Part-of-speech tagging and Hidden Markov Models

Jeremy Barnes
HAP/LAP Master

14.01.2022

Universidad del País Vasco — Euskal Herriko Unibertsitatea

HiTZ Hizkuntza Teknologiako Zentroa — Basque Center for Language Technology

Part of speech

Parts of speech

Definition

- A basic syntactic concept that refers to the syntactic role of each word in a sentence.

Parts of speech

Definition

- A basic syntactic concept that refers to the syntactic role of each word in a sentence.
- Parts-of-speech can help to disentangle or explain various linguistic problems.

Parts of speech

Definition

- A basic syntactic concept that refers to the syntactic role of each word in a sentence.
- Parts-of-speech can help to disentangle or explain various linguistic problems.
- Intuitively, what is the difference between the first sentence which is grammatically acceptable and the second that is not?

Parts of speech

Definition

- A basic syntactic concept that refers to the syntactic role of each word in a sentence.
- Parts-of-speech can help to disentangle or explain various linguistic problems.
- Intuitively, what is the difference between the first sentence which is grammatically acceptable and the second that is not?

(a) Colorless green ideas sleep furiously.

(b) *Ideas colorless furiously green sleep.

Parts of speech

(a) Colorless green ideas sleep furiously.

(b) *Ideas colorless furiously green sleep.

Parts of speech

(a) Colorless green ideas sleep furiously.
(b) *Ideas colorless furiously green sleep.

The first sentence contains normal transitions for English (ADJ NOUN) (NOUN VERB), while the second contains unnatural transitions (NOUN ADJ), (ADJ VERB).

Parts of speech

(a) Colorless green ideas sleep furiously.
(b) *Ideas colorless furiously green sleep.

The first sentence contains normal transitions for English (ADJ NOUN) (NOUN VERB), while the second contains unnatural transitions (NOUN ADJ), (ADJ VERB).

Definition

- Vocabulary: V = {colorless, green, ideas, person, dog, ...}
- Tag set: T = {DET, NOUN, VERB, ADJ, ...}
- Task: given a sequence of tokens, assign the corresponding labels (POS tags):

Parts of speech

(a) Colorless green ideas sleep furiously.
(b) *Ideas colorless furiously green sleep.

The first sentence contains normal transitions for English (ADJ NOUN) (NOUN VERB), while the second contains unnatural transitions (NOUN ADJ), (ADJ VERB).

Definition

- Vocabulary: V = {colorless, green, ideas, person, dog, ...}
- Tag set: T = {DET, NOUN, VERB, ADJ, ...}
- Task: given a sequence of tokens, assign the corresponding labels (POS tags):

Colorless/ADJ green/ADJ ideas/NOUN sleep/VERB furiously/ADV ./PUNCT

Parts of speech

What purpose do they have?

- Part-of-speech tags are morpho-syntactic in nature.
- They do not describe semantic categories.
- pos_tag("... the howling of the shrieking storm") =

Parts of speech

What purpose do they have?

- Part-of-speech tags are morpho-syntactic in nature.
- They do not describe semantic categories.
- pos_tag("... the howling of the shrieking storm") =

... the/DET howling/NOUN of/ADP the/DET shrieking/ADJ storm/NOUN

Parts of speech

What purpose do they have?

- Part-of-speech tags are morpho-syntactic in nature.
- They do not describe semantic categories.
- pos_tag("... the howling of the shrieking storm") =

... the/DET howling/NOUN of/ADP the/DET shrieking/ADJ storm/NOUN

Part-of-speech approaches

Penn Treebank tagset

- Penn treebank was one of the first large-scaled tagged corpora.
- Penn treebank tagset was most popular for English.
- 36 tags, many specific for English
- `https://aclanthology.org/J93-2004/`

Part-of-speech approaches

Penn Treebank tagset

- Penn treebank was one of the first large-scaled tagged corpora.
- Penn treebank tagset was most popular for English.
- 36 tags, many specific for English
- https://aclanthology.org/J93-2004/

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 participle	*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*

Figure 8.2 Penn Treebank part-of-speech tags.

Part-of-speech approaches

Universal Dependencies

- Attempt at a language-invariant tagset.
- Data available in 100+ languages.
- 17 tags.
- `https://universaldependencies.org/`

Part-of-speech approaches

Universal Dependencies

- Attempt at a language-invariant tagset.
- Data available in 100+ languages.
- 17 tags.
- https://universaldependencies.org/

	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

Tagging problem: First approximation

Let's start with a naïve approach

Tagging problem: First approximation

Let's start with a naïve approach

- Assign the most likely POS tag to each token independently.
- pos_tag("Colorless green ideas sleep furiously") =

Tagging problem: First approximation

Let's start with a naïve approach

- Assign the most likely POS tag to each token independently.
- pos_tag("Colorless green ideas sleep furiously") =
- pos_tag("Colorless") $\rightarrow$ ADJ
- pos_tag("green") $\rightarrow$ ADJ
- pos_tag("ideas") $\rightarrow$ NOUN
- pos_tag("sleep") $\rightarrow$ VERB
- pos_tag("furiously") $\rightarrow$ ADV

Tagging problem: First approximation

Let's start with a naïve approach

- Assign the most likely POS tag to each token independently.
- pos_tag("Colorless green ideas sleep furiously") =
- pos_tag("Colorless") $\rightarrow$ ADJ
- pos_tag("green") $\rightarrow$ ADJ
- pos_tag("ideas") $\rightarrow$ NOUN
- pos_tag("sleep") $\rightarrow$ VERB
- pos_tag("furiously") $\rightarrow$ ADV

What problems are there with this model?

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.
- pos_tag("Colorless green ideas sleep furiously") =

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.
- pos_tag("Colorless green ideas sleep furiously") =
- pos_tag(*, "Colorless") → ADJ

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.
- pos_tag("Colorless green ideas sleep furiously") $=$
- pos_tag(*, "Colorless") $\rightarrow$ ADJ
- pos_tag(ADJ, "green") $\rightarrow$ ADJ

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.
- pos_tag("Colorless green ideas sleep furiously") $=$
- pos_tag(*, "Colorless") $\rightarrow$ ADJ
- pos_tag(ADJ, "green") $\rightarrow$ ADJ
- pos_tag(ADJ, "ideas") $\rightarrow$ NOUN

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.
- pos_tag("Colorless green ideas sleep furiously") =
- pos_tag(*, "Colorless") $\rightarrow$ ADJ
- pos_tag(ADJ, "green") $\rightarrow$ ADJ
- pos_tag(ADJ, "ideas") $\rightarrow$ NOUN
- pos_tag(NOUN, "sleep") $\rightarrow$ VERB

Sequential tagging

Assign tags sequentially from left to right:

- Previously predicted tags can be taken into account.
- Includes a bounded dependency.
- pos_tag("Colorless green ideas sleep furiously") $=$
- pos_tag(*, "Colorless") $\rightarrow$ ADJ
- pos_tag(ADJ, "green") $\rightarrow$ ADJ
- pos_tag(ADJ, "ideas") $\rightarrow$ NOUN
- pos_tag(NOUN, "sleep") $\rightarrow$ VERB
- pos_tag(VERB, "furiously") $\rightarrow$ ADV

Sequential tagging

Two types of constraints

- Local: the word sleep is more commonly used as a VERB, rather than NOUN.
- Contextual: the tag VERB often follows ADJ NOUN

Sequential tagging

- Imagine we have a set of sentences X and tag sequences Y.

Sequential tagging

- Imagine we have a set of sentences X and tag sequences Y.
- Each sentence is a sequence of words: $X_i = w_1, w_2, \ldots, w_n$
- and tag sequences: $Y_i = t_1, t_2, \ldots, t_n$

Sequential tagging

- Imagine we have a set of sentences X and tag sequences Y.
- Each sentence is a sequence of words: $X_i = w_1, w_2, \ldots, w_n$
- and tag sequences: $Y_i = t_1, t_2, \ldots, t_n$
- We need a function to map sentences to tag sequences $f(x) = y$

- Imagine we have a set of sentences X and tag sequences Y.
- Each sentence is a sequence of words: $X_i = w_1, w_2, \ldots, w_n$
- and tag sequences: $Y_i = t_1, t_2, \ldots, t_n$
- We need a function to map sentences to tag sequences $f(x) = y$

How could we incorporate the two constraints (local and contextual) into a tagging model?

(Talk with partners for 5 minutes)

Sequential tagging

Generative model

- We want to find $p(x, y)$

Sequential tagging

Generative model

- We want to find $p(x, y)$
- Again, we use the refactoring available from probability theory:
- $p(x, y) = p(y)p(x|y)$

Sequential tagging

Generative model

- We want to find $p(x, y)$
- Again, we use the refactoring available from probability theory:
- $p(x, y) = p(y)p(x|y)$
- Models that apply this method are called generative models

Sequential tagging

Generative model

- We want to find $p(x, y)$
- Again, we use the refactoring available from probability theory:
- $p(x, y) = p(y)p(x|y)$
- Models that apply this method are called generative models
  - First, we generate y with probability $p(y)$: called hidden state

Sequential tagging

Generative model

- We want to find $p(x, y)$
- Again, we use the refactoring available from probability theory:
- $p(x, y) = p(y)p(x|y)$
- Models that apply this method are called generative models
  - First, we generate y with probability $p(y)$: called hidden state
  - Next, given y, we generate x with probability $p(x|y)$: we generate the observed data from our current hidden state

Sequential tagging

Generative model

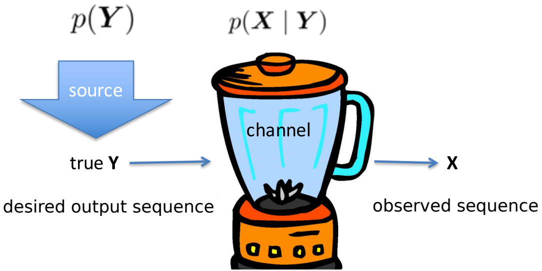
- We want to find $p(x, y)$
- Again, we use the refactoring available from probability theory:
- $p(x, y) = p(y)p(x|y)$
- Models that apply this method are called generative models
  - First, we generate y with probability $p(y)$: called hidden state
  - Next, given y, we generate x with probability $p(x|y)$: we generate the observed data from our current hidden state
  - Note that this does not model $p(y|x)$, i.e. 'given our input, what should be the output'...

Sequential tagging

Generative model

- We want to find $p(x, y)$
- Again, we use the refactoring available from probability theory:
- $p(x, y) = p(y)p(x|y)$
- Models that apply this method are called generative models
  - First, we generate y with probability $p(y)$: called hidden state
  - Next, given y, we generate x with probability $p(x|y)$: we generate the observed data from our current hidden state
  - Note that this does not model $p(y|x)$, i.e. 'given our input, what should be the output'...
  - instead it answers the counterintuitive question 'If we were going to generate y, how likely is it that the model will be x?'

Noisy Channel



decoding rule:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y}) \times p(\boldsymbol{y})$$

Sequential tagging

Generative model

- After training, these models can also be used to “generate” random instances (outcomes), either of an observation and target (x, y), or of an observation x given a target value y $(x|y)$.

Generative model

- After training, these models can also be used to "generate" random instances (outcomes), either of an observation and target (x, y), or of an observation x given a target value y $(x|y)$.
- Given a model of one conditional probability, and estimated probability distributions for the variables x and y, denoted P(x) and P(y), one can estimate the opposite conditional probability using Bayes' rule:
- $P(x|y)P(y) = P(y|x)P(x)$

Generative model

- After training, these models can also be used to "generate" random instances (outcomes), either of an observation and target (x, y), or of an observation x given a target value y $(x|y)$.
- Given a model of one conditional probability, and estimated probability distributions for the variables x and y, denoted P(x) and P(y), one can estimate the opposite conditional probability using Bayes' rule:
- $P(x|y)P(y) = P(y|x)P(x)$
- If we make the same simplifying assumption about the conditional probability as we did with n-grams (Markov assumption), these models are called Hidden Markov Models.

Hidden Markov Models

Model

- $p(x, y) = p(y)p(x|y)$
- $x = w_1, w_2, \ldots, w_n$
- $y = t_1, t_2, \ldots, t_n$

Model

- $p(x, y) = p(y)p(x|y)$
- $x = w_1, w_2, \ldots, w_n$
- $y = t_1, t_2, \ldots, t_n$
- We want $max_{y \in Y} p(x, y) = max_{y \in Y} p(y)p(x|y)$

Hidden Markov Models

Model

- $p(x, y) = p(y)p(x|y)$
- $x = w_1, w_2, \ldots, w_n$
- $y = t_1, t_2, \ldots, t_n$
- We want $max_{y \in Y} p(x, y) = max_{y \in Y} p(y)p(x|y)$
- Unlike with N-grams, we want the most likely sequence of tags, not the probability itself.
- $argmax_{y \in Y} p(x, y) = argmax_{y \in Y} p(y)p(x|y)$

Hidden Markov Models

Model

- $p(x, y) = p(y)p(x|y)$
- $x = w_1, w_2, \ldots, w_n$
- $y = t_1, t_2, \ldots, t_n$
- We want $max_{y \in Y} p(x, y) = max_{y \in Y} p(y)p(x|y)$
- Unlike with N-grams, we want the most likely sequence of tags, not the probability itself.
- $argmax_{y \in Y} p(x, y) = argmax_{y \in Y} p(y)p(x|y)$
- Example:
- f(the best food) $= argmax_{y \in Y} p(x, y) = argmax_{y \in Y} p(y)p(x|y) =$ DET ADJ NOUN

Hidden Markov Models

Formalization

- a set of N states $Q = q_1, q_2, \ldots, q_N$
- a sequence of T observations $O = o_1, o_2, \ldots, o_T$
- a transition probability matrix $A = a_{11} \ldots a_{ij} \ldots a_{NN}$
- a sequence of observation likelihoods $B = b_i(o_t)$
- and an initial probability distribution over states $\pi = \pi_1, \pi_2, \ldots, \pi_N$

Hidden Markov Models

Formalization

- Two simplifying assumptions

Hidden Markov Models

Formalization

- Two simplifying assumptions
- Markov assumption: $P(q_i|q_1, q_2, \ldots, q_{i-1}) = P(q_i|q_{i-1})$
  - That is, the probability of a particular state only depends on the previous state.

Hidden Markov Models

Formalization

- Two simplifying assumptions
- Markov assumption: $P(q_i|q_1, q_2, \ldots, q_{i-1}) = P(q_i|q_{i-1})$
  - That is, the probability of a particular state only depends on the previous state.
- Output independence: $P(o_i|q_1, q_2, \ldots, q_T, o_1, o_2, \ldots, o_T) = P(o_i|q_i)$
  - That is, the probability of an output observation depends only on the current state.

Hidden Markov Models

Let's look at a concrete example:

- Given a sequence of observed words "It will be"
- and a current tag only for 'it will' DET AUX

Hidden Markov Models

Let's look at a concrete example:

- Given a sequence of observed words "It will be"
- and a current tag only for 'it will' DET AUX

1. What is the probability that the current tag is VERB?
   - $P(t_i|t_{i-1}) = \frac{count(AUX,VERB)}{count(AUX)} = \frac{10471}{13124} = .80$
2. What is the probability of 'be', given VERB?
   - $P(w_i|t_i) = \frac{count(VERB,'be')}{count(VERB)} = \frac{4046}{13126} = .31$

Let's look at a concrete example:

- Given a sequence of observed words "It will be"
- and a current tag only for 'it will' DET AUX

1. What is the probability that the current tag is VERB?
   - $P(t_i|t_{i-1}) = \frac{count(AUX,VERB)}{count(AUX)} = \frac{10471}{13124} = .80$
2. What is the probability of 'be', given VERB?
   - $P(w_i|t_i) = \frac{count(VERB,'be')}{count(VERB)} = \frac{4046}{13126} = .31$

- $P(t_i|t_{i-1}) \times P(w_i|t_i) = .80 * .31 = .248$

Hidden Markov Models

Summary

- $f(x_1, x_2, \ldots, x_n) =$
  $argmax_{y \in Y} p(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_{n+1}) =$
  $argmax_{y \in Y} \prod_{i=1}^{n+1} p(y_i | y_{i-1}) \prod_{i=1}^{n} p(x_i | y_i)$

Full example

- p("it will be", DET AUX VERB) $=$

  $q(DET|*) \times q(AUX|DET) \times q(VERB|AUX) \times q(STOP|VERB) \times b(it|DET) \times b(will|AUX) \times b(be|VERB)$

Hidden Markov Models

Full example

- p("it will be", DET AUX VERB) =

  $q(DET|*) \times q(AUX|DET) \times q(VERB|AUX) \times$
  $q(STOP|VERB) \times b(it|DET) \times b(will|AUX) \times b(be|VERB)$
- $y_o = *$, $y_{n+1} = STOP$

Digression: generative vs. discriminative models

Different views

- generative models: $p(x, y)$
- discriminative: $p(y|x)$

Digression: generative vs. discriminative models

Different views

- generative models: $p(x, y)$
  - Language models
  - Hidden Markov Models
  - Naive Bayes
- discriminative: $p(y|x)$

Digression: generative vs. discriminative models

Different views

- generative models: $p(x, y)$
  - Language models
  - Hidden Markov Models
  - Naive Bayes
- discriminative: $p(y|x)$
  - MaxEntropy model
  - Conditional Random Fields
  - Support Vector Machines

Parameter estimation

Same as with language models

- $q(VERB|AUX) = \frac{count(AUX, VERB)}{count(AUX)}$
- A typical tagset has between 20-60 tags.
- If a tagset has 20 tags, how many possible bigrams?

Parameter estimation

Same as with language models

- $q(VERB|AUX) = \frac{count(AUX, VERB)}{count(AUX)}$
- A typical tagset has between 20-60 tags.
- If a tagset has 20 tags, how many possible bigrams?
- Again, many paramaters would be zero.

Parameter estimation

Linear interpolation

- $q(\mathit{VERB}|\mathit{AUX}) = \lambda_1 \frac{count(\mathit{AUX}, \mathit{VERB})}{count(\mathit{AUX})} + \lambda_2 \frac{count(\mathit{VERB})}{count()}$

Unseen words

- What happens if we find a word that we haven't seen in training?

Parameter estimation

Unseen words

- What happens if we find a word that we haven't seen in training?
- Same as before, 0 probability.

Parameter estimation

Unseen words

- What happens if we find a word that we haven't seen in training?
- Same as before, 0 probability.
- Solution: replace all infrequent words with a single *UNK* token.

Parameter estimation

Example

- Hawke indicated Djokovic had his visa restored only on " procedural fairness grounds "

Parameter estimation

Example

- Hawke indicated Djokovic had his visa restored only on " procedural fairness grounds "
- *UNK* indicated *UNK* had his visa restored only on " *UNK* fairness grounds "

Parameter estimation

Example

- Hawke indicated Djokovic had his visa restored only on “ procedural fairness grounds ”
- *UNK* indicated *UNK* had his visa restored only on ” *UNK* fairness grounds ”
- Unfortunately, this conflates a lot of information.

Parameter estimation

Example

- Hawke indicated Djokovic had his visa restored only on “ procedural fairness grounds ”
- *UNK* indicated *UNK* had his visa restored only on ” *UNK* fairness grounds ”
- Unfortunately, this conflates a lot of information.
- Can you think of a simple way of augmenting this approach to differentiate KINDS of unseen words?

Parameter estimation

Example

- Hawke indicated Djokovic had his visa restored only on " procedural fairness grounds "
- *UNK* indicated *UNK* had his visa restored only on " *UNK* fairness grounds "
- Unfortunately, this conflates a lot of information.
- Can you think of a simple way of augmenting this approach to differentiate KINDS of unseen words?
- *UNK*-NAME indicated *UNK*-NAME had his visa restored only on " *UNK*-al fairness grounds "

Assigning tags to a new sentence

- So far, we've examined:

Assigning tags to a new sentence

- So far, we've examined:
  - The problem of POS tagging
  - HMMs and how they are parameterized

Assigning tags to a new sentence

- So far, we've examined:
    - The problem of POS tagging
    - HMMs and how they are parameterized
- but we are still missing how to apply the model to a new sentence

Example

I	suspect	the	present	forecast	is	pessimistic	.
CD	JJ	DT	JJ	NN	NNS	JJ	.
NN	NN	JJ	NN	VB	VBZ		
NNP	VB	NN	RB	VBD			
PRP	VBP	NNP	VB	VBN			
		VBP	VBP	VBP			
4	4	5	5	5	2	1	1

4,000 possible state sequences!

Assigning tags to a new sentence

Naïve solutions

1. List all possible sequences
   - Correct, but inefficient.
2. Move left to right through trellis and greedily choose best state t_i based t_{i-1} and w_i.
   - Fast, but not ensured to give *argmax* $p(x, y)$

Assigning tags to a new sentence

What if...

- If I knew the score of every sequence $t_1, \ldots, t_{n-1}$, I could reason easily about t_n

Assigning tags to a new sentence

What if...

- If I knew the score of every sequence $t_1, \ldots, t_{n-1}$, I could reason easily about t_n
- BUT my decision about t_n would only really depend on t_{n-1}!

Assigning tags to a new sentence

What if...

- If I knew the score of every sequence $t_1, \ldots, t_{n-1}$, I could reason easily about t_n
- BUT my decision about t_n would only really depend on t_{n-1}!
- So I really only need to know the score of the single best sequence ending in each t_{n-1}.

Assigning tags to a new sentence

What if...

- If I knew the score of every sequence $t_1, \ldots, t_{n-1}$, I could reason easily about t_n
- BUT my decision about t_n would only really depend on t_{n-1}!
- So I really only need to know the score of the single best sequence ending in each t_{n-1}.
- Recurrence!

Assigning tags to a new sentence

What if...

- If I knew the score of every sequence $t_1, \ldots, t_{n-1}$, I could reason easily about t_n
- BUT my decision about t_n would only really depend on t_{n-1}!
- So I really only need to know the score of the single best sequence ending in each t_{n-1}.
- Recurrence!
- We can precalculate everything before and keep the best scores.

Viterbi algorithm

Algorithm

- Instead of enumerating all possible tag sequences

Viterbi algorithm

Algorithm

- Instead of enumerating all possible tag sequences
- Viterbi makes use of the Markov assumption in the HMM.

Viterbi algorithm

Algorithm

- Instead of enumerating all possible tag sequences
- Viterbi makes use of the Markov assumption in the HMM.
- $argmax_{y \in Y} \prod_{i=1}^{n+1} p(y_i | y_{i-1}) \prod_{i=1}^{n} p(x_i | y_i)$
- This fact will make calculations faster.

Viterbi algorithm

Steps

Viterbi algorithm

Steps

- Initialize a path probability matrix *viterbi*$[N, T]$

Viterbi algorithm

Steps

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$

Viterbi algorithm

Steps

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$
- Recursion step:
  - for each timestep t from 2 to T do
    for each state s from 1 to S do
    $viterbi[s, t] = max\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

Viterbi algorithm

Steps

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$
- Recursion step:
  - for each timestep t from 2 to T do
    for each state s from 1 to S do
    $viterbi[s, t] = max\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
- Termination step:
  - bestPathProb $= max_{s=1}^{N}\ viterbi[s, T]$

Viterbi algorithm

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4E-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24
	1	2	3	4	5	6	7	8

Viterbi algorithm

However, this only gets us the best probability :/

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4E-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24
	1	2	3	4	5	6	7	8

Viterbi algorithm

To get best path

- Initialize a path probability matrix *viterbi*$[N, T]$

Viterbi algorithm

To get best path

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$
  - $backpointer[s, 1] \leftarrow 0$

Viterbi algorithm

To get best path

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$
  - $backpointer[s, 1] \leftarrow 0$
- Recursion step:
  - for each timestep t from 2 to T do

    for each state s from 1 to S do

    $viterbi[s, t] = max\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

    $backpointer[s, t] \leftarrow argmax\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

Viterbi algorithm

To get best path

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$
  - $backpointer[s, 1] \leftarrow 0$
- Recursion step:
  - for each timestep t from 2 to T do
    for each state s from 1 to S do
    $viterbi[s, t] = max\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s, t] \leftarrow argmax\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
- Termination step:
  - bestPathProb $= max_{s=1}^{N}\ viterbi[s, T]$
  - bestPathPointer $\leftarrow argmax_{s=1}^{N}\ viterbi[s, T]$

Viterbi algorithm

To get best path

- Initialize a path probability matrix $viterbi[N, T]$
- Initialization step:
  - $viterbi[s, 1] = \pi_s * b_s(o_1)$
  - $backpointer[s, 1] \leftarrow 0$
- Recursion step:
  - for each timestep t from 2 to T do

    for each state s from 1 to S do

    $viterbi[s, t] = max\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$

    $backpointer[s, t] \leftarrow argmax\ viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
- Termination step:
  - bestPathProb $= max_{s=1}^{N}\ viterbi[s, T]$
  - bestPathPointer $\leftarrow argmax_{s=1}^{N}\ viterbi[s, T]$

For best path, start at bestPathPointer and follow backpointer[].

Viterbi algorithm

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4E-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24

Viterbi algorithm

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4E-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24

Assigning POS tags

Tagging using HMMs

Assigning POS tags

Tagging using HMMs

- Easy to train (A and B parameters)
- Relatively good results.

Assigning POS tags

Tagging using HMMs

- Easy to train (A and B parameters)
- Relatively good results.
- Limited on types of elements that are used for prediction
  - Previous $n-1$ tags
  - The current word, given its tag

Assigning POS tags

Tagging using HMMs

- Easy to train (A and B parameters)
- Relatively good results.
- Limited on types of elements that are used for prediction
  - Previous $n-1$ tags
  - The current word, given its tag
- Difficulty modeling $b(word|tag)$
  - Difficult for unknown words
  - This becomes even more of a problem for morphologically complex languages
  - Etxe, etxea, etxeak, etxearen, etxeen, etxetik, etxeetatik, etc.

Models that use more context

Discriminative models are less constrained:

- Conditional Random Fields (CRF)
- Perceptron Tagger
- Neural networks

Models that use more context

General idea (CRF, Perceptron)

Models that use more context

General idea (CRF, Perceptron)

- Can easily use as context:
  - Three previous tags
  - Two previous words
  - morphological tag of two previous words
  - lemma of two previous words

Models that use more context

General idea (CRF, Perceptron)

- Can easily use as context:
  - Three previous tags
  - Two previous words
  - morphological tag of two previous words
  - lemma of two previous words
- $p(t_i|t_{i-2}, t_{i-1}, w_{i-2}, w_{i-1}, m_{i-2}, m_{i-1}, l_{i-2}, l_{i-1})$ instead of $p(t_i|t_{i-1}) * p(w_i|t_i)$

Features

Feature engineering

Features

Feature engineering

- The selection of features in discriminative models in the late 90's/early 2000's was equivalent to hyperparameter tuning with today's neural networks.
- Important for results $\rightarrow$ lots of time devoted but also a bit of a black art.

Features

Feature engineering

- The selection of features in discriminative models in the late 90's/early 2000's was equivalent to hyperparameter tuning with today's neural networks.
- Important for results $\rightarrow$ lots of time devoted but also a bit of a black art.
- Common features:

Features

Feature engineering

- The selection of features in discriminative models in the late 90's/early 2000's was equivalent to hyperparameter tuning with today's neural networks.
- Important for results $\rightarrow$ lots of time devoted but also a bit of a black art.
- Common features:
  - current, previous, next words
  - previous POS tags
  - For rare words: first and last character sequences
  - Whether the word contains a number, uppercase, or hyphen

Neural approaches

Many, many proposed models

- RNNs, CNNs, Transformers, etc.

Neural approaches

Many, many proposed models

- RNNs, CNNs, Transformers, etc.
- The state-of-the-art models generally have the following ideas:

Neural approaches

Many, many proposed models

- RNNs, CNNs, Transformers, etc.
- The state-of-the-art models generally have the following ideas:
    - Some kind of RNN for contextualization
    - Some kind of character-level information for unseen words
    - CRF layer at the end, to better model the interdependencies between output labels

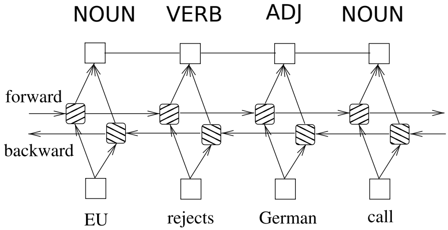


Figure 7: A BI-LSTM-CRF model.

Exercises in class (from "Natural Language Processing" J. Eisenstein)

- Given the following tables of emission and transition scores
- Calculate the best POS sequence for They can fish.

	they	*can*	*fish*
N	-2	-3	-3
V	-10	-1	-3

(a) Weights for emission features.

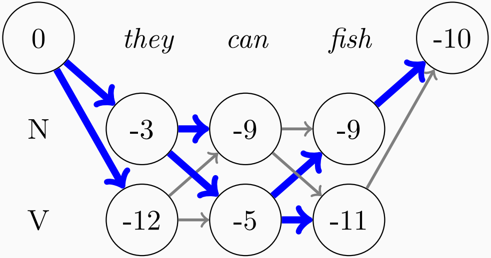
	N	V	$\blacklozenge$
$\lozenge$	-1	-2	$-\infty$
N	-3	-1	-1
V	-1	-3	-1

(b) Weights for transition features. The "from" tags are on the columns, and the "to" tags are on the rows.

Table 7.1: Feature weights for the example trellis shown in Figure 7.1. Emission weights from $\lozenge$ and $\blacklozenge$ are implicitly set to $-\infty$.

Exercises in class

- Given the following tables of emission and transition scores (note that these are log probabilities)
- Calculate the best POS sequence for They can fish.



Exercise 2 in class

Assuming a bigram transition model, calculate the most likely tag sequence for the aged bottle flies fast.

Emission probabilities:

	the	aged	bottle	flies	fast
ADJ	-inf	-2	-inf	-inf	-2
ADV	-inf	-inf	-inf	-inf	-1
DET	-1	-inf	-inf	-inf	-inf
NOUN	-inf	-inf	-2	-3	-5
VERB	-inf	-5	-2	-3	-2

Transition probabilities:

	ADJ	ADV	DET	NOUN	VERB	STOP
*	-1	-3	-3	-4	-4	-inf
ADJ	-3	-4	-inf	-2	-3	-3
ADV	-3	-3	-4	-4	-3	-3
DET	-2	-4	-inf	-0.5	-4	-inf
NOUN	-4	-4	-4	-3	-2	-3
VERB	-3	-2	-4	-3	-inf	-3