Empirical Methods in Natural Language Processing
Lecture 9
Part-of-speech tagging and HMMs

(based on slides by Sharon Goldwater and Philipp Koehn)

22 February 2024

What is part of speech tagging?

- Given a string:

  This is a simple sentence

- Identify parts of speech (syntactic categories):

  This/DET is/VB a/DET simple/ADJ sentence/NOUN

Why do we care about POS tagging?

- POS tagging is a first step towards syntactic analysis.

- Illustrates the use of **hidden Markov models** (HMMs), which are also used for many other tagging (sequence labelling) tasks.

Examples of other tagging tasks

- **Named entity recognition:** e.g., label words as belonging to **persons**, **organizations**, **locations**, or none of the above:

  Barack/PER Obama/PER spoke/NON from/NON the/NON White/LOC House/LOC today/NON ./NON

- **Information field segmentation:** Given specific type of text (classified advert, bibiography entry), identify which words belong to which "fields" (price/size/location, author/title/year)

  3BR/SIZE flat/TYPE in/NON Bruntsfield/LOC ,/NON near/LOC main/LOC roads/LOC ./NON Bright/FEAT ,/NON well/FEAT maintained/FEAT ...

Sequence labelling: key features

In all of these tasks, deciding the correct label depends on

- the word to be labeled
  - NER: Smith is probably a person.
  - POS tagging: chair is probably a noun.

- the labels of surrounding words
  - NER: if following word is an organization (say Corp.), then this word is more likely to be organization too.
  - POS tagging: if preceding word is a modal verb (say will) then this word is more likely to be a verb.

HMM combines these sources of information probabilistically.

Parts of Speech: reminder

- **Open class words** (or content words)
  - nouns, verbs, adjectives, adverbs
  - mostly content-bearing: they refer to objects, actions, and features in the world
  - *open* class, since there is no limit to what these words are, new ones are added all the time (email, website).

- **Closed class words** (or function words)
  - pronouns, determiners, prepositions, connectives, ...
  - there is a limited number of these
  - mostly functional: to tie the concepts of a sentence together

How many parts of speech?

- Both linguistic and practical considerations
- Corpus annotators decide. Distinguish between
  - proper nouns (names) and common nouns?
  - singular and plural nouns?
  - past and present tense verbs?
  - auxiliary and main verbs?
  - etc
- Commonly used tagsets for English usually have 40-100 tags. For example, the Penn Treebank has 45 tags.

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

J&M Fig 5.6: Penn Treebank POS tags

POS tags in other languages

- Morphologically rich languages often have compound morphosyntactic tags

`Noun+A3sg+P2sg+Nom` (J&M, p.196)

- Hundreds or thousands of possible combinations

- Predicting these requires more complex methods than what we will discuss (e.g., may combine an FST with a probabilistic disambiguation system)

Why is POS tagging hard?

The usual reasons!

- Ambiguity:

glass of water/NOUN	vs.	water/VERB the plants	
lie/VERB down	vs.	tell a lie/NOUN	
wind/VERB down	vs.	a mighty wind/NOUN	(homographs)

  How about time flies like an arrow?

- Sparse data:
  - Words we haven't seen before (at all, or in this context)
  - Word-Tag pairs we haven't seen before (e.g., if we verb a noun)

Relevant knowledge for POS tagging

Remember, we want a model that decides tags based on

- The word itself
  - Some words may only be nouns, e.g. arrow
  - Some words are ambiguous, e.g. like, flies
  - Probabilities may help, if one tag is more likely than another

- Tags of surrounding words
  - two determiners rarely follow each other
  - two base form verbs rarely follow each other
  - determiner is almost always followed by adjective or noun

A probabilistic model for tagging

To incorporate these sources of information, we imagine that the sentences we observe were generated probabilistically as follows.

- To generate sentence of length n:

  Let $t_0 =$`<s>`
  For $i = 1$ to n
  Choose a tag conditioned on previous tag: $P(t_i \mid t_{i-1})$
  Choose a word conditioned on its tag: $P(w_i \mid t_i)$

- So, the model assumes:
  - Each tag depends only on previous tag: a bigram tag model.
  - Words are independent given tags

A probabilistic model for tagging

In math:

$$P(T, W) = \prod_{i=1}^{n} P(t_i \mid t_{i-1}) \times P(w_i \mid t_i)$$

A probabilistic model for tagging

In math:

$$P(T,W) = \prod_{i=1}^{n} P(t_i \mid t_{i-1}) \times P(w_i \mid t_i) \times P(\texttt{</s>} \mid t_n)$$

where $w_0 = \texttt{<s>}$ and $|W| = |T| = n$

A probabilistic model for tagging

In math:

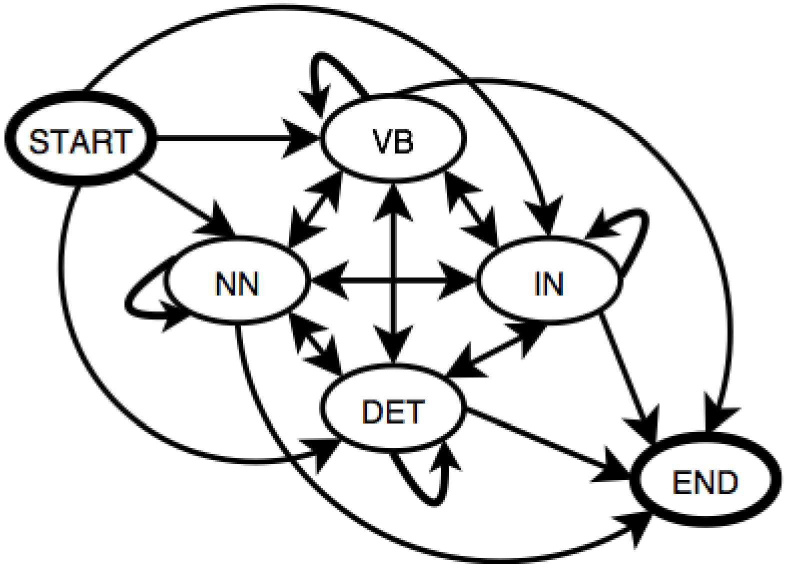
$$
\begin{aligned}
P(T, W) &= \prod_{i=1}^{n} P(t_i \mid t_{i-1}) \times P(w_i \mid t_i) \\
&\quad \times P(\texttt{</s>} \mid t_n)
\end{aligned}
$$

where $w_0 = \texttt{<s>}$ and $|W| = |T| = n$

- This can be thought of as a language model over words + tags. (Kind of a hybrid of a bigram language model and naïve Bayes.)
- But typically, we don't know the tags—i.e. they're **hidden**. It is therefore a **bigram hidden Markov model** (**HMM**).

Probabilistic finite-state machine

- One way to view the model: sentences are generated by walking through **states** in a graph. Each state represents a tag.



- Prob of moving from state s to s' (**transition probability**): $P(t_i = s' \mid t_{i-1} = s)$

Example transition probabilities

$t_{i-1} \backslash t_i$	NNP	MD	VB	JJ	NN	...
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	...
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	...
MD	0.0008	0.0002	0.7968	0.0005	0.0008	...
VB	0.0322	0.0005	0.0050	0.0837	0.0615	...
JJ	0.0306	0.0004	0.0001	0.0733	0.4509	...
...	...	...	...	...	...	...

- Probabilities estimated from tagged WSJ corpus, showing, e.g.:
  - Proper nouns (NNP) often begin sentences: $P(\texttt{NNP}|\texttt{<s>}) \approx 0.28$
  - Modal verbs (MD) nearly always followed by bare verbs (VB).
  - Adjectives (JJ) are often followed by nouns (NN).

Table excerpted from J&M draft 3rd edition, Fig 8.5

Example transition probabilities

$t_{i-1} \backslash t_i$	NNP	MD	VB	JJ	NN	...
<s>	0.2767	0.0006	0.0031	0.0453	0.0449	...
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	...
MD	0.0008	0.0002	0.7968	0.0005	0.0008	...
VB	0.0322	0.0005	0.0050	0.0837	0.0615	...
JJ	0.0306	0.0004	0.0001	0.0733	0.4509	...
...	...	...	...	...	...	...

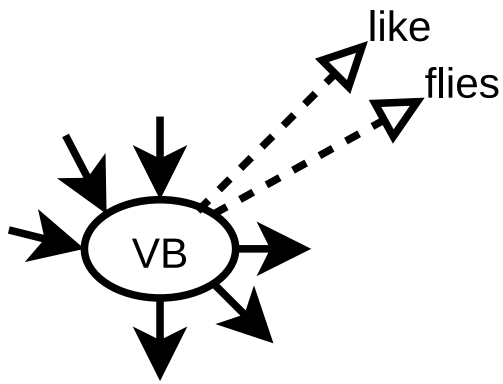
- This table is incomplete!

- In the full table, every row must sum up to 1 because it is a **distribution** over the next state (given previous).

Table excerpted from J&M draft 3rd edition, Fig 8.5

Probabilistic finite-state machine: outputs

- When passing through each state, emit a word.



- Prob of emitting w from state s (**emission** or **output probability**):
  $P(w_i = w \mid t_i = s)$

Example output probabilities

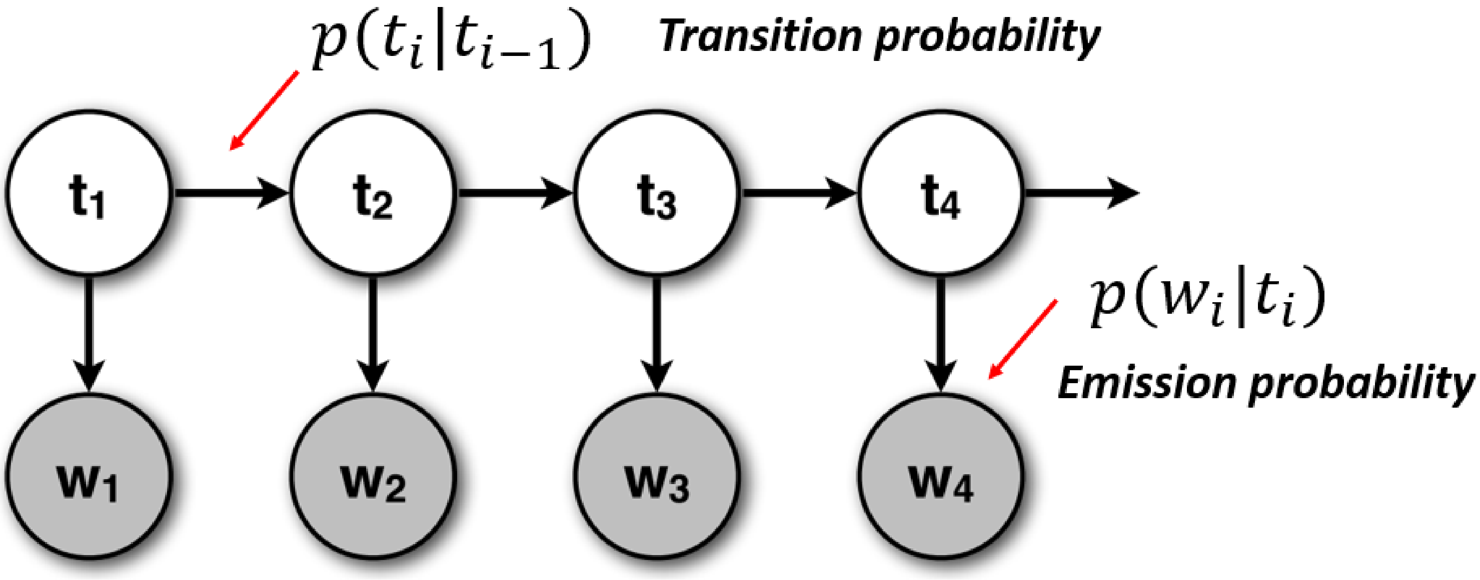
$t_i \backslash w_i$	Janet	will	back	the	...
NNP	0.000032	0	0	0.000048	...
MD	0	0.308431	0	0	...
VB	0	0.000028	0.000672	0	...
DT	0	0	0	0.506099	...
...	...	...	...	...	...

- MLE probabilities from tagged WSJ corpus, showing, e.g.:
  - 0.0032% of proper nouns are *Janet*: $P(\texttt{Janet}|\texttt{NNP}) = 0.000032$
  - About half of determiners (DT) are *the*.
  - *the* can also be a proper noun. (Annotation error?)
- Again, in full table, rows would sum to 1.

From J&M draft 3rd edition, Fig 8.6

Graphical Model Diagram

In **graphical model** notation, circles = random variables, and each arrow = a conditional probability factor in the joint likelihood:



$\rightarrow$ = a lookup in the transition distribution,
$\downarrow$ = a lookup in the emission distribution.

http://www.cs.virginia.edu/~hw5x/Course/CS6501-Text-Mining/_site/mps/mp3.html

What can we do with this model?

- If we know the transition and output probabilities, we can compute the probability of a tagged sentence.

- That is,
  - suppose we have sentence $W = w_1 \dots w_n$ and its tags $T = t_1 \dots t_n$.
  - what is the probability that our probabilistic FSM would generate exactly that sequence of words and tags, if we stepped through at random?

What can we do with this model?

- If we know the transition and output probabilities, we can compute the probability of a tagged sentence.
  - suppose we have sentence $W = w_1 \dots w_n$ and its tags $T = t_1 \dots t_n$.
  - what is the probability that our probabilistic FSM would generate exactly that sequence of words and tags, if we stepped through at random?

- This is the **joint probability**

$$P(W,T) = \prod_{i=1}^{n} P(t_i \mid t_{i-1}) P(w_i \mid t_i) \cdot P(\texttt{</s>} \mid t_n)$$

Example: computing joint prob. $P(W, T)$

What's the probability of this tagged sentence?

$$\text{This/DET is/VB a/DET simple/JJ sentence/NN}$$

Example: computing joint prob. $P(W,T)$

What's the probability of this tagged sentence?

This/DET is/VB a/DET simple/JJ sentence/NN

- First, add begin- and end-of-sentence <s> and </s>. Then:

$$
\begin{aligned}
P(W,T) &= \left[\prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i)\right] P(\texttt{</s>}|t_n) \\
&= P(\mathsf{DET}|\texttt{<s>})P(\mathsf{VB}|\mathsf{DET})P(\mathsf{DET}|\mathsf{VB})P(\mathsf{JJ}|\mathsf{DET})P(\mathsf{NN}|\mathsf{JJ})P(\texttt{</s>}|\mathsf{NN}) \\
&\quad \cdot P(\mathsf{This}|\mathsf{DET})P(\mathsf{is}|\mathsf{VB})P(\mathsf{a}|\mathsf{DET})P(\mathsf{simple}|\mathsf{JJ})P(\mathsf{sentence}|\mathsf{NN})
\end{aligned}
$$

- Then, plug in the probabilities we estimated from our corpus.

But... tagging?

Normally, we want to use the model to find the best tag sequence for an *untagged* sentence.

- Thus, the name of the model: **hidden Markov model**
  - **Markov**: because of Markov independence assumption (each tag/state only depends on fixed number of previous tags/states—here, just one).
  - **hidden**: because at test time we only see the words/emissions; the tags/states are hidden (or **latent**) variables.
- FSM view: given a sequence of words, what is the most probable state path that generated them?

Hidden Markov Model (HMM)

HMM is actually a very general model for sequences. Elements of an HMM:

- a set of states (here: the tags)
- an output alphabet (here: words)
- intitial state (here: beginning of sentence)
- state transition probabilities (here: $P(t_i \mid t_{i-1})$)
- symbol emission probabilities (here: $P(w_i \mid t_i)$)

Relationship to previous models

- **N-gram model**: a model for sequences that also makes a Markov assumption, but has no hidden variables.

- **Naïve Bayes**: a model with hidden variables (the classes) but no sequential dependencies.

- **HMM**: a model for sequences with hidden variables.

Like many other models with hidden variables, we will use Bayes' Rule to help us infer the values of those variables.

(In NLP, we usually assume hidden variables *are* observed during training—though there are *unsupervised* methods that do not.)

Relationship to other models

Side note for those interested:

- **Naïve Bayes**: a generative model (use Bayes' Rule, strong independence assumptions) for classification.

- **MaxEnt**: a discriminative model (model $P(y \mid x)$ directly, use arbitrary features) for classification.

- **HMM**: a generative model for sequences with hidden variables.

- **MEMM**: a discriminative model for sequences with hidden variables. Other sequence models can also use more features than HMM: e.g., **Conditional Random Field** (CRF) or **structured perceptron**.

Formalizing the tagging problem

Find the best tag sequence T for an *untagged* sentence W:

$$\arg\max_{T} \; P(T \mid W)$$

Formalizing the tagging problem

Find the best tag sequence T for an *untagged* sentence W:

$$\arg\max_T \; P(T \mid W)$$

- Bayes' rule gives us:

$$P(T \mid W) = \frac{P(W \mid T)\; P(T)}{P(W)}$$

- We can drop $P(W)$ if we are only interested in $\arg\max_T$:

$$\arg\max_T \; P(T \mid W) = \arg\max_T \; P(W \mid T)\; P(T)$$

Decomposing the model

Now we need to compute $P(W \mid T)$ and $P(T)$ (actually, their product $P(W \mid T)P(T) = P(W, T)$).

- We already defined how!
- $P(T)$ is the state transition sequence:

$$P(T) = \prod_i P(t_i \mid t_{i-1})$$

- $P(W \mid T)$ are the emission probabilities:

$$P(W \mid T) = \prod_i P(w_i \mid t_i)$$

Search for the best tag sequence

- We have defined a model, but how do we use it?
  - given: word sequence W
  - wanted: best tag sequence T^*

- For any specific tag sequence T, it is easy to compute $P(W,T) = P(W \mid T)P(T)$.

$$P(W \mid T)\ P(T) = \prod_i P(w_i \mid t_i)\ P(t_i \mid t_{i-1})$$

- So, can't we just enumerate all possible T, compute their probabilites, and choose the best one?

Enumeration won't work

- Suppose we have c possible tags for each of the n words in the sentence.

- How many possible tag sequences?

Enumeration won't work

- Suppose we have c possible tags for each of the n words in the sentence.
- How many possible tag sequences?
- There are c^n possible tag sequences: the number grows *exponentially* in the length n.
- For all but small n, too many sequences to efficiently enumerate.

The Viterbi algorithm

- We'll use a **dynamic programming** algorithm to solve the problem.
- Dynamic programming algorithms order the computation efficiently so partial values can be computed once and reused.
- The **Viterbi algorithm** finds the best tag sequence without explicitly enumerating all sequences.
- Partial results are stored in a **chart** to avoid recomputing them.
- Details next time.

Viterbi as a decoder

The problem of finding the best tag sequence for a sentence is sometimes called **decoding**.

- Because, like spelling correction etc., HMM can also be viewed as a noisy channel model.
  - Someone wants to send us a sequence of tags: $P(T)$
  - During encoding, "noise" converts each tag to a word: $P(W|T)$
  - We try to decode the observed words back to the original tags.

- In fact, *decoding* is a general term in NLP for inferring the hidden variables in a test instance (so, finding correct spelling of a misspelled word is also decoding).

Computing marginal prob. $P(W)$

Recall that the HMM can be thought of as a language model over words AND tags. What about estimating probabilities of JUST the words of a sentence?

Computing marginal prob. $P(W)$

Recall that the HMM can be thought of as a language model over words AND tags. What about estimating probabilities of JUST the words of a sentence?

$$P(W) = \sum_{T} P(W, T)$$

Computing marginal prob. $P(W)$

Recall that the HMM can be thought of as a language model over words AND tags. What about estimating probabilities of JUST the words of a sentence?

$$P(W) = \sum_{T} P(W, T)$$

Again, cannot enumerate all possible taggings T. Instead, use the **forward algorithm** (dynamic programming algorithm closely related to Viterbi—see textbook if interested).

Could be used to measure perplexity of held-out data.

Supervised learning

The HMM consists of

- transition probabilities
- emission probabilities

How can these be learned?

Supervised learning

The HMM consists of

- transition probabilities
- emission probabilities

How can these be estimated? From counts in a treebank such as the Penn Treebank!

Supervised learning

The HMM consists of

- transition probabilities: given tag t, what is the probability that t' follows?

- emission probabilities: given tag t, what is the probability that the word is w?

How can these be estimated? From counts in a treebank such as the Penn Treebank!

Do transition & emission probs. need smoothing?

Do transition & emission probs. need smoothing?

- **Emissions:** yes, because if there is any word w in the test data such that $P(w_i = w \mid t_i = t) = 0$ for all tags t, the whole joint probability will go to 0.

- **Transitions:** not necessarily, but if any transition probabilities are estimated as 0, that tag bigram will never be predicted.

  – What are some transitions that should NEVER occur in a bigram HMM?

Do transition & emission probs. need smoothing?

- **Emissions:** yes, because if there is any word w in the test data such that $P(w_i = w \mid t_i = t) = 0$ for all tags t, the whole joint probability will go to 0.

- **Transitions:** not necessarily, but if any transition probabilities are estimated as 0, that tag bigram will never be predicted.
  - What are some transitions that should NEVER occur in a bigram HMM?

    • $\rightarrow$ <s>

    </s> $\rightarrow$ •

    <s> $\rightarrow$ </s>

Unsupervised learning

- With the number of hidden tags specified but no tagged training data, the learning is **unsupervised**.

- The Forward-Backward algorithm, a.k.a. Baum-Welch EM, clusters the data into "tags" that will give the training data high probability under the HMM. This is used in speech recognition.

- See the textbook for details if interested.

Higher-order HMMs

- The "Markov" part means we ignore history of more than a fixed number of words.
- Equations thus far have been for bigram HMM: i.e., transitions are $P(t_i \mid t_{i-1})$.
- But as with language models, we can increase the N in the N-gram: **trigram** HMM transition probabilities are $P(t_i \mid t_{i-2}, t_{i-1})$, etc.
- As usual, smoothing the transition distributions becomes more important with higher-order models.

Summary

- Part-of-speech tagging is a sequence labelling task.
- HMM uses two sources of information to help resolve ambiguity in a word's POS tag:
  - The words itself
  - The tags assigned to surrounding words
- Can be viewed as a probabilistic FSM.
- Given a tagged sentence, easy to compute its probability. But finding the best tag sequence will need a clever algorithm.