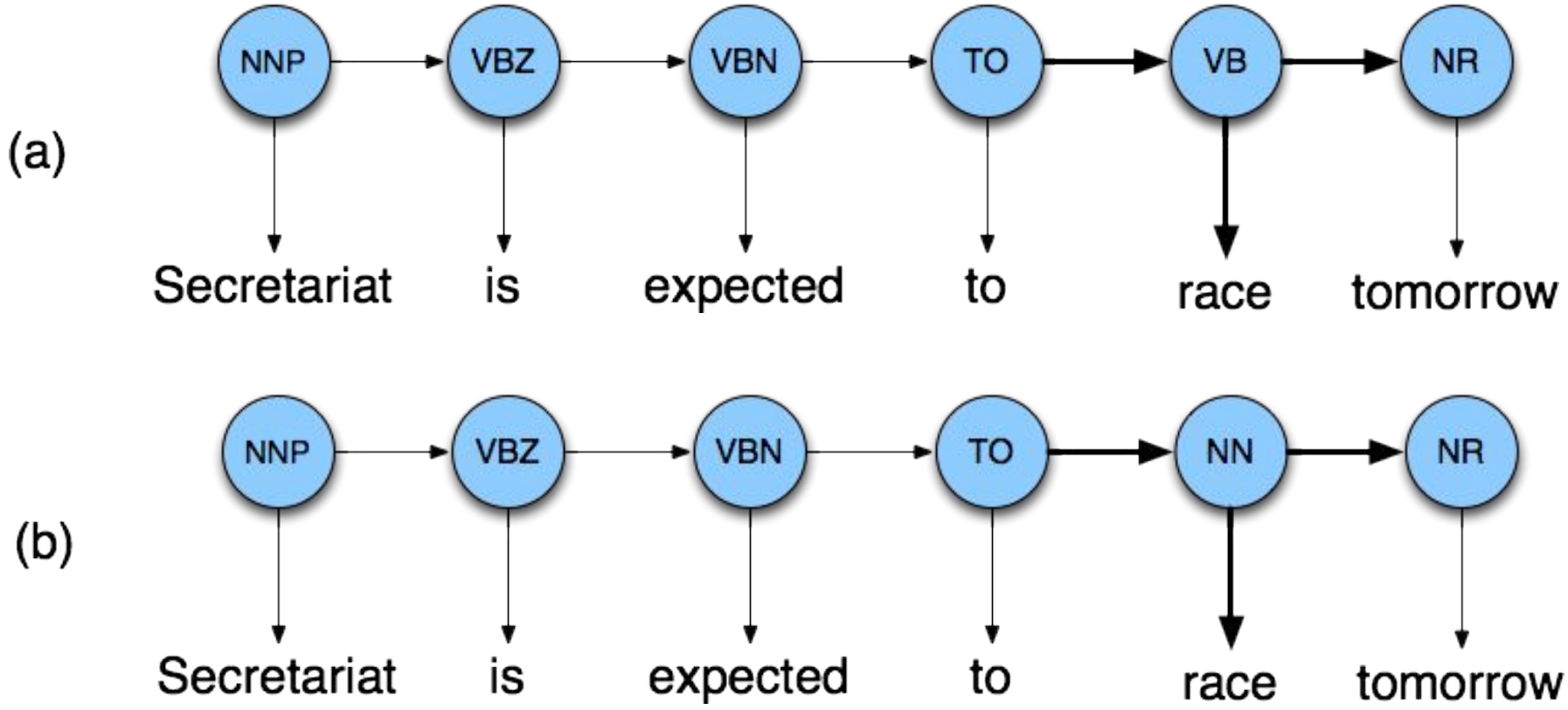
$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n) \approx \underset{t_1 \ldots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

The two parts correspond to the B emission probability and A transition probability.

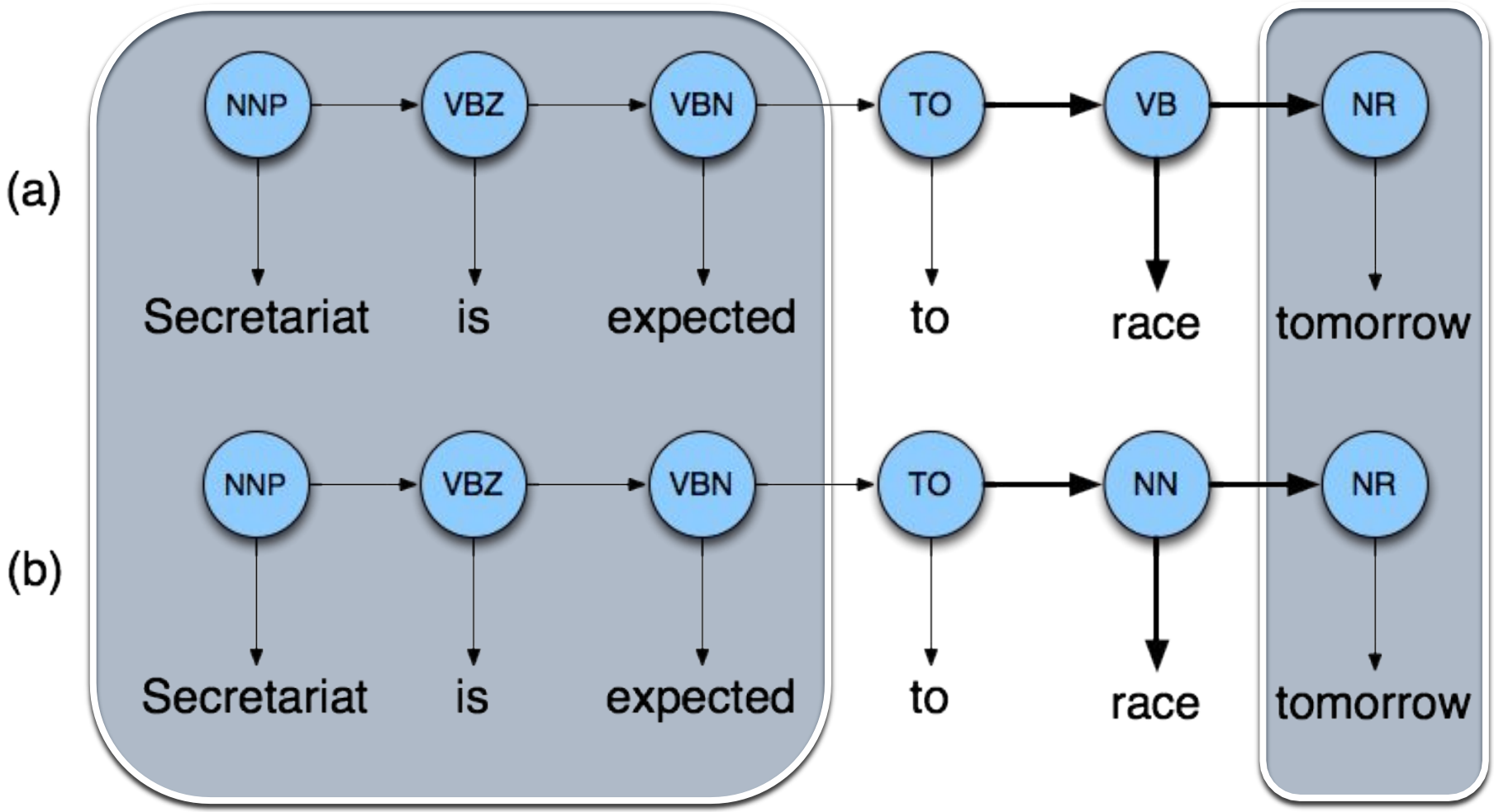
Example: The word "race"

- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

Disambiguating "race"

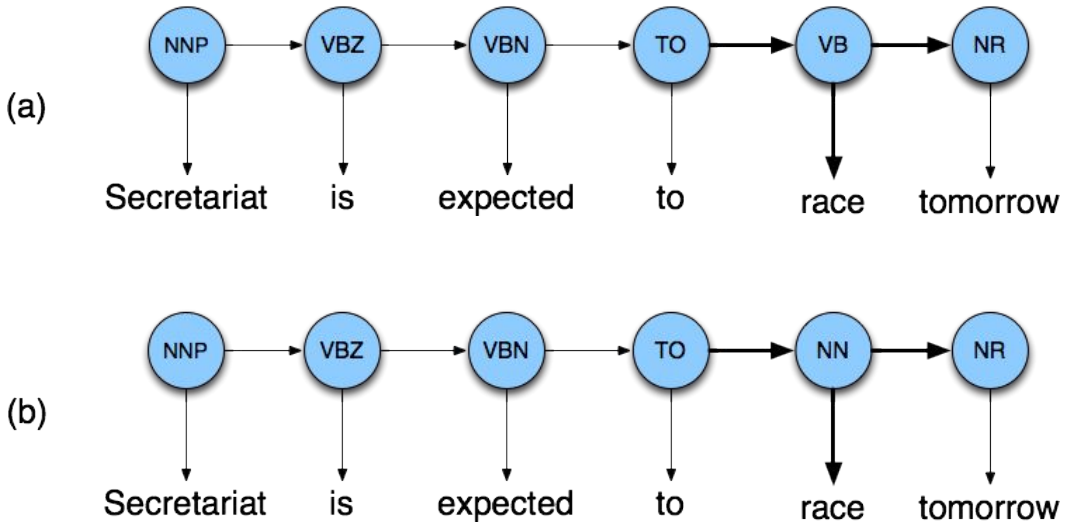


Disambiguating "race"



Example : NLTK POS tags

- P(NN|TO) = .00047
- P(VB|TO) = .83
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027
- P(NR|NN) = .0012



NR-Adverbial Noun (tomorrow), TO-proposition

- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb tag for “race”

https://www.guru99.com/pos-tagging-chunking-nltk.html

innovate achieve lead

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

The Viterbi Algorithm

The Viterbi Algorithm

The decoding algorithm for HMMs is the Viterbi algorithm

It is an instance of dynamic programming

Given an observation sequence and an HMM $\lambda = (A;B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

The Viterbi algorithm first sets up a probability matrix or lattice, with one column for each observation o_t and one row for each state in the state graph. Each column thus has a cell for each state q_i in the single combined automaton.

The Viterbi Algorithm

Each cell of the lattice, $v_t(j)$, represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence $q_1 \ldots q_{t-1}$.

The value of each cell $v_t(j)$ is computed by recursively taking the most probable path that could lead us to this cell.

Each cell expresses the probability.

$$v_t(j) = \max_{q_1,\ldots,q_{t-1}} P(q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j|\lambda)$$

Viterbi fills each cell recursively. Given that we had already computed the probability of being in every state at time t-1

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

The Viterbi Algorithm

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
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; 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*

Viterbi Algorithm for POS Tagging Example

Consider the sentence:

Janet will back the bill

We need an HMM with transition and observation probabilities

Viterbi Algorithm for POS Tagging Example

	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

The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus P(VB|MD) is 0.7968

Viterbi Algorithm for POS Tagging Example

	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

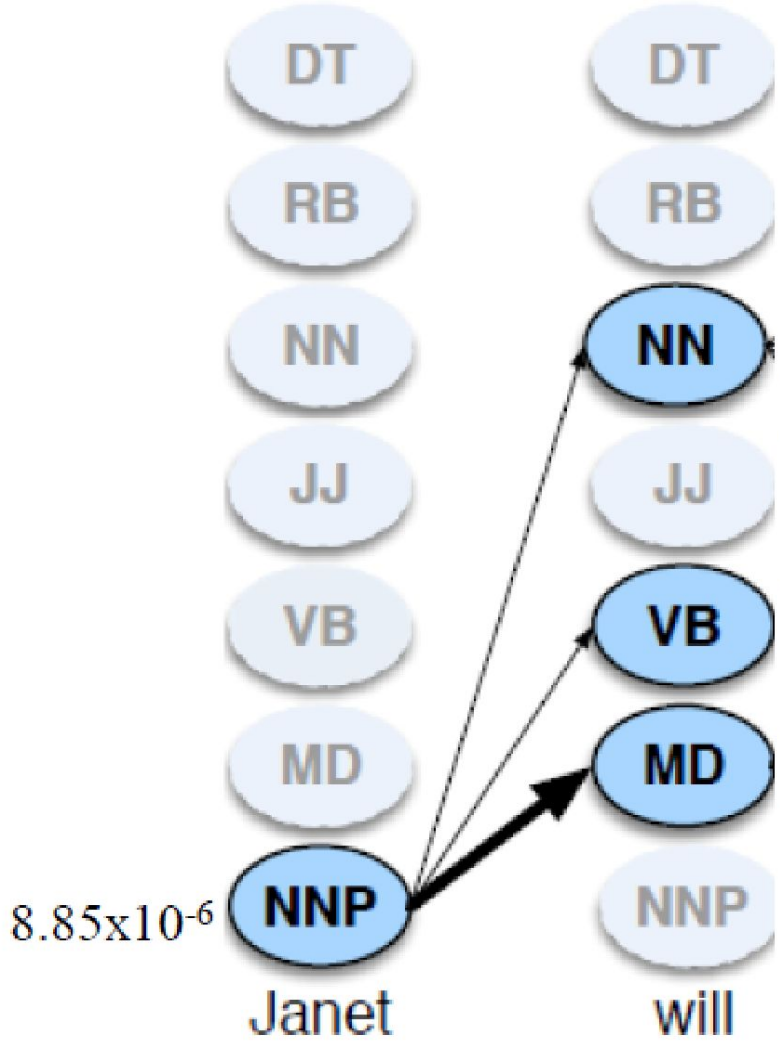
Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

Viterbi Algorithm for POS Tagging Example

Viterbi[NNP,Janet] =

P(NNP|<s>)*P(Janet|NNP) = 0.2767*0.000032 = 0.00000885 = 8.85×10^{-6}

Viterbi Algorithm for POS Tagging Example

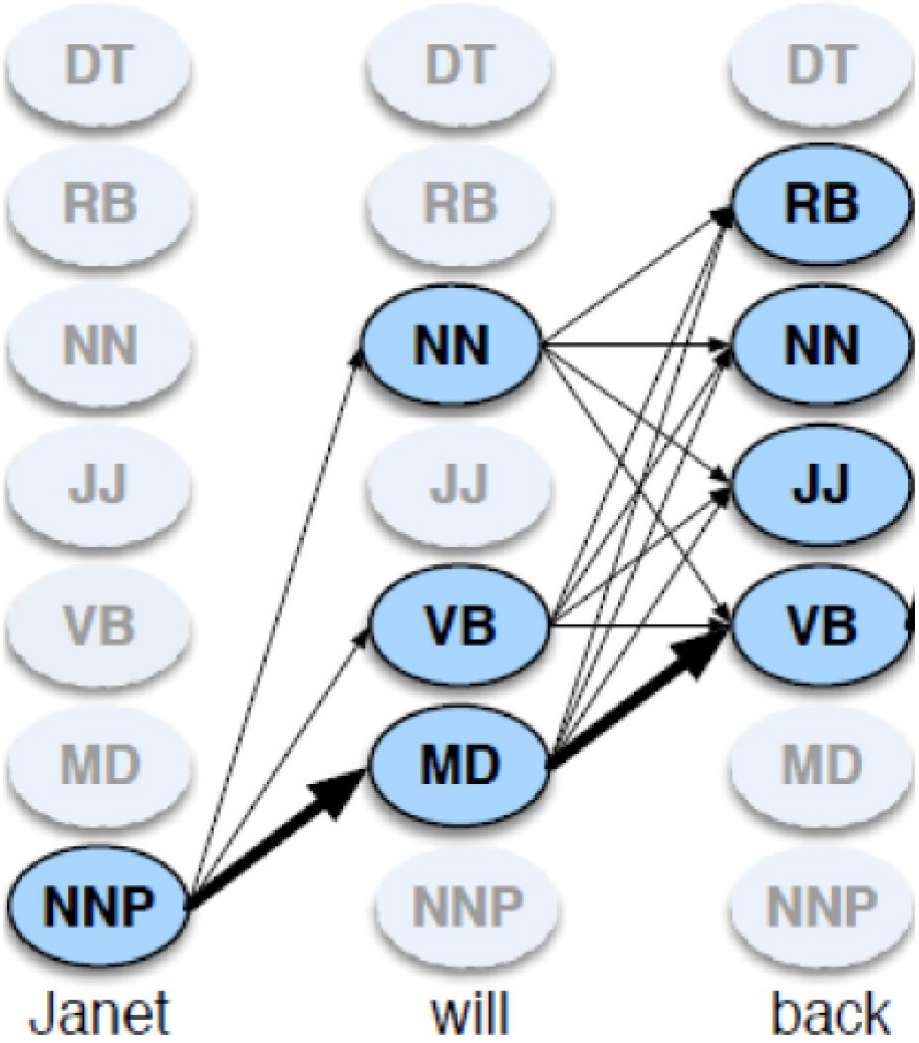


Viterbi[NN,will] = Viterbi[NNP,Janet]*P(NN|NNP)*P(will|NN)
= $8.85 \times 10^{-6} * 0.0584 * 0.0002 = 1.03368 \times 10^{-9}$

Viterbi[VB,will] = Viterbi[NNP,Janet]*P(VB|NNP)*P(will|VB)
= $8.85 \times 10^{-6} * 0.0009 * 0.000028 = 0.22302 \times 10^{-12}$

Viterbi[MD,will] = Viterbi[NNP,Janet]*P(MD|NNP)*P(will|MD)
= $8.85 \times 10^{-6} * 0.011 * 0.308431 = 0.203 \times 10^{-6}$

Viterbi Algorithm for POS Tagging Example



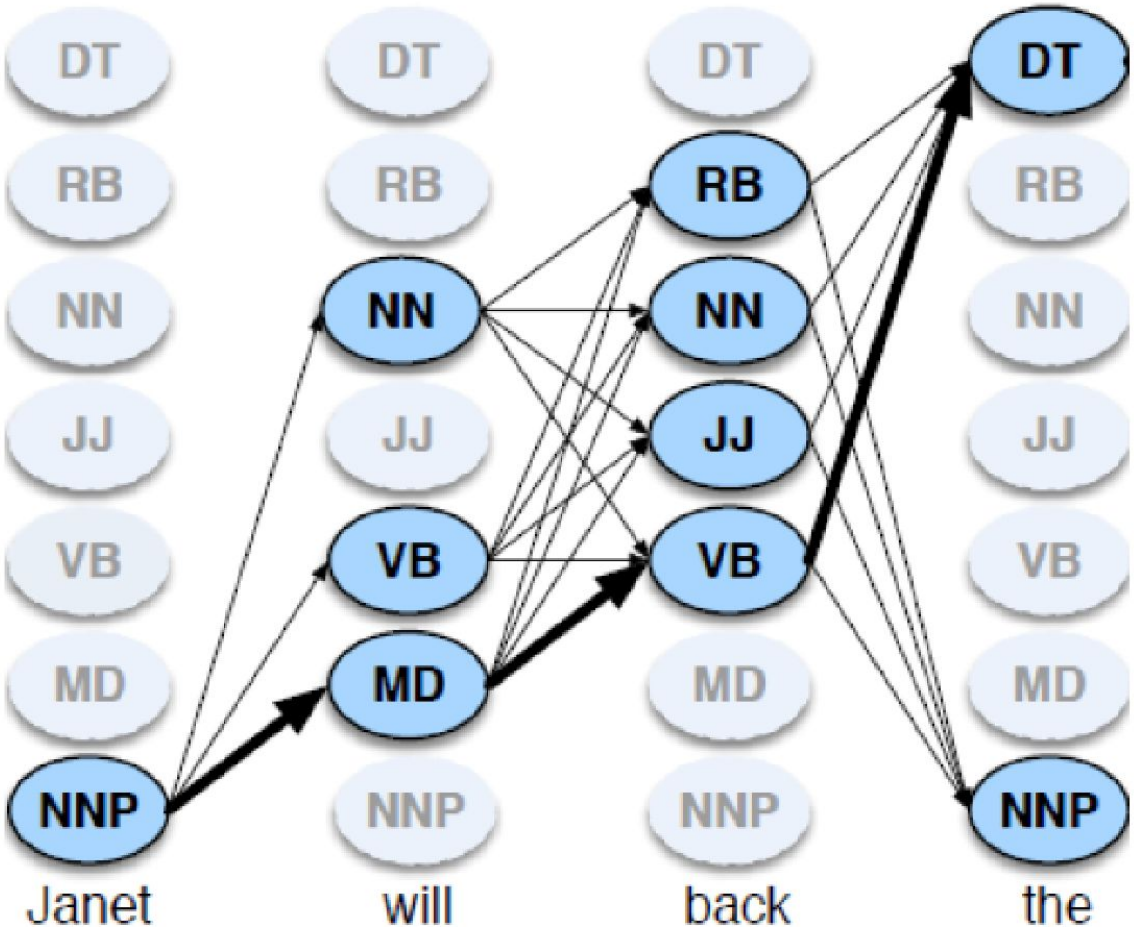
Viterbi[RB,back]=max({Viterbi[NN,will]*P(RB|NN)*P(back|RB),
Viterbi[VB,will]*P(RB|VB)*P(back|RB),
Viterbi[MD,will]*P(RB|MD)*P(back|RB)})

Viterbi[NN,back]=max({Viterbi[NN,will]*P(NN|NN)*P(back|NN),
Viterbi[VB,will]*P(NN|VB)*P(back|NN),
Viterbi[MD,will]*P(NN|MD)*P(back|NN)})

Viterbi[JJ,back]=max({Viterbi[NN,will]*P(JJ|NN)*P(back|JJ),
Viterbi[VB,will]*P(JJ|VB)*P(back|JJ),
Viterbi[MD,will]*P(JJ|MD)*P(back|JJ)})

Viterbi[VB,back]=max({Viterbi[NN,will]*P(VB|NN)*P(back|VB),
Viterbi[VB,will]*P(VB|VB)*P(back|VB),
Viterbi[MD,will]*P(VB|MD)*P(back|VB)})

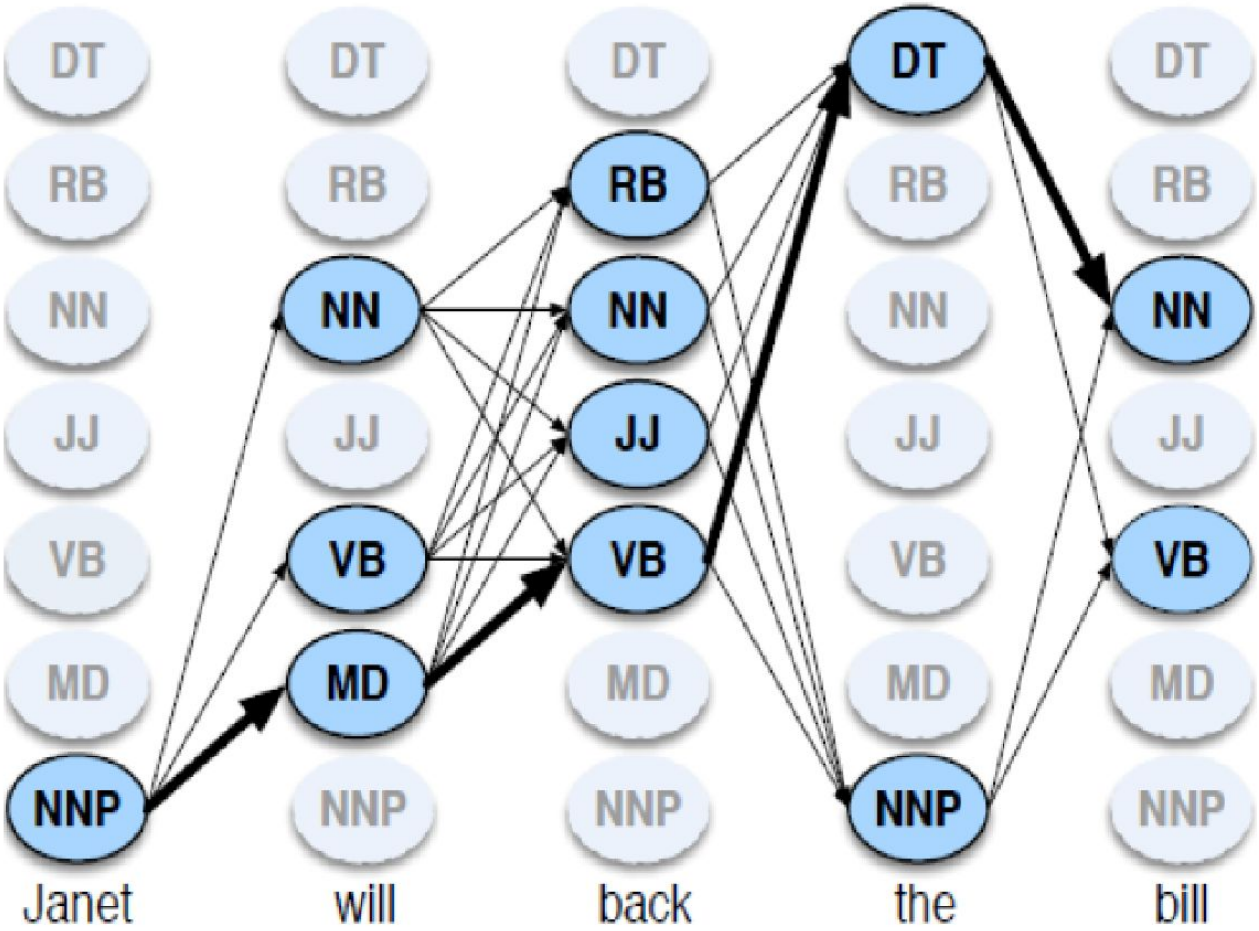
Viterbi Algorithm for POS Tagging Example



Viterbi[DT,the]=
max({Viterbi[RB,back]*P(DT|RB)*P(the|DT),
Viterbi[NN,back]*P(DT|NN)*P(the|DT),
Viterbi[JJ,back]*P(DT|JJ)*P(the|DT),
Viterbi[VB,back]*P(DT|VB)*P(the|DT)})

Viterbi[NNP,the]=
max({Viterbi[RB,back]*P(NNP|RB)*P(the|NNP),
Viterbi[NN,back]*P(NNP|NN)*P(the|NNP),
Viterbi[JJ,back]*P(NNP|JJ)*P(the|NNP),
Viterbi[VB,back]*P(NNP|VB)*P(the|NNP)})

Viterbi Algorithm for POS Tagging Example



Viterbi[NN,bill]=
max({Viterbi[DT,the]*P(NN|DT)*P(bill|NN),
Viterbi[NNP,the]*P(NN|NNP)*P(bill|NN)})

Viterbi[VB,bill]=
max({Viterbi[DT,the]*P(VB|DT)*P(bill|VB),
Viterbi[NNP,the]*P(VB|NNP)*P(bill|VB)})

Further with HMM

More on HMM

Hidden Markov models should be characterized by three fundamental problems

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

What we discussed is using HMM for problem 2.

HMM Training: The Forward-Backward Algorithm

Learning the parameters of an HMM, that is, the A and B matrices. This is necessary for decoding POS.

Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B

- The input to such a learning algorithm would be an unlabeled sequence of observations O and a vocabulary of potential hidden states Q
- The standard algorithm for HMM training is the forward-backward, or Baum-Welch algorithm (Baum, 1972), a special case of theExpectation-Maximization or EM algorithm

Conditional Random Fields (CRFs)

Conditional Random Fields (CRFs)

In POS tagging as in other tasks, we often run into unknown words:

- proper names and acronyms are created very often, and
- even new common nouns and verbs enter the language

We need ways to add arbitrary features to overcome this based on features e.g.

- capitalization or morphology (words starting with capital letters are likely to be proper nouns,
- words ending with -ed tend to be past tense (VBD or VBN), etc.)
- Or knowing the previous or following words might be a useful feature (if the previous word is the, the current tag is unlikely to be a verb)

Conditional Random Fields (CRFs)

CRF is discriminative sequence model based on log-linear models: on lines of logistic regression for single observation

Say, we have a sequence of input words.

$$X = x_1^n = x_1 \ldots x_n$$

and want to compute a sequence of output tags

$$Y = y_1^n = y_1 \ldots yn$$

In an HMM to compute the best tag sequence

$$\begin{aligned}
\hat{Y} &= \operatorname*{argmax}_Y p(Y|X) \\
&= \operatorname*{argmax}_Y p(X|Y)p(Y) \\
&= \operatorname*{argmax}_Y \prod_i p(x_i|y_i) \prod_i p(y_i|y_{i-1})
\end{aligned}$$

Conditional Random Fields (CRFs)

In a CRF, we compute the posterior p(Y|X) directly, training the CRF to discriminate among the possible tag sequences

$$\hat{Y} = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} P(Y|X)$$

- The CRF does not compute a probability for each tag at each time step.
- At each time step the CRF computes log-linear functions over a set of relevant features, and these local features are aggregated and normalized to produce a global probability for the whole sequence

Conditional Random Fields (CRFs)

CRF is like a giant version of multinomial logistic regression (for a single token). (the feature function f in regular multinomial logistic regression maps a tuple of a token x and a label y into a feature vector)

- CRF is for a sequence. Here, the function F maps an entire input sequence X and an entire output sequence Y to a feature vector.
- Let's assume we have K features, with a weight w_k for each feature F_k

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

Feature Functions

The K functions $F_k(X, Y)$ are global features, since each one is a property of the entire input sequence X and output sequence Y.

We compute them by decomposing into a sum of local features for each position i in Y:

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

Each of these local features f_k in a linear-chain CRF is allowed to make use of the current output token y_i, the previous output token y_{i-1}, the entire input string X (or any subpart of it), and the current position i.

Features in a CRF POS Tagger

Here are some example features

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

The reason to use a discriminative sequence model is that it's easier to incorporate a lot of features

For simplicity, assume all CRF features take on the value 1 or 0

For the example Janet/NNP will/MD back/VB the/DT bill/NN, when x_i is the word back, the following features would be generated with value of 1:

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

CRF with Unknown Words

One of the most important is word shape features, which represent the abstract letter pattern of the word by mapping

- lower-case letters to ‘x’, upper-case to ‘X’,
- numbers to ’d’, and retaining punctuation.

Thus for example I.M.F would map to X.X.X. and DC10-30 would map to XXdd-dd.

A second class of shorter word shape features is also used.

- In these features consecutive character types are removed, so words in all caps map to X, words with initial-caps map to Xx.

DC10-30 would be mapped to Xd-d but I.M.F would still map to X.X.X.

Prefix and suffix features are also useful.

Thank You

References

	Author(s), Title, Edition, Publishing House
T1	Speech and Language processing: An introduction to Natural Language Processing, Computational Linguistics and speech Recognition by Daniel Jurafsky and James H. Martin[3rd edition]
T2	Foundations of statistical Natural language processing by Christopher D.Manning and Hinrich schutze