Sequence labeling

IN4080
Natural Language Processing

Yves Scherrer

Regularization

Logistic regression is prone to overfitting to the training data.

Regularization reduces overfitting by penalizing large weight values.

- No single feature/weight should override the others.

- L2 regularization: $R(w) = \sum_0^n w_i^2$
- L1 regularization: $R(w) = \sum_0^n |w_i|$
- Can be specified with the *penalty* and *C* parameters in Scikit-Learn.
  - See J&M chapter 5.7 for details

Linear classifiers overview

Overview

Naive Bayes	Perceptron	Logistic regression
Probabilistic	Non-probabilistic	Probabilistic
Generative	Discriminative	Discriminative
Batch training	Online training	Batch or online training

Generative vs discriminative classifiers

Consider a classification problem in which we want to learn to distinguish between elephants (y = 1) and dogs (y = 0), based on some features of an animal. Given a training set, a [**discriminative**] algorithm [...] (basically) tries to find a straight line—that is, a decision boundary—that separates the elephants and dogs. Then, to classify a new animal as either an elephant or a dog, it checks on which side of the decision boundary it falls, and makes its prediction accordingly.

Here's a different [**generative**] approach. First, looking at elephants, we can build a model of what elephants look like. Then, looking at dogs, we can build a separate model of what dogs look like. Finally, to classify a new animal, we can match the new animal against the elephant model, and match it against the dog model, to see whether the new animal looks more like the elephants or more like the dogs we had seen in the training set.

Andrew Ng, http://cs229.stanford.edu/notes/cs229-notes2.pdf

Generative classifiers

Underlying idea: a generative process

- For each document x:
    - Determine a class label according to $\hat{P}(y)$
    - Generate word counts according to the $\hat{P}(x_i|y)$
- How likely is it that the observed document x (i.e. its bag of words) with label y has been generated from the distributions $\hat{P}$?

Generative models solve a harder problem than just classification

- Modelling the generative process may not be useful if we just want to predict labels

Online vs batch learning

Batch learning algorithms process all the training data at once and create one model at the end of an epoch.

Online learning algorithms process one (or a few) training instances at a time, gradually refining the model.

- Intermediate models can be used at any time
- Easier to work with big data sets

Linear classifiers: Pros and cons

	Pros	Cons
Naive Bayes	• Easy to implement, fast to train • Probabilistic • Good for small data sets	• Often poor accuracy • Limited choice of features to avoid correlation
Perceptron	• Easy to implement • Typically high accuracy	• Not probabilistic • Hard to know when to stop • Lack of margin can lead to overfitting
Logistic regression	• Probabilistic (interpretable) • Typically more accurate than Naive Bayes	• Can overtrain on correctly labeled examples

Sequence labeling problems

Part-of-speech tagging

Profits soared at Boeing Co. , easily topping forecasts on Wall Street , as their CEO Alan Mulally announced first quarter results .

Profits/NOUN soared/VERB at/ADP Boeing/PROPN Co./PROPN ,/PUNCT easily/ADV topping/VERB forecasts/NOUN on/ADP Wall/PROPN Street/PROPN ,/PUNCT as/SCONJ their/DET CEO/PROPN Alan/PROPN Mulally/PROPN announced/VERB first/ADJ quarter/NOUN results/NOUN ./PUNCT

Part-of-speech tagging

Verticalized format:

Profits	NOUN
soared	VERB
at	ADP
Boeing	PROPN
Co.	PROPN
,	PUNCT
easily	ADV
topping	VERB
forecasts	NOUN
on	ADP
Wall	PROPN
Street	PROPN
,	PUNCT
as	SCONJ
their	DET
CEO	PROPN
Alan	PROPN
Mulally	PROPN
announced	VERB
first	ADJ
quarter	NOUN
results	NOUN
.	PUNCT

Part-of-speech tagging

What is a part-of-speech (POS)?

- A basic grammatical category (noun, verb, etc.)

How many parts-of-speech are there?

- Traditional (Greek) grammar:
  - **8**: noun, verb, pronoun, preposition, adverb, conjunction, participle, article
- English corpora from the 1990s:
  - **87, 35, 44**
- Universal POS tagset (2012):
  - **17**: ADJ, ADV, INTJ, NOUN, PROPN, VERB, ADP, AUX, CCONJ, DET, NUM, PART, PRON, SCONJ, PUNCT, SYM, X

Tagsets

There are various tagsets, even for a single language:

			Brown	Penn	Universal
	he	she	PPS	PRP	PRON
I			PPSS	PRP	PRON
me	him	her	PPO	PRP	PRON
my	his	her	PP$	PRP$	DET
mine	his	hers	PP$$	PRP$	PRON

Part-of-speech tagging

Why is POS tagging useful?

- Search for syntactic patterns in a corpus
- Information extraction: focus on content words
- Text-to-speech (how do we pronounce “lead”?)
- As a preprocessing step for other types of analysis

Sequence labeling

Goal:

- Predict a label for each element of a sequence
- I.e. predict a label for each word of a sentence

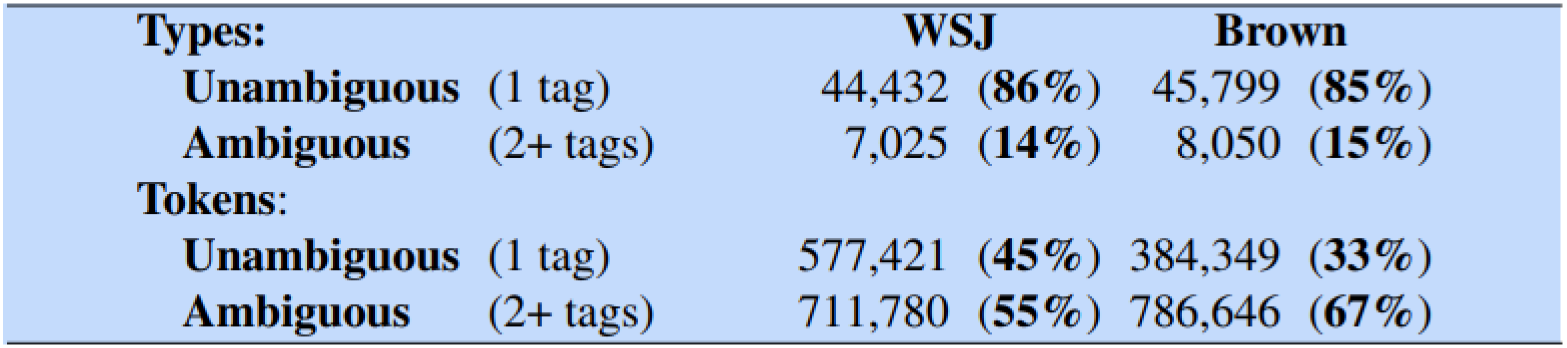
Easy cases:

- Unambiguous words (words seen together with a single label during training)

Difficult cases:

- Unknown words (words not seen during training)
- Ambiguous words (words seen together with several labels during training)

Ambiguity of tags



Types:		WSJ		Brown	
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Figure 8.4 Tag ambiguity in the Brown and WSJ corpora (Treebank-3 45-tag tagset).

J&M, 3rd edition, chapter 8.

Another task: Named entity recognition

Named entities:

- Person names
- Place names
- Date and time expressions
- Organization names

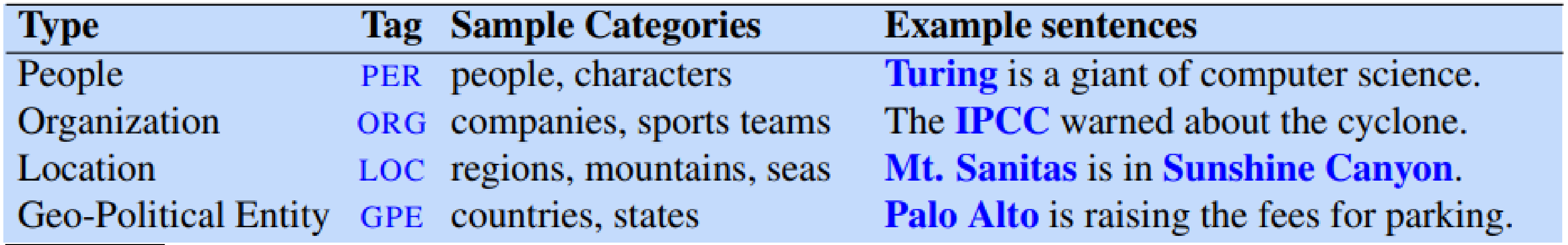
Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

Another task: Named entity recognition

Named entities often cover several words.
They can be represented with named brackets:

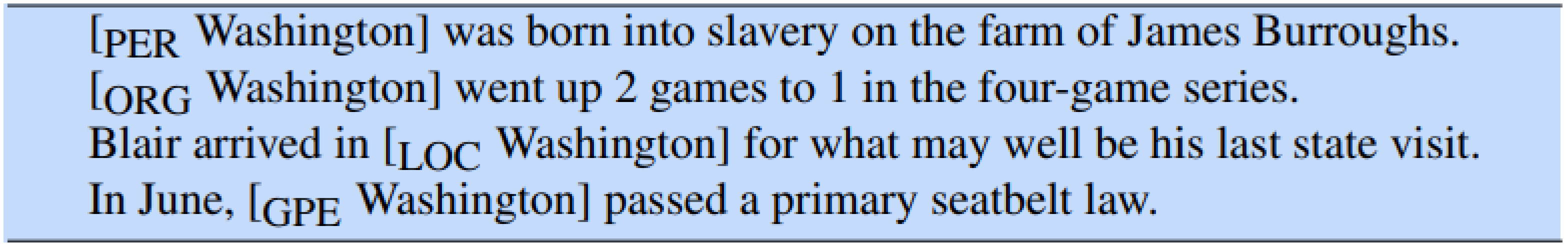
Citing high fuel prices, [$_{\text{ORG}}$ United Airlines] said [$_{\text{TIME}}$ Friday] it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. [$_{\text{ORG}}$ American Airlines], a unit of [$_{\text{ORG}}$ AMR Corp.], immediately matched the move, spokesman [$_{\text{PER}}$ Tim Wagner] said. [$_{\text{ORG}}$ United], a unit of [$_{\text{ORG}}$ UAL Corp.], said the increase took effect [$_{\text{TIME}}$ Thursday] and applies to most routes where it competes against discount carriers, such as [$_{\text{LOC}}$ Chicago] to [$_{\text{LOC}}$ Dallas] and [$_{\text{LOC}}$ Denver] to [$_{\text{LOC}}$ San Francisco].

Named entities



Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	**Mt. Sanitas** is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states	**Palo Alto** is raising the fees for parking.

Figure 8.5 A list of generic named entity types with the kinds of entities they refer to.



[$_{\text{PER}}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.

Figure 8.6 Examples of type ambiguities in the use of the name *Washington*.

J&M, SLP 3rd ed., ch. 8

BIO encoding

- B = Beginning of a new named entity
- I = Inside an existing named entity
- O = Outside of any named entity

Citing/O high/O fuel/O prices/O ,/O United/B-ORG Airlines/I-ORG said/O Friday/B-TIME it/O …

… such/O as/O Chicago/B-LOC to/O Dallas/B-LOC and/O Denver/B-LOC to/O San/B-LOC Francisco/I-LOC ./O

This looks exactly like a POS tagging task now.

Sequence labeling models

- Generative vs discriminative models
- Probabilistic vs non-probabilistic models
- Unigram models vs sequence models
- Greedy decoding vs exact search

Generative sequence models

Naïve Bayes

Hidden Markov Models

Naïve Bayes for sequence labeling

A first (and probably stupid) idea:

- Look at one word at a time (no context)
- Predict the most probable tag according to training data

Prediction function:

- $\hat{y} = \arg\max_{y \in Y}(P(y) \cdot P(x|y))$

Estimate probabilities from training data:

- $P(x|y) = \frac{Count(x,y)}{Count(y)} \qquad P(y) = \frac{Count(y)}{N}$

Let's try!

Training corpus:

dogs/N eat/V fish/N

cats/N eat/V mice/N

cats/N like/V fish/N

dogs/N fish/V

Compute priors:

$P(N) = \frac{7}{11}$ $P(V) = \frac{4}{11}$

Compute likelihoods:

$P(\text{dogs}|N) = \frac{2}{7}$ $P(\text{dogs}|V) = 0$

$P(\text{cats}|N) = \frac{2}{7}$ $P(\text{cats}|V) = 0$

$P(\text{eat}|N) = 0$ $P(\text{eat}|V) = \frac{2}{4}$

$P(\text{like}|N) = 0$ $P(\text{like}|V) = \frac{1}{4}$

$P(\text{fish}|N) = \frac{2}{7}$ $P(\text{fish}|V) = \frac{1}{4}$

$P(\text{mice}|N) = \frac{1}{7}$ $P(\text{mice}|V) = 0$

Let’s try!

Test sentence: dogs like fish

$$\hat{y} = \arg\max_{y \in Y}(P(y) \cdot P(x|y))$$

Computations:

$P(N|\text{dogs}) = \frac{7}{11} \cdot \frac{2}{7} = \frac{2}{11}$ $P(V|\text{dogs}) = \frac{4}{11} \cdot 0 = 0$ $\hat{y} = N$

$P(N|\text{like}) = \frac{7}{11} \cdot 0 = 0$ $P(V|\text{like}) = \frac{4}{11} \cdot \frac{1}{4} = \frac{1}{11}$ $\hat{y} = V$

$P(N|\text{fish}) = \frac{7}{11} \cdot \frac{2}{7} = \frac{2}{11}$ $P(V|\text{fish}) = \frac{4}{11} \cdot \frac{1}{4} = \frac{1}{11}$ $\hat{y} = N$

Let's try!

Test sentence: cats fish fish

$$\hat{y} = \arg\max_{y \in Y}(P(y) \cdot P(x|y))$$

Computations:

$P(N|\text{cats}) = \frac{7}{11} \cdot \frac{2}{7} = \frac{2}{11}$ $\quad$ $P(V|\text{cats}) = \frac{4}{11} \cdot 0 = 0$ $\quad$ $\hat{y} = N$

$P(N|\text{fish}) = \frac{7}{11} \cdot \frac{2}{7} = \frac{2}{11}$ $\quad$ $P(V|\text{fish}) = \frac{4}{11} \cdot \frac{1}{4} = \frac{1}{11}$ $\quad$ $\hat{y} = N$

$P(N|\text{fish}) = \frac{7}{11} \cdot \frac{2}{7} = \frac{2}{11}$ $\quad$ $P(V|\text{fish}) = \frac{4}{11} \cdot \frac{1}{4} = \frac{1}{11}$ $\quad$ $\hat{y} = N$

Naïve Bayes for sequence labeling

- Unambiguous words
  - Seems fine…
- Ambiguous words
  - Not great…
  - cats/N fish/N fish/N
- Unknown words
  - cats/N fish/N cyprinids/?
    - $P(N|\text{cyprinids}) = \frac{7}{11} \cdot 0 = 0$
    - $P(V|\text{cyprinids}) = \frac{4}{11} \cdot 0 = 0$ $\hat{y} = ???$
  - We can solve that with smoothing!

Dealing with unknown words

Option 1: add-one smoothing

Compute likelihoods:

$P(\text{dogs}|N) = \frac{2+1}{7+6}$ $P(\text{dogs}|V) = \frac{0+1}{4+6}$

$P(\text{cats}|N) = \frac{2+1}{7+6}$ $P(\text{cats}|V) = \frac{0+1}{4+6}$

$P(\text{eat}|N) = \frac{0+1}{7+6}$ $P(\text{eat}|V) = \frac{2+1}{4+6}$

$P(\text{like}|N) = \frac{0+1}{7+6}$ $P(\text{like}|V) = \frac{1+1}{4+6}$

$P(\text{fish}|N) = \frac{2+1}{7+6}$ $P(\text{fish}|V) = \frac{1+1}{4+6}$

$P(\text{mice}|N) = \frac{1+1}{7+6}$ $P(\text{mice}|V) = \frac{0+1}{4+6}$

Compute priors:

$P(N) = \frac{7+1}{11+2}$ $P(V) = \frac{4+1}{11+2}$

Test computations:

$$P(N|\text{cyprinids}) = \frac{7+1}{11+2} \cdot \frac{0+1}{7+6} = 0.047$$

$$P(V|\text{cyprinids}) = \frac{4+1}{11+2} \cdot \frac{0+1}{4+6} = 0.038$$

Dealing with unknown words

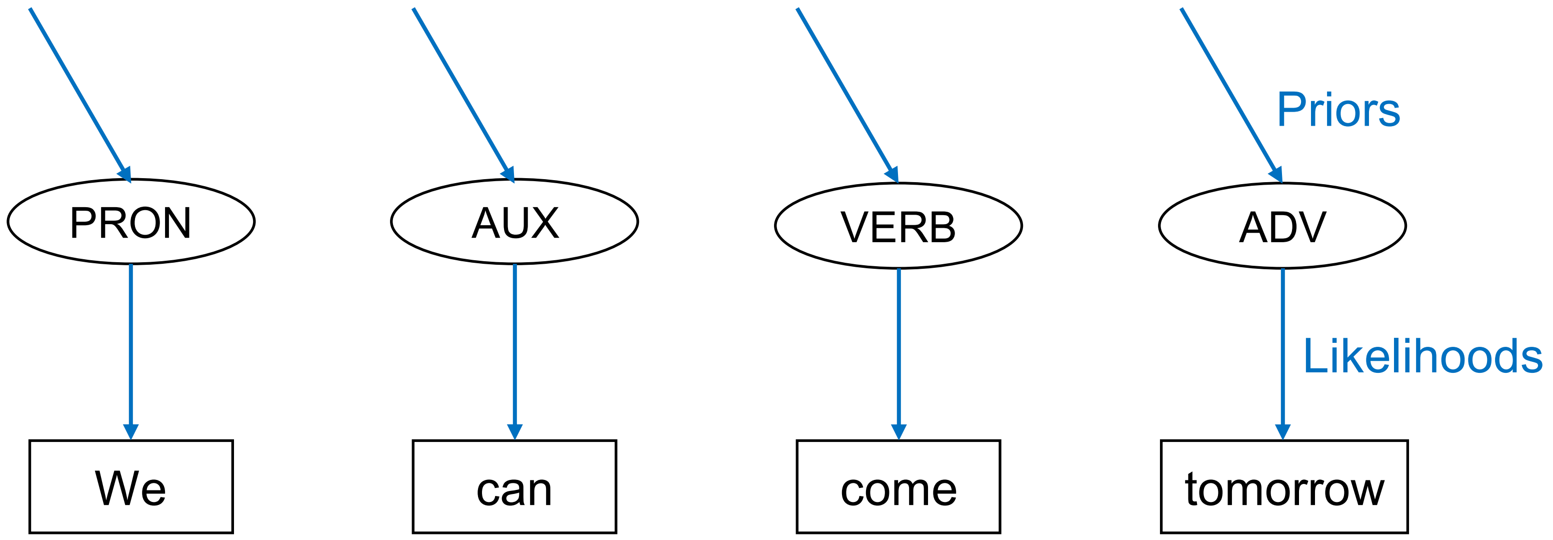
Option 2: use rare words classes

- Split vocabulary into two sets:
    - Rare words: words occurring (for example) < 5 times in the training data
    - Frequent words: all other words
- Replace all rare words in training data by _rare_ before computing probabilities
- Replace unknown words in test data by _rare_
    - $P(N|\text{cyprinids}) = P(N|\text{\_rare\_})$
    - $P(V|\text{cyprinids}) = P(V|\text{\_rare\_})$

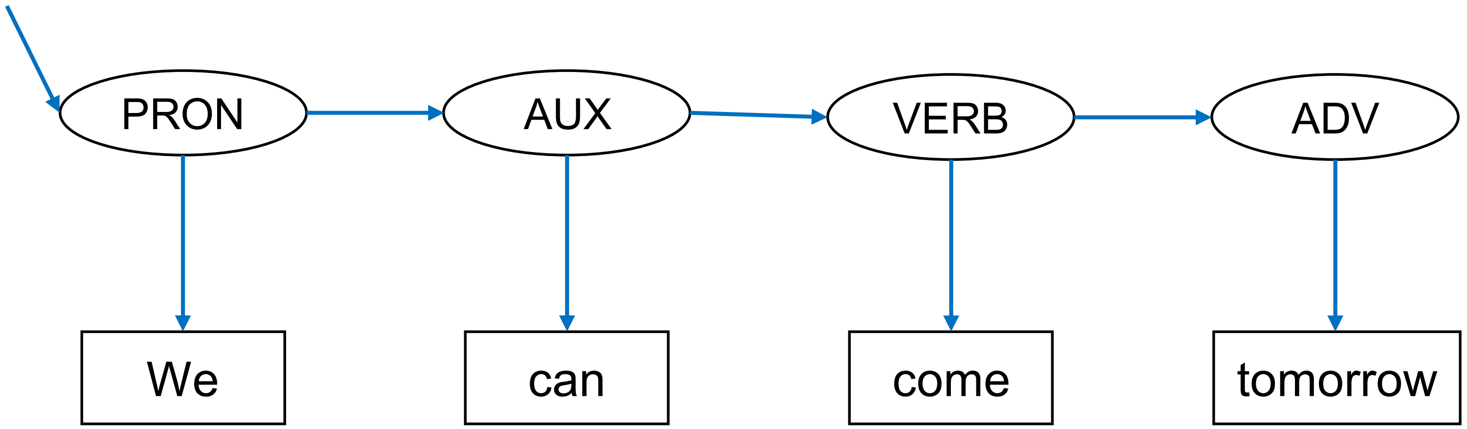
What about ambiguous words?

- Our Naive Bayes model is a **unigram model**
  - It looks at one word/label at a time
- It labels each word independently of its context:
  - We/PRON **can/AUX** come/VERB tomorrow/ADV ./PUNCT
  - The/DET trash/NOUN **can/AUX** is/VERB empty/ADJ ./PUNCT
  - Tomorrow/ADV empty/ADJ come/VERB **can/AUX** ./PUNCT
- It cannot handle ambiguous words by design
  - We have to change that design…

Generative unigram models



Generative bigram models



A simple Hidden Markov Model

Recall the Naïve Bayes prediction function:

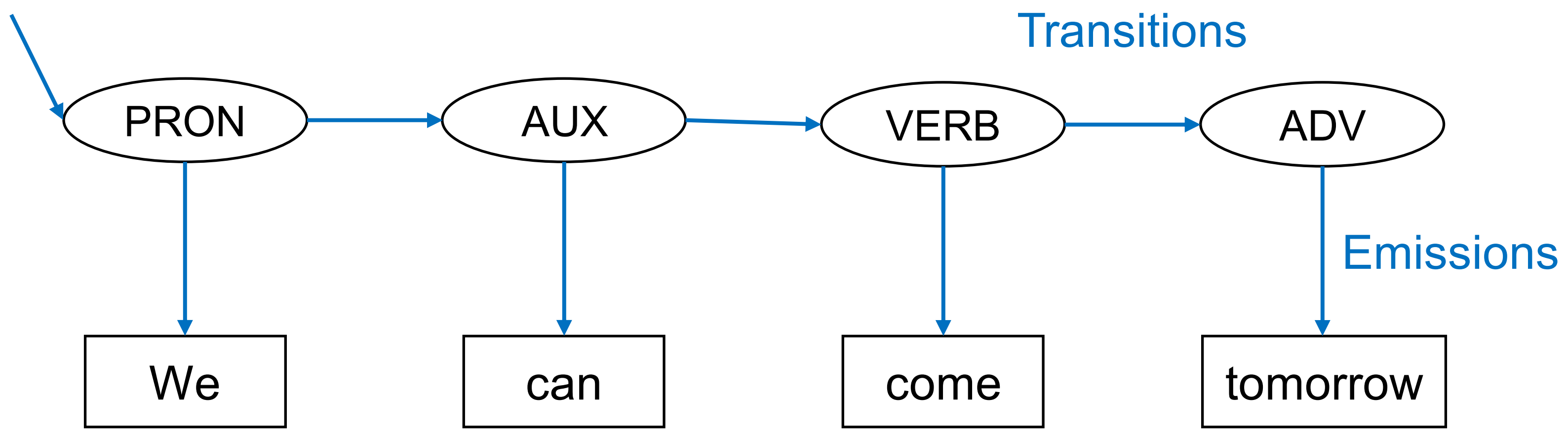
$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}(\underbrace{P(y_i)}_{\text{Priors}} \cdot \underbrace{P(x_i|y_i)}_{\text{Likelihoods}})$$

We can just replace the prior probability by a conditional probability:

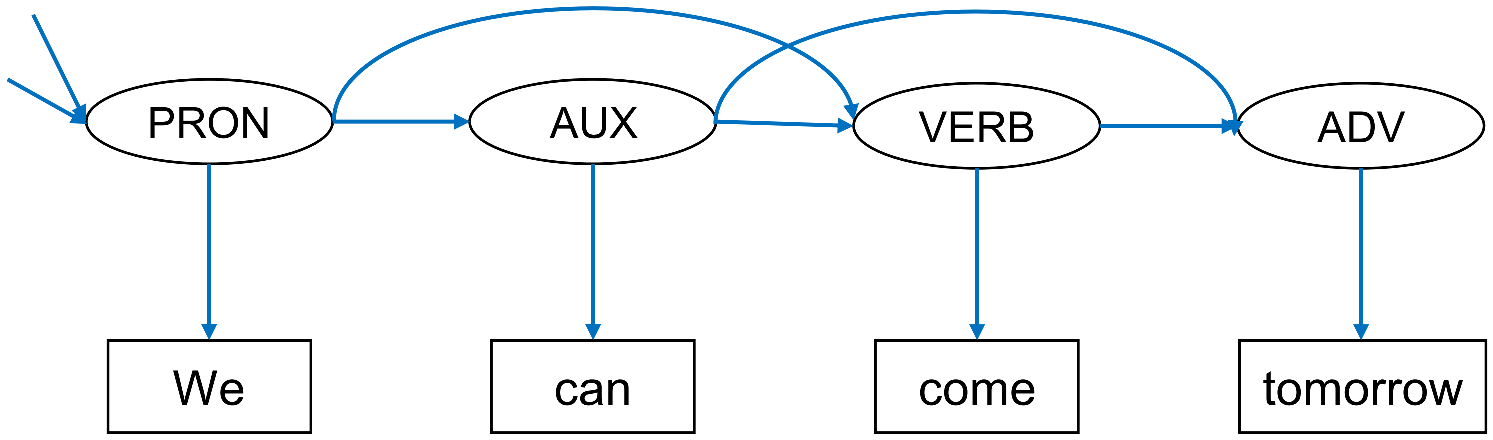
$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}(\underbrace{P(y_i|y_{i-1})}_{\text{Transitions}} \cdot \underbrace{P(x_i|y_i)}_{\text{Emissions}})$$

- This is a simple **Hidden Markov model**.

Bigram HMM


Transitions
PRON
AUX
VERB
ADV
Emissions
We
can
come
tomorrow

Trigram HMM



$$\hat{y}_i = \arg\max_{y_i \in Y}(P(y_i | y_{i-2}, y_{i-1}) \cdot P(x_i | y_i))$$

Different ways to look at the context

Option 1: Look at the neighboring words:

$$\hat{y}_i = \arg\max_{y_i \in Y} P(y_i | x_{i-1}, x_i, x_{i+1})$$

- Doesn't work with generative learning algorithms
- But works well with discriminative algorithms (later)

Option 2: Look at the previous tag decisions:

$$\hat{y}_i = \arg\max_{y_i \in Y} P(y_i | x_i, y_{i-1})$$

- Works well with generative learning algorithms
- This is the idea behind HMMs

A Hidden Markov Model

Training:

- For each x, y in training data:
  - Compute and store $P_E(x|y) = \frac{Count(x,y)}{Count(y)}$
- For each bigram y_{i-1}, y_i in training data:
  - Compute and store $P_T(y_i|y_{i-1}) = \frac{Count(y_{i-1},y_i)}{Count(y_{i-1})}$

We will need to apply a trick for the first word of each sentence…

Testing/Prediction (first try):

- For each sentence in test data:
  - For each position i in sentence:
    - Compute $\hat{y}_i = \arg\max_{y_i \in Y}(P_T(y_i|y_{i-1}) \cdot P_E(x_i|y_i))$
    - Store $\hat{y}_i$ to make it accessible for the next position

Example Training

Training corpus:

/* dogs/N eat/V fish/N

/* cats/N eat/V mice/N

/* cats/N like/V fish/N

/* dogs/N fish/V

Trick: add a dummy label * at the beginning of each sentence

Emissions:

$P_E(\text{dogs}\|N) = \frac{2}{7}$	$P_E(\text{dogs}\|V) = 0$
$P_E(\text{cats}\|N) = \frac{2}{7}$	$P_E(\text{cats}\|V) = 0$
$P_E(\text{eat}\|N) = 0$	$P_E(\text{eat}\|V) = \frac{2}{4}$
$P_E(\text{like}\|N) = 0$	$P_E(\text{like}\|V) = \frac{1}{4}$
$P_E(\text{fish}\|N) = \frac{2}{7}$	$P_E(\text{fish}\|V) = \frac{1}{4}$
$P_E(\text{mice}\|N) = \frac{1}{7}$	$P_E(\text{mice}\|V) = 0$

Transitions:

$$P_T(N|N) = \frac{0}{4} = 0$$

$$P_T(V|N) = \frac{4}{4} = 1$$

$$P_T(N|V) = \frac{3}{3} = 1$$

$$P_T(V|V) = \frac{0}{3} = 0$$

$$P_T(N|\ *) = \frac{4}{4} = 1$$

$$P_T(V|\ *) = \frac{0}{4} = 0$$

Example – Prediction

Test sentence: dogs like fish

$$\hat{y}_i = \arg\max_{y_i \in Y} (P_T(y_i | y_{i-1}) \cdot P_E(x_i | y_i))$$

Computations:

	dogs	like	fish
N	$1 \cdot \frac{2}{7} = 0.286$	$0 \cdot 0 = 0$	$1 \cdot \frac{2}{7} = 0.286$
V	$0 \cdot 0 = 0$	$1 \cdot \frac{1}{4} = 0.25$	$0 \cdot \frac{1}{4} = 0$
	$\hat{y} = N$	$\hat{y} = V$	$\hat{y} = N$

Due to the small number of examples, a lot of probabilities become 0. We can use add-one smoothing to avoid this.

Example Training

Training corpus:

/* dogs/N eat/V fish/N

/* cats/N eat/V mice/N

/* cats/N like/V fish/N

/* dogs/N fish/V

Smoothed emissions:

$$P(\text{dogs}|N) = \frac{2+1}{7+6} \quad P(\text{dogs}|V) = \frac{0+1}{4+6}$$

$$P(\text{cats}|N) = \frac{2+1}{7+6} \quad P(\text{cats}|V) = \frac{0+1}{4+6}$$

$$P(\text{eat}|N) = \frac{0+1}{7+6} \quad P(\text{eat}|V) = \frac{2+1}{4+6}$$

$$P(\text{like}|N) = \frac{0+1}{7+6} \quad P(\text{like}|V) = \frac{1+1}{4+6}$$

$$P(\text{fish}|N) = \frac{2+1}{7+6} \quad P(\text{fish}|V) = \frac{1+1}{4+6}$$

$$P(\text{mice}|N) = \frac{1+1}{7+6} \quad P(\text{mice}|V) = \frac{0+1}{4+6}$$

Smoothed transitions:

$$P_T(N|N) = \frac{0+1}{4+2}$$

$$P_T(V|N) = \frac{4+1}{4+2}$$

$$P_T(N|V) = \frac{3+1}{3+2}$$

$$P_T(V|V) = \frac{0+1}{3+2}$$

$$P_T(N|*) = \frac{4+1}{4+2}$$

$$P_T(V|*) = \frac{0+1}{4+2}$$

Example – Prediction

Test sentence: dogs like fish

$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}(P_T(y_i|y_{i-1}) \cdot P_E(x_i|y_i))$$

Computations:

	dogs	like	fish
N	$\frac{5}{6} \cdot \frac{3}{13} = 0.192$	$\frac{1}{6} \cdot \frac{1}{13} = 0.013$	$\frac{4}{5} \cdot \frac{3}{13} = 0.185$
V	$\frac{1}{6} \cdot \frac{1}{10} = 0.017$	$\frac{5}{6} \cdot \frac{2}{10} = 0.167$	$\frac{1}{5} \cdot \frac{2}{10} = 0.040$
	$\hat{y} = N$	$\hat{y} = V$	$\hat{y} = N$

No more zeros, but the predictions remain the same.

Another example

Test sentence: cats fish fish

$$\hat{y}_i = \underset{y_i \in Y}{\arg\max}(P_T(y_i|y_{i-1}) \cdot P_E(x_i|y_i))$$

Computations:

	cats	fish	fish
N	$\frac{5}{6} \cdot \frac{3}{13} = 0.192$	$\frac{1}{6} \cdot \frac{3}{13} = 0.038$	$\frac{4}{5} \cdot \frac{3}{13} = 0.185$
V	$\frac{1}{6} \cdot \frac{1}{10} = 0.017$	$\frac{5}{6} \cdot \frac{2}{10} = 0.167$	$\frac{1}{5} \cdot \frac{2}{10} = 0.040$
	$\hat{y} = N$	$\hat{y} = V$	$\hat{y} = N$

Seems to work fine for ambiguous words!

Computing the probabilities

- Now, we compute one argmax for each position:

$$\hat{y}_i = \arg\max_{y_i \in Y}(P_E(x_i|y_i) \cdot P_T(y_i|y_{i-1}))$$

- What if we want to get the joint probability of words and predicted labels?

$$P(x_{1..n}, \widehat{y_{1..n}}) = \prod_{i=1}^{n} \max_{y_i \in Y}(P_E(x_i|y_i) \cdot P_T(y_i|y_{i-1}))$$

  - Replace the argmax by max
  - Multiply the probabilities of each position
- We haven't done this multiplication yet...
  - It doesn't change the predictions, but it will become important later...

Example – Prediction

Test sentence: dogs like fish

Computations:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \prod_{i=1}^{n} \max_{y_i \in Y}(P_E(x_i|y_i) \cdot P_T(y_i|y_{i-1}))$$

	dogs	like	fish
N	$\frac{5}{6} \cdot \frac{3}{13} = 0.192$	$0.192 \cdot \frac{1}{6} \cdot \frac{1}{13} = 0.0025$	$0.0321 \cdot \frac{4}{5} \cdot \frac{3}{13} = 0.0059$
V	$\frac{1}{6} \cdot \frac{1}{10} = 0.017$	$0.192 \cdot \frac{5}{6} \cdot \frac{2}{10} = 0.0321$	$0.0321 \cdot \frac{1}{5} \cdot \frac{2}{10} = 0.0013$
	$\hat{y} = N$	$\hat{y} = V$	$\hat{y} = N$

Probabilities get smaller and smaller as we proceed, but the predictions remain the same.

Sequence labeling

Easy cases:

- Unambiguous words
  - Solved

Difficult cases:

- Unknown words
  - Solved with smoothing
- Ambiguous words
  - Solved by conditioning on previously predicted tag
  - Really???

What’s the problem with this approach?

On the basis of the training corpus below, can you find an example sentence that does not get the correct labels?

/* dogs/N eat/V fish/N

/* cats/N eat/V mice/N

/* cats/N like/V fish/N

/* dogs/N fish/V

Example Training

Training corpus:

/* dogs/N eat/V fish/N

/* cats/N eat/V mice/N

/* cats/N like/V fish/N

/* dogs/N fish/V

Smoothed emissions:

$P(\text{dogs}|N) = \frac{2+1}{7+6}$ $P(\text{dogs}|V) = \frac{0+1}{4+6}$

$P(\text{cats}|N) = \frac{2+1}{7+6}$ $P(\text{cats}|V) = \frac{0+1}{4+6}$

$P(\text{eat}|N) = \frac{0+1}{7+6}$ $P(\text{eat}|V) = \frac{2+1}{4+6}$

$P(\text{like}|N) = \frac{0+1}{7+6}$ $P(\text{like}|V) = \frac{1+1}{4+6}$

$P(\text{fish}|N) = \frac{2+1}{7+6}$ $P(\text{fish}|V) = \frac{1+1}{4+6}$

$P(\text{mice}|N) = \frac{1+1}{7+6}$ $P(\text{mice}|V) = \frac{0+1}{4+6}$

Smoothed transitions:

$P_T(N|N) = \frac{0+1}{4+2}$

$P_T(V|N) = \frac{4+1}{4+2}$

$P_T(N|V) = \frac{3+1}{3+2}$

$P_T(V|V) = \frac{0+1}{3+2}$

$P_T(N|*) = \frac{4+1}{4+2}$

$P_T(V|*) = \frac{0+1}{4+2}$

Example – Prediction

Test sentence: fish dogs like cats

Computations:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \prod_{i=1}^{n} \max_{y_i \in Y}(P_E(x_i|y_i) \cdot P_T(y_i|y_{i-1}))$$

	fish	dogs	like	cats
N	$\frac{5}{6} \cdot \frac{3}{13} = \mathbf{0.192}$	$0.192 \cdot \frac{1}{6} \cdot \frac{3}{13} = 0.0074$	$0.016 \cdot \frac{4}{5} \cdot \frac{1}{13} = \mathbf{0.00098}$	$0.00098 \cdot \frac{1}{6} \cdot \frac{3}{13} = 0.000038$
V	$\frac{1}{6} \cdot \frac{2}{10} = 0.033$	$0.192 \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{0.016}$	$0.016 \cdot \frac{1}{5} \cdot \frac{2}{10} = 0.00064$	$0.00098 \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{0.000082}$
	N	V	N	V

What’s the problem?

- In our example, the transitions are strongly biased towards N – V – N – V ... sequences
  - When transition and emission probabilities contradict each other, the “wrong one” may win
- The model has trouble getting out of a bad situation:
  - Once it makes an error, the rest of the sentence is likely to be erroneous as well
- What would be the result of the correct labeling?

Example – Prediction

Test sentence: fish dogs like cats

Computations:

This probability is much higher than the one we found for N V N V: 0.000082

	fish	dogs	like	cats
N	$\frac{5}{6} \cdot \frac{3}{13} = 0.192$	$0.192 \cdot \frac{1}{6} \cdot \frac{3}{13} = 0.0074$	$0.0074 \cdot \frac{1}{6} \cdot \frac{1}{13} = 0.00009$	$0.0012 \cdot \frac{4}{5} \cdot \frac{3}{13} = 0.00022$
V	$\frac{1}{6} \cdot \frac{2}{10} = 0.033$	$0.192 \cdot \frac{5}{6} \cdot \frac{1}{10} = 0.016$	$0.0074 \cdot \frac{5}{6} \cdot \frac{2}{10} = 0.0012$	$0.0012 \cdot \frac{1}{5} \cdot \frac{1}{10} = 0.000024$
	N	N	V	N

What’s the problem?

- One wrong decision (dogs/V) led the model on a suboptimal path from which it could not recover.
- If we “help” the model with this decision, it gets the rest right, and produces a much higher overall probability.

- The model takes decisions too early and cannot go back to correct a bad decision.

What's the problem?

What we actually compute:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \prod_{i=1}^{n} \max_{y_i \in Y} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

Check all labels for one position

What we really should compute:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^n} \prod_{i=1}^{n} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

Check all combinations of labels for the entire sentence

The results are not the same...

What’s the problem?

What we actually compute:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \prod_{i=1}^{n} \max_{y_i \in Y} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

- Greedy/approximate inference/decoding

What we really should compute:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^n} \prod_{i=1}^{n} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

- Exact inference/decoding

The results are not the same...

What’s the problem?

What we actually compute:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \prod_{i=1}^{n} \max_{y_i \in Y} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1})\right)$$

- Example: sentence of 20 words, 17 labels (UPOS)
- $20 \cdot 17 = 340$ computations

What we really should compute:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^n} \prod_{i=1}^{n} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1})\right)$$

- Example: sentence of 20 words, 17 labels (UPOS)
- $17^{20} = \sim 4\,000\,000\,000\,000\,000\,000\,000\,000$ computations! That looks bad...

Exact inference

$$P(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^n} \prod_{i=1}^{n} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

- That's one big computation for the whole sentence
- This computation is intractable
  - But there are two tricks!
  - The **Viterbi algorithm** includes these two tricks

Trick 1: Decomposing the product

A lot of repetitions!

Assume 4 words, 2 tags (A, B). That's $2^4 = 16$ computations, 112 operations:

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_A) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_A)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_A) \cdot P_T(y_A|y_B)$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_3|y_B) \cdot P_T(y_B|y_B) \cdot P_E(x_4|y_B) \cdot P_T(y_B|y_B)$

Trick 1: Decomposing the product

Let’s proceed one position at a time and save the intermediate results:

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) = \pi_A$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) = \pi_B$

- $\pi_A \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) = \pi_{AA}$
- $\pi_A \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) = \pi_{AB}$
- $\pi_B \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) = \pi_{BA}$
- $\pi_B \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) = \pi_{BB}$

- $\pi_{AA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{AAA}$
- …

That’s $2 + 4 + 8 + 16 = 30$ computations, but only $2 + 8 + 24 + 48 = 82$ multiplication operations.

Trick 2: The Markov assumption

Ultimately, we are interested in the sequence with the maximum probability. We can identify uninteresting paths and skip them.

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) = \pi_A$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) = \pi_B$

- $\pi_A \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) = \pi_{AA}$
- $\pi_A \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) = \pi_{AB}$
- $\pi_B \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) = \pi_{BA}$
- $\pi_B \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) = \pi_{BB}$

If $\pi_{AA} < \pi_{BA}$, then $\pi_{AAi} < \pi_{BAi}$ for any i.

- ~~$\pi_{AA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{AAA}$~~
- $\pi_{BA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{BAA}$

We can skip all computations starting with π_{AA}.

Trick 2: The Markov assumption

The prediction formula only depends on the previous label, not on all labels back to y_0:

$$P(x_{1..n}, \widehat{y_{1..n}}) = \max_{y_{1..n} \in Y^*} \prod_{i=1}^{n} \left(P_E(x_i | y_i) \cdot P_T(y_i | y_{i-1}) \right)$$

This is called the (bigram) **Markov assumption**.

- At each position, we have to consider each label and each path from a previous label.
- But there is only one best path towards that previous label.
- The number of paths to consider does not grow exponentially, but remains at $|Y|^2$ at each position.

Trick 2: The Markov assumption

Ultimately, we are interested in the sequence with the maximum probability. We can identify uninteresting paths and skip them.

- $P_E(x_1|y_A) \cdot P_T(y_A|y_*) = \pi_A$
- $P_E(x_1|y_B) \cdot P_T(y_B|y_*) = \pi_B$

- $\pi_A \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_A) = \pi_{AA}$
- $\pi_A \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_A) = \pi_{AB}$
- $\pi_B \cdot P_E(x_2|y_A) \cdot P_T(y_A|y_B) = \pi_{BA}$
- $\pi_B \cdot P_E(x_2|y_B) \cdot P_T(y_B|y_B) = \pi_{BB}$

- ~~$\pi_{AA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{AAA}$~~
- $\pi_{BA} \cdot P_E(x_3|y_A) \cdot P_T(y_A|y_A) = \pi_{BAA}$

That's $2 + 4 + 4 + 4 = 14$ computations and $2 + 8 + 8 + 8 = 26$ operations.

The Viterbi algorithm

HMM prediction with the two tricks is known as the **Viterbi algorithm**:

- At each position i, compute the probability of each possible path from position $i - 1$
- Store intermediate computations in a table and remove low-probability paths according to the Markov assumption

The Viterbi algorithm

Setup		Max computations	Multiplications
m words, n labels	Brute force	m^n	$m^n \cdot 2 \cdot m$
	Viterbi	$m \cdot n^2$	$m \cdot n^2 \cdot 2$
4 words, 2 labels	Brute force	$2^4 = 16$	$16 \cdot 2 \cdot 4 = 122$
	Viterbi	$4 \cdot (2^2) = 16$	$16 \cdot 2 = 32$
5 words, 2 labels	Brute force	$2^5 = 32$	$32 \cdot 2 \cdot 5 = 320$
	Viterbi	$5 \cdot (2^2) = 20$	$20 \cdot 2 = 40$
6 words, 2 labels	Brute force	$2^6 = 64$	$64 \cdot 2 \cdot 6 = 768$
	Viterbi	$6 \cdot (2^2) = 24$	$24 \cdot 2 = 48$
20 words, 17 labels	Brute force	$17^{20} = 4e24$	$4e24 \cdot 2 \cdot 17$ $= 1.4e26$
	Viterbi	$20 \cdot (17^2) = 5780$	$5780 \cdot 2 = 11560$

In practice, a bit less because the first position of the sentence has fewer options.

Example

Test sentence: fish dogs like cats

Computations:

	fish	dogs	like	cats
N	$1\cdot\frac{5}{6}\cdot\frac{3}{13}=\mathbf{\frac{5}{26}}$	$\frac{5}{26}\cdot\frac{1}{6}\cdot\frac{3}{13}=\mathbf{\frac{5}{676}}$ $\cancel{\frac{1}{30}\cdot\frac{4}{5}\cdot\frac{3}{13}=\frac{2}{325}}$	$\cancel{\frac{5}{676}\cdot\frac{1}{6}\cdot\frac{1}{13}=\frac{5}{52728}}$ $\frac{5}{312}\cdot\frac{4}{5}\cdot\frac{1}{13}=\mathbf{\frac{1}{1014}}$	$\cancel{\frac{1}{1014}\cdot\frac{1}{6}\cdot\frac{3}{13}=\frac{1}{26364}}$ $\frac{5}{4056}\cdot\frac{4}{5}\cdot\frac{3}{13}=\mathbf{\frac{1}{4394}}$
V	$1\cdot\frac{1}{6}\cdot\frac{2}{10}=\mathbf{\frac{1}{30}}$	$\frac{5}{26}\cdot\frac{5}{6}\cdot\frac{1}{10}=\mathbf{\frac{5}{312}}$ $\cancel{\frac{1}{30}\cdot\frac{1}{5}\cdot\frac{1}{10}=\frac{1}{1500}}$	$\frac{5}{676}\cdot\frac{5}{6}\cdot\frac{2}{10}=\mathbf{\frac{5}{4056}}$ $\cancel{\frac{5}{312}\cdot\frac{1}{5}\cdot\frac{2}{10}=\frac{1}{1560}}$	$\frac{1}{1014}\cdot\frac{5}{6}\cdot\frac{1}{10}=\mathbf{\frac{1}{12168}}$ $\cancel{\frac{5}{4056}\cdot\frac{1}{5}\cdot\frac{1}{10}=\frac{1}{40560}}$

Example

Test sentence: fish dogs like cats

Computations:

	fish	dogs	like	cats
N	$\boxed{1 \cdot \frac{5}{6} \cdot \frac{3}{13} = \mathbf{\frac{5}{26}}}$	$\boxed{\frac{5}{26} \cdot \frac{1}{6} \cdot \frac{3}{13} = \mathbf{\frac{5}{676}}}$ $\cancel{\frac{1}{30} \cdot \frac{4}{5} \cdot \frac{3}{13} = \frac{2}{325}}$	$\cancel{\frac{5}{676} \cdot \frac{1}{6} \cdot \frac{1}{13} = \frac{5}{52728}}$ $\frac{5}{312} \cdot \frac{4}{5} \cdot \frac{1}{13} = \mathbf{\frac{1}{1014}}$	$\cancel{\frac{1}{1014} \cdot \frac{1}{6} \cdot \frac{3}{13} = \frac{1}{26364}}$ $\boxed{\frac{5}{4056} \cdot \frac{4}{5} \cdot \frac{3}{13} = \mathbf{\frac{1}{4394}}}$
V	$1 \cdot \frac{1}{6} \cdot \frac{2}{10} = \mathbf{\frac{1}{30}}$	$\frac{5}{26} \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{\frac{5}{312}}$ $\cancel{\frac{1}{30} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{1500}}$	$\boxed{\frac{5}{676} \cdot \frac{5}{6} \cdot \frac{2}{10} = \mathbf{\frac{5}{4056}}}$ $\cancel{\frac{5}{312} \cdot \frac{1}{5} \cdot \frac{2}{10} = \frac{1}{1560}}$	$\frac{1}{1014} \cdot \frac{5}{6} \cdot \frac{1}{10} = \mathbf{\frac{1}{12168}}$ $\cancel{\frac{5}{4056} \cdot \frac{1}{5} \cdot \frac{1}{10} = \frac{1}{40560}}$
	N	N	V	N

The Viterbi algorithm

Training:

- For each x, y in training data:
  - Compute and store $P_E(x|y) = \frac{Count(x,y)}{Count(y)}$
- For each bigram y_{i-1}, y_i in training data:
  - Compute and store $P_T(y_i|y_{i-1}) = \frac{Count(y_{i-1},y_i)}{Count(y_{i-1})}$

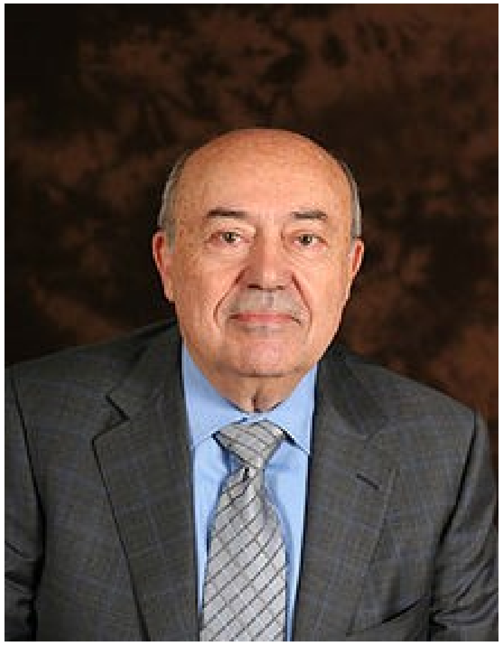
Same as before (with or without smoothing)

Testing/Prediction:

- Fill a $m \times y$ table position by position
- When reaching the end of the sentence, go back to find the optimal label sequence

Credits

Andrei Markov,
1856-1922



Andrea/Andrew Viterbi,
*1935

Readings

- Jurafsky & Martin, chapter 8 (up to 8.4)

- Jurafsky & Martin, appendix A
  - Implementation details of the Viterbi algorithm