ANLP Lecture 8
Part-of-speech tagging

Sharon Goldwater
(based on slides by Philipp Koehn)

1 October 2019

School of informatics

Orientation

Lectures 5-6	
Task:	Language modelling
Model:	Sequence model, all variables directly observed

Lecture 7	
Task:	Text classification
Model:	Bag-of-words model, Includes hidden variables (categories of documents)

Orientation

Lectures 5-6	
Task:	Language modelling
Model:	Sequence model, all variables directly observed

Lecture 7	
Task:	Text classification
Model:	Bag-of-words model, Includes hidden variables (categories of documents)

Lectures 8-9	
Task:	Part-of-speech tagging
Model:	Sequence model, Includes hidden variables (categories of words in sequence)

Today's lecture

- What are parts of speech and POS tagging?
- What linguistic information should we consider?
- What are some different tagsets and cross-linguistic issues?
- What is a Hidden Markov Model?
- (Next time: what algorithms do we need for HMMs?)

What is part of speech tagging?

- Given a string:

  This is a simple sentence

- Identify parts of speech (syntactic categories):

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

- First step towards syntactic analysis
- Illustrates use of hidden Markov models to label sequences

Other tagging tasks

Other problems can also be framed as tagging (sequence labelling):

- **Case restoration:** If we just get lowercased text, we may want to restore proper casing, e.g. the river Thames
- **Named entity recognition:** it may also be useful to find names of persons, organizations, etc. in the text, e.g. Barack Obama
- **Information field segmentation:** Given specific type of text (classified advert, bibiography entry), identify which words belong to which "fields" (price/size/#bedrooms, author/title/year)
- **Prosodic marking:** In speech synthesis, which words/syllables have stress/intonation changes, e.g. He's going. vs He's going?

Parts of Speech

- **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

English POS tag sets

Usually have 40-100 tags. For example,

- Brown corpus (87 tags)
  - One of the earliest large corpora collected for computational linguistics (1960s)
  - A **balanced** corpus: different genres (fiction, news, academic, editorial, etc)
- Penn Treebank corpus (45 tags)
  - First large corpus annotated with POS and full syntactic trees (1992)
  - Possibly the most-used corpus in NLP
  - Originally, just text from the Wall Street Journal (WSJ)

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)

Universal POS tags (Petrov et al., 2011)

- A move in the other direction
- Simplify the set of tags to lowest common denominator across languages
- Map existing annotations onto universal tags

  $\{\text{VB, VBD, VBG, VBN, VBP, VBZ, MD}\} \Rightarrow \text{VERB}$

- Allows interoperability of systems across languages
- Promoted by Google and others

Universal POS tags (Petrov et al., 2011)

NOUN (nouns)
VERB (verbs)
ADJ (adjectives)
ADV (adverbs)
PRON (pronouns)
DET (determiners and articles)
ADP (prepositions and postpositions)
NUM (numerals)
CONJ (conjunctions)
PRT (particles)
'.' (punctuation marks)
X (anything else, such as abbreviations or foreign words)

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

Relevant knowledge for POS tagging

- 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
- Local context
  - 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

Let's define a new generative process for sentences.

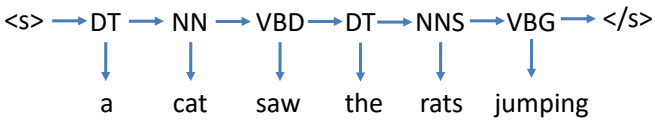
- To generate sentence of length n:

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

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

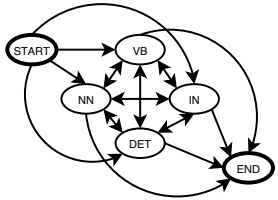
Generative process example

- Arrows indicate probabilistic dependencies:



Probabilistic finite-state machine

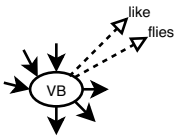
- One way to view the model: sentences are generated by walking through **states** in a graph. Each state represents a tag.



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

Probabilistic finite-state machine

- When passing through a state, emit a word.



- Prob of emitting w from state s (**emission probability**): $P(w_i = w|t_i = s)$

What can we do with this model?

- Simplest thing: if we know the parameters (tag transition and word emission probabilities), can compute the probability of a tagged sentence.

- Let $S = w_1 \dots w_n$ be the sentence and $T = t_1 \dots t_n$ be the corresponding tag sequence. Then

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

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

What's the probability of this tagged sentence?

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

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

What's the probability of this tagged sentence?

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

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

$$\begin{aligned} p(S,T) &= \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i) \\ &= P(\text{DT}|\text{<s>})P(\text{VB}|\text{DT})P(\text{DT}|\text{VB})P(\text{JJ}|\text{DT})P(\text{NN}|\text{JJ})P(\text{</s>}|\text{NN}) \\ &\quad \cdot P(\text{This}|\text{DT})P(\text{is}|\text{VB})P(\text{a}|\text{DT})P(\text{simple}|\text{JJ})P(\text{sentence}|\text{NN}) \end{aligned}$$

- But now we need to plug in probabilities... from where?

Training the model

Given a corpus annotated with tags (e.g., Penn Treebank), we *estimate* $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ using familiar methods (MLE/smoothing)

Training the model

Given a corpus annotated with tags (e.g., Penn Treebank), we *estimate* $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ using familiar methods (MLE/smoothing)

	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	0.0306	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

Figure 8.5 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.

(Fig from J&M draft 3rd edition)

Training the model

Given a corpus annotated with tags (e.g., Penn Treebank), we *estimate* $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ using familiar methods (MLE/smoothing)

	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	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.6 Observation likelihoods *B* computed from the WSJ corpus without smoothing.

(Fig from J&M draft 3rd edition)

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 assumption (tag/state only depends on immediately previous tag/state).
  - **hidden**: because we only observe 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: $p(t_i|t_{i-1})$)
- symbol emission probabilities (here: $p(w_i|t_i)$)

Formalizing the tagging problem

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

$$\text{argmax}_T \, p(T|S)$$

Formalizing the tagging problem

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

$$\text{argmax}_T \; p(T|S)$$

- Bayes' rule gives us:

$$p(T|S) = \frac{p(S|T)\; p(T)}{p(S)}$$

- We can drop $p(S)$ if we are only interested in argmax_T:

$$\text{argmax}_T \; p(T|S) = \text{argmax}_T \; p(S|T)\; p(T)$$

Decomposing the model

Now we need to compute $P(S|T)$ and $P(T)$ (actually, their product $P(S|T)P(T) = P(S,T)$).

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

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

- $P(S|T)$ are the emission probabilities:

$$P(S|T) = \prod_i P(w_i|t_i)$$

Search for the best tag sequence

- We have defined a model, but how do we use it?
  - given: word sequence S
  - wanted: best tag sequence T^*
- For any specific tag sequence T, it is easy to compute $P(S,T) = P(S|T)P(T)$.

$$P(S|T)\; P(T) = \prod_i P(w_i|t_i)\; P(t_i|t_{i-1})$$

- 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.

Finding the best path

- The **Viterbi algorithm** finds this path without explicitly enumerating all paths.
- Our second example of a **dynamic programming** (or **memoization**) algorithm.
- Like min. edit distance, the algorithm stores partial results in a **chart** to avoid recomputing them.

Tagging example

Words:	<s>	one	dog	bit	</s>
Possible tags: (ordered by frequency for each word)	<s>	CD	NN	NN	</s>
		NN	VB	VBD	
		PRP			

Tagging example

Words:	<s>	one	dog	bit	</s>
Possible tags: (ordered by frequency for each word)	<s>	CD	NN	NN	</s>
		NN	VB	VBD	
		PRP			

- Choosing the best tag for each word independently gives the wrong answer (<s> CD NN NN </s>).
- P(VBD|bit) < P(NN|bit), but may yield a better *sequence* (<s> CD NN VB </s>)
  - because P(VBD|NN) and P(</s>|VBD) are high.

Viterbi: intuition

Words:	<s>	one	dog	bit	</s>
Possible tags: (ordered by frequency for each word)	<s>	CD	NN	NN	</s>
		NN	VB	VBD	
		PRP			

- Suppose we have already computed
  a) The best tag sequence for <s> … bit that ends in NN.
  b) The best tag sequence for <s> … bit that ends in VBD.
- Then, the best full sequence would be either
  – sequence (a) extended to include </s>, or
  – sequence (b) extended to include </s>.

Viterbi: intuition

Words:	<s>	one	dog	bit	</s>
Possible tags: (ordered by frequency for each word)	<s>	CD	NN	NN	</s>
		NN	VB	VBD	
		PRP			

- But similarly, to get
  a) The best tag sequence for <s> … bit that ends in NN.
- We could extend one of:
  – The best tag sequence for <s> … dog that ends in NN.
  – The best tag sequence for <s> … dog that ends in VB.
- And so on…

Viterbi: high-level picture

- Intuition: the best path of length t ending in state q must include the best path of length t-1 to the previous state. (t now a *time step*, not a *tag*).

Viterbi: high-level picture

- Intuition: the best path of length t ending in state q must include the best path of length t-1 to the previous state. (t now a *time step*, not a *tag*). So,
  – Find the best path of length t-1 to each state.
  – Consider extending each of those by 1 step, to state q.
  – Take the best of those options as the best path to state q.

Summary

- Parts of speech (syntactic categories) provide the beginning of syntactic analysis, categorizing words by their behaviour.
- Hidden Markov models are a probabilistic model for POS tagging (and other sequence labelling tasks)
- HMM defines the joint probability of (tags, words).
- To find the best tag sequence, use the Viterbi Algorithm (details next time).

Questions and exercises

1. Do JM3 Exercise 8.1.
2. POS taggers are normally evaluated using *accuracy*: (the percentage of the tags assigned by the tagger that agree with the gold standard). If we are using an HMM for Named Entity Recognition, does this measure still make sense, or is there some other evaluation measure that could make more sense? Why?
3. I motivated the HMM by saying that local context should affect the model's POS prediction. Which term in the model is responsible for incorporating context information? Is the POS prediction affected by words on *both* sides of the current word, or only on one side? Explain your answer.

References

Petrov, S., Das, D., and McDonald, R. (2011). A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.