Noisy Channel and Hidden Markov Models

Natural Language Processing
CS 4120/6120—Spring 2016
Northeastern University

David Smith
with material from Jason Eisner & Andrew McCallum
One thing I wanted to ask you about is this. A most serious problem, for UNESCO and for the constructive and peaceful future of the planet, is the problem of translation, as it unavoidably affects the communication between peoples. Huxley has recently told me that they are appalled by the magnitude and the importance of the translation job.

Recognizing fully, even though necessarily vaguely, the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate. Even if it would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but intelligible) result, it would seem to me worth while.

Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: “This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”
Word Segmentation

theprophetssaidtothecity

- What does this say?
  - And what other words are substrings?

- Given $L$ = a "lexicon" FSA that matches all English words.
- How to apply to this problem?
- What if Lexicon is weighted?
- From unigrams to bigrams?
- Smooth $L$ to include unseen words?
Spelling correction

- Spelling correction also needs a lexicon \( L \).
- But there is distortion ...
  - Let \( T \) be a transducer that models common typos and other spelling errors
    - \( \text{ance} \to \text{ence} \) (deliverance, ...)
    - \( e \to \varepsilon \) (deliverance, ...)
    - \( \varepsilon \to e \) / Cons _ Cons (athlete, ...)
    - \( rr \to r \) (embarrass occurrence, ...)
    - \( ge \to dge \) (privilege, ...)
    - etc.
  - Now what can you do with \( L \) o. \( T \) ?
- Should \( T \) and \( L \) have probabilities?
- Want \( T \) to include “all possible” errors ...
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$
Noisy Channel Model

real language \( X \)

noisy channel \( X \rightarrow Y \)

yucky language \( Y \)

want to recover \( X \) from \( Y \)

- correct spelling
- typos
- misspelling
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$

(lexicon space)*
delete spaces
text w/o spaces
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$
Noisy Channel Model

real language \( X \)

noisy channel \( X \rightarrow Y \)

yucky language \( Y \)

want to recover \( X \) from \( Y \)

(language model)

(acoustic model)

(lexicon space)*

pronunciation

speech
Noisy Channel Model

real language $X$

noisy channel $X \to Y$

yucky language $Y$

want to recover $X$ from $Y$
Noisy Channel Model

real language \( X \) → noisy channel \( X \rightarrow Y \) → yucky language \( Y \)

language model

translation model

“target” language

“source” language

want to recover \( X \) from \( Y \)
Noisy Channel Model

real language \( X \)

noisy channel \( X \rightarrow Y \)

yucky language \( Y \)

want to recover \( X \) from \( Y \)
Noisy Channel Model

real language $X$

probabilistic CFG

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

$p(X)$

$\ast$

$p(Y \mid X)$

$=$

$p(X,Y)$
Noisy Channel Model

\[ p(X) \ast p(Y | X) = p(X,Y) \]

want to recover \( x \in X \) from \( y \in Y \)
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

$p(X)$

$p(Y \mid X)$

$\ast$

$= p(X,Y)$

want to recover $x \in X$ from $y \in Y$

choose $x$ that maximizes $p(x \mid y)$ or equivalently $p(x,y)$
Noisy Channel Model

\[ p(X) \ast \]

\[ p(Y \mid X) \]

\[ = \]

\[ p(X,Y) \]
Noisy Channel Model

\[ p(X) \]

\[ p(Y | X) \]

\[ p(X, Y) = a \cdot \frac{0.7}{a} \cdot b \cdot \frac{0.3}{b} \]
Noisy Channel Model

\[ p(X) \]
\[ p(Y | X) \]
\[ p(X, Y) \]

\[ \begin{align*}
& a : a / 0.7 \\
& a : C / 0.1 \\
& a : D / 0.9
\end{align*} \quad \begin{align*}
& b : b / 0.3 \\
& b : C / 0.8 \\
& b : D / 0.2
\end{align*} \]

\[ p(X) \times p(Y | X) = p(X, Y) \]
Noisy Channel Model

\[ p(X) \]
\[ p(Y | X) \]
\[ p(X, Y) \]
Noisy Channel Model

\[
p(X) \ast p(Y \mid X) = p(X,Y)
\]
Noisy Channel Model

\[ p(X) \]

\[ p(Y | X) \]

\[ p(X,Y) \]

\[ = \]

\[ a:a/0.7 \]
\[ b:b/0.3 \]

\[ a:C/0.1 \]
\[ b:C/0.8 \]

\[ = \]

\[ a:D/0.9 \]
\[ b:D/0.2 \]

\[ a:C/0.07 \]
\[ b:C/0.24 \]

\[ = \]

\[ a:D/0.63 \]
\[ b:D/0.06 \]

Note \( p(x,y) \) sums to 1.
Noisy Channel Model

\[ p(X,Y) = a:D/0.9 \]
\[ p(Y|X) = a:C/0.1 \]
\[ b:C/0.8 \]
\[ b:D/0.2 \]
\[ o. \]
\[ a:a/0.7 \]
\[ b:b/0.3 \]
\[ p(X) * \]
\[ p(Y|X) = p(X,Y) \]
\[ a:C/0.07 \]
\[ b:C/0.24 \]
\[ a:D/0.63 \]
\[ b:D/0.06 \]

Note \( p(x,y) \) sums to 1.
Suppose \( y=\text{“C”}; \) what is best \( \text{“x”}? \)
Noisy Channel Model

\[
\begin{align*}
\mathbf{p}(X) &= a:D/0.9 + a:C/0.1 & b:C/0.8 + b:D/0.2 \\
\mathbf{p}(Y | X) &= a:a/0.7 + a:C/0.1 & b:b/0.3 + b:C/0.8 \\
\mathbf{p}(X,Y) &= a:C/0.07 + a:D/0.63 & b:C/0.24 + b:D/0.06 \\
\end{align*}
\]

Suppose \( y = "C" \); what is best \( x \)?
Noisy Channel Model

\[ p(X) \ast p(Y \mid X) = p(X, y) \]
Noisy Channel Model

\[
p(X) = a:D/0.9 \times a:C/0.1 \times b:C/0.8 \times b:D/0.2 \\
= a:C/0.07 \times b:C/0.24 = p(X, y)
\]

restrict just to paths compatible with output “C”
Noisy Channel Model

\[ p(X) \]

\[ p(Y \mid X) \]

\[ p(X, y) \]

restrict just to paths compatible with output “C”

\[ p(X) \]

\[ p(Y \mid X) \]

\[ (Y=y)? \]

\[ = \]

\[ p(X, y) \]
Noisy Channel Model

\[ p(X) \]

\[ p(Y \mid X) \]

\[ p(X, y) \]

\[ a:D/0.9 \]

\[ a:C/0.1 \]

\[ b:C/0.8 \]

\[ b:D/0.2 \]

restrict just to paths compatible with output "C"

\[ p(X) \]

\[ * \]

\[ p(Y \mid X) \]

\[ * \]

\[ (Y=y)? \]

\[ = \]

\[ p(X, y) \]

best path
Let Lexicon be a machine that matches all Turkish words

- Same problem as word segmentation (in, e.g., Chinese)
- Just at a lower level: morpheme segmentation

Turkish word: uygarlaştıramadıklarımızdan mıssınızcasına

= uygar+la+ş+tı+ra+ma+dık+lä+ri+mız+dan+miş+sınız+ca+sı+na

(behaving) as if you are among those whom we could not cause to become civilized

- Some constraints on morpheme sequence: bigram probs
- Generative model – concatenate then fix up joints
  - stop + -ing = stopping,  fly + -s = flies,  vowel harmony
  - Use a cascade of transducers to handle all the fixups

- But this is just morphology!
- Can use probabilities here too (but people often don’t)
Edit Distance Transducer

- O(k) deletion arcs
- O(k) insertion arcs
- O(k) no-change arcs
- O(k^2) substitution arcs
Stochastic Edit Distance Transducer

Likely edits = high-probability arcs
Stochastic Edit Distance Transducer

clarac:
  .0.
  a:ε
  b:ε
  a:b
  ε:a
  ε:b
  b:b
  a:a
  .0.
caca
Stochastic Edit Distance Transducer

Best path (by Dijkstra’s algorithm)
Speech Recognition by FST Composition
(Pereira & Riley 1996)

- **trigram language model**: \( p(\text{word seq}) \)
- **pronunciation model**: \( p(\text{phone seq} | \text{word seq}) \)
- **acoustic model**: \( p(\text{acoustics} | \text{phone seq}) \)
Speech Recognition by FST Composition
(Pereira & Riley 1996)

- **trigram language model**
  - \( p(\text{word seq}) \)
- **pronunciation model**
  - \( p(\text{phone seq} | \text{word seq}) \)
- **acoustic model**
  - \( p(\text{acoustics} | \text{phone seq}) \)
- **observed acoustics**
Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model

\[ p(\text{word seq}) \]

\[ p(\text{phone seq} \mid \text{word seq}) \]

\[ p(\text{acoustics} \mid \text{phone seq}) \]
Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model

\[ p(\text{word seq}) \]

\[ p(\text{phone seq} \mid \text{word seq}) \]

\[ p(\text{acoustics} \mid \text{phone seq}) \]

\[ \text{CAT:}k \, \text{æt} \]

\[ \text{æ:} \]
Transliteration
(Knight & Graehl, 1998)

Angela Johnson
アンジラ・ジョンソン
(a n jira jyo n so n)

New York Times
ニューヨーク・タイムズ
(nyu u yo o ku ta i mu zu)

ice cream
アイスクリーム
(a i su ku rii mu)

Omaha Beach
オマハビーチ
(omahabiitchi)

pro soccer
プロサッカー
(purosakkaa)

Tonya Harding
トーニャ・ハードィング
(toonya haadingu)

ramp
ランプ
(ranpu)

lamp
ランプ
(ranpu)

casual fashion
カジュアルヒアッション
(kajyuaruhashshyon)

team leader
チームリーダー
(chiimuriidaa)

1. \(P(w)\) — generates written English word sequences.
2. \(P(e|w)\) — pronounces English word sequences.
3. \(P(j|e)\) — converts English sounds into Japanese sounds.
4. \(P(k|j)\) — converts Japanese sounds to katakana writing.
5. \(P(o|k)\) — introduces misspellings caused by optical character recognition (OCR).
Part-of-Speech Tagging
Bigram LM as FSM

The quick brown fox jumped
Bigram LM as FSM

V states
Bigram LM as FSM

- $V$ states
- $O(V^2)$ arcs (& parameters)
Bigram LM as FSM

V states

What about a trigram model?

O(V^2) arcs (& parameters)
Bigram LM as FSM

- V states
- $O(V^2)$ arcs (& parameters)
- What about a trigram model?
- What about backoff?
Grammatical Categories

- “Parts of speech” (partes orationis)
  - Some Cool Kids call them “word classes”
- Folk definitions
  - Nouns: people, places, concepts, things, ...
  - Verbs: expressive of action
  - Adjectives: properties of nouns
- In linguistics, defined by role in syntax

The \{sad, intelligent, green, fat, \ldots\} one is in the corner.

“Substitution test”
The Tagging Task
The Tagging Task

Input: the lead paint is unsafe
The Tagging Task

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj
The Tagging Task

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj
The Tagging Task

Input: the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj

- Uses:
The Tagging Task

Input:  the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- Uses:
  - text-to-speech (how do we pronounce “lead”?)
The Tagging Task

Input: the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj

- Uses:
  - text-to-speech (how do we pronounce “lead”?)
  - can write regexps like (Det) Adj* N+ over the output
The Tagging Task

Input:  the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj

- Uses:
  - text-to-speech (how do we pronounce “lead”?)
  - can write regexps like (Det) Adj* N+ over the output
  - preprocessing to speed up parser (but a little dangerous)
The Tagging Task

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- Uses:
  - text-to-speech (how do we pronounce “lead”?)
  - can write regexps like (Det) Adj* N+ over the output
  - preprocessing to speed up parser (but a little dangerous)
  - if you know the tag, you can back off to it in other tasks
Why Do We Care?

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj
Why Do We Care?

Input: the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
Why Do We Care?

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
- Been done to death by different methods
Why Do We Care?

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
Why Do We Care?

Input: the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task (in English)
Why Do We Care?

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task (in English)
  - Can be done well with methods that look at local context
Why Do We Care?

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task (in English)
  - Can be done well with methods that look at local context
  - Though should “really” do it by parsing!
Tagged Data Sets

- Brown Corpus
  - Designed to be a representative sample from 1961
    - news, poetry, “belles lettres”, short stories
  - 87 different tags

- Penn Treebank
  - 45 different tags
  - Currently most widely used for English
  - Now a paradigm in lots of other languages
    - Chinese Treebank has over 200 tags
<table>
<thead>
<tr>
<th>PART-OF-SPEECH</th>
<th>TAG</th>
<th>EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjective</td>
<td>JJ</td>
<td>happy, bad</td>
</tr>
<tr>
<td>Adjective, comparative</td>
<td>JJR</td>
<td>happier, worse</td>
</tr>
<tr>
<td>Adjective, cardinal number</td>
<td>CD</td>
<td>3, fifteen</td>
</tr>
<tr>
<td>Adverb</td>
<td>RB</td>
<td>often, particularly</td>
</tr>
<tr>
<td>Conjunction, coordination</td>
<td>CC</td>
<td>and, or</td>
</tr>
<tr>
<td>Conjunction, subordinating</td>
<td>IN</td>
<td>although, when</td>
</tr>
<tr>
<td>Determiner</td>
<td>DT</td>
<td>this, each, other, the, a, some</td>
</tr>
<tr>
<td>Determiner, postdeterminer</td>
<td>JJ</td>
<td>many, same</td>
</tr>
<tr>
<td>Noun</td>
<td>NN</td>
<td>aircraft, data</td>
</tr>
<tr>
<td>Noun, plural</td>
<td>NNS</td>
<td>women, books</td>
</tr>
<tr>
<td>Noun, proper, singular</td>
<td>NNP</td>
<td>London, Michael</td>
</tr>
<tr>
<td>Noun, proper, plural</td>
<td>NNPS</td>
<td>Australians, Methodists</td>
</tr>
<tr>
<td>Pronoun, personal</td>
<td>PRP</td>
<td>you, we, she, it</td>
</tr>
<tr>
<td>Pronoun, question</td>
<td>WP</td>
<td>who, whoever</td>
</tr>
<tr>
<td>Verb, base present form</td>
<td>VBP</td>
<td>take, live</td>
</tr>
</tbody>
</table>
Word Class Classes

- Importantly for predicting POS tags, there are two broad classes
- “Closed class” words
  - Belong to classes that don’t accept new members
  - Determiners: the, a, an, this, ...
  - Prepositions: in, on, of, ...
- “Open class” words
  - Nouns, verbs, adjectives, adverbs, ...
- “Closed” is relative: These words are born and die over longer time scales (e.g., “regarding”)
Ambiguity in Language

Fed raises interest rates 0.5% in effort to control inflation

NY Times headline 17 May 2000
## Part-of-speech Ambiguity

Fed raises interest rates 0.5% in effort to control inflation
Degree of Supervision
Degree of Supervision

- **Supervised**: Training corpus is tagged by humans
Degree of Supervision

- **Supervised**: Training corpus is tagged by humans
- **Unsupervised**: Training corpus isn’t tagged
Degree of Supervision

- **Supervised**: Training corpus is tagged by humans
- **Unsupervised**: Training corpus isn’t tagged
- **Partly supervised**: Training corpus isn’t tagged, but you have a dictionary giving possible tags for each word
Degree of Supervision

- **Supervised**: Training corpus is tagged by humans
- **Unsupervised**: Training corpus isn’t tagged
- **Partly supervised**: Training corpus isn’t tagged, but you have a dictionary giving possible tags for each word
Degree of Supervision

- **Supervised**: Training corpus is tagged by humans
- **Unsupervised**: Training corpus isn’t tagged
- **Partly supervised**: Training corpus isn’t tagged, but you have a dictionary giving possible tags for each word

- We’ll start with the supervised case and move to decreasing levels of supervision.
Current Performance

Input: the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj
Current Performance

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- How many tags are correct?
Current Performance

Input:  the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- How many tags are correct?
  - About 97% currently
**Current Performance**

**Input:** the lead paint is unsafe

**Output:** the/Det lead/N paint/N is/V unsafe/Adj

- How many tags are correct?
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
    - Tag every word with its most frequent tag
    - Tag unknown words as nouns
What Should We Look At?

Bill directed a cortege of autos through the dunes
What Should We Look At?

**Correct tags**

PN Verb Det Noun Prep Noun Prep Det Noun

Bill directed a cortege of autos through the dunes
What Should We Look At?

correct tags
PN  Verb  Det  Noun  Prep  Noun  Prep  Det  Noun  
Bill  directed  a  cortege  of  autos  through  the  dunes  
PN  Adj  Det  Noun  Prep  Noun  Prep  Det  Noun  
Verb  Verb  Noun  Verb  
Adj  Prep  
...?

some possible tags for each word (maybe more)
What Should We Look At?

**correct tags**

PN  Verb  Det  Noun  Prep Noun  Prep  Det  Noun
Bill directed a cortege of autos through the dunes

PN  Adj  Det  Noun  Prep Noun  Prep  Det  Noun

Verb  Verb  Noun  Verb

Adj  Prep  some possible tags for each word (maybe more)

...?
What Should We Look At?

correct tags

PN  Verb  Det  Noun  Prep  Noun  Prep  Det  Noun
Bill  directed  a  cortege  of  autos  through  the  dunes

PN  Adj  Det  Noun  Prep  Noun  Prep  Det  Noun
Verb  Verb  Noun  Verb

Adj  Prep  some possible tags for each word (maybe more)
...?
What Should We Look At?

**correct tags**

Bill directed a cortege of autos through the dunes

**some possible tags for each word (maybe more)**

...?
What Should We Look At?

**correct tags**

Bill directed a cortege of autos through the dunes

Each unknown tag is **constrained** by its word
What Should We Look At?

Correct tags

PN  Verb  Det  Noun  Prep  Noun  Prep  Det  Noun
Bill  directed  a  cortege  of  autos  through  the  dunes

PN  Adj  Det  Noun  Prep  Noun  Prep  Det  Noun
Verb  Verb  Noun  Verb

Adj  Prep  some possible tags for each word (maybe more)
...

Each unknown tag is constrained by its word and by the tags to its immediate left and right.
What Should We Look At?

correct tags

Bill directed a cortege of autos through the dunes

Each unknown tag is constrained by its word and by the tags to its immediate left and right. But those tags are unknown too ...
What Should We Look At?

correct tags

PN   Verb   Det   Noun  Prep  Noun   Prep  Det  Noun
Bill directed a cortege of autos through the dunes

PN   Adj    Det   Noun  Prep  Noun   Prep  Det  Noun
Verb  Verb  Noun  Verb

Adj    Prep    ...?

some possible tags for each word (maybe more)

Each unknown tag is constrained by its word and by the tags to its immediate left and right. But those tags are unknown too ...
What Should We Look At?

correct tags

PN  Verb  Det  Noun  Prep  Noun  Prep  Det  Noun

Bill directed a cortege of autos through the dunes

PN  Adj  Det  Noun  Prep  Noun  Prep  Det  Noun
Verb  Verb  Noun  Verb

Adj  Prep  some possible tags for each word (maybe more) ...

Each unknown tag is constrained by its word and by the tags to its immediate left and right. But those tags are unknown too ...
Finite-State Approaches

- Noisy Channel Model (statistical)

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$

part-of-speech tags (n-gram model)

replace tags with words

text
Review: Noisy Channel

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

\[ p(X) \star p(Y \mid X) = p(X,Y) \]
Review: Noisy Channel

real language \( X \)

noisy channel \( X \rightarrow Y \)

yucky language \( Y \)

\[ p(X) \]

\[ p(Y | X) \]

\[ = \]

\[ p(X,Y) \]

want to recover \( x \in X \) from \( y \in Y \)
Review: Noisy Channel

want to recover \( x \in X \) from \( y \in Y \)
choose \( x \) that maximizes \( p(x \mid y) \) or equivalently \( p(x,y) \)

\[
p(X) \quad * \quad p(Y \mid X) \quad = \quad p(X,Y)
\]
Noisy Channel for Tagging

**acceptor:** \( p(\text{tag sequence}) \)

"Markov Model"

\( p(X) \)

\( \ast \)

**transducer:** tags \( \rightarrow \) words

"Unigram Replacement"

\( p(Y \mid X) \)

\( \ast \)

**acceptor:** the observed words

"straight line"

\( (Y = y) \)?

\( = \)

**transducer:** scores candidate tag seqs on their joint probability with obs words;

pick best path

\( p(X, y) \)
Markov Model (bigrams)
Markov Model (bigrams)

Diagram:
- Start
- Det
- Adj
- Noun
- Verb
- Prep
- Stop
Markov Model (bigrams)

`Start` → `Det` → `Adj` → `Noun` → `Verb` → `Prep` → `Stop`
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model
Markov Model

- **Start**
  - Det
    - Adj
    - Noun
  - 0.3
  - 0.7
- Verb
- Prep
- Stop
Markov Model

Diagram showing the transitions between different parts of speech:

- **Start**
  - Det with probability 0.3
  - Adj with probability 0.4

- Adj
  - Det with probability 0.7
  - Noun with probability 0.5

- Noun
  - Verb
  - Prep

- Stop
  - Back to Start with probability 0.1
Markov Model

Diagram showing transitions between parts of speech:
- Start to Det with probability 0.8
- Det to Adj with probability 0.3
- Adj to Noun with probability 0.1
- Det to Verb with probability 0.7
- Verb to Prep with probability 0.4
- Prep to Noun with probability 0.5
- Noun to Stop with probability 0.2
- Stop to Start with probability 0.1
Markov Model

\[ p(\text{tag seq}) \]

\[
\text{Start} \xrightarrow{0.8} \text{Det} \xrightarrow{0.3} \text{Adj} \xrightarrow{0.4} \text{Adj} \xrightarrow{0.5} \text{Noun} \xrightarrow{0.1} \text{Stop} = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2
\]
Markov Model as an FSA

\[ p(\text{tag seq}) \]

\[ \text{Start} \xrightarrow{0.8} \text{Det} \xrightarrow{0.3} \text{Adj} \xrightarrow{0.4} \text{Adj} \xrightarrow{0.5} \text{Noun} \xrightarrow{0.7} \text{Verb} \xrightarrow{0.2} \text{Stop} \]

\[ \text{Start Det Adj Adj Noun Stop} = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2 \]
Markov Model as an FSA

\[ p(\text{tag seq}) = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2 \]
Markov Model as an FSA

$p(\text{tag seq})$

\[ \text{Start} \xrightarrow{\text{Det } 0.8} \text{Det} \xrightarrow{\text{Adj } 0.3} \text{Adj} \xrightarrow{\text{Adj } 0.4} \text{Adj} \xrightarrow{\text{Noun } 0.5} \text{Noun} \xrightarrow{\varepsilon 0.1} \text{Start} \]

\[ \text{Det} \xrightarrow{\text{Adj } 0.4} \text{Adj} \xrightarrow{\text{Noun } 0.7} \text{Verb} \]

\[ \text{Prep} \xrightarrow{\varepsilon 0.2} \text{Stop} \]

\[ \text{Start Det Adj Adj Noun Stop} = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2 \]
**Markov Model (tag bigrams)**

\[ p(\text{tag seq}) \]

\[ \text{Start} \quad \text{Det} 0.8 \quad \text{Adj} 0.3 \quad \text{Adj} 0.4 \quad \text{Det} \quad \text{Adj} 0.4 \quad \text{Noun} 0.5 \quad \text{Stop} \quad \varepsilon 0.2 \]

\[ \text{Start Det Adj Adj Noun Stop} = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2 \]
Noisy Channel for Tagging

**automaton:** $p(\text{tag sequence})$

“Markov Model”

$\mathbf{.o.}$

**transducer:** tags $\rightarrow$ words

“Unigram Replacement”

$\mathbf{.o.}$

$\mathbf{p}(\mathbf{X})$

**automaton:** the observed words

“straight line”

$\mathbf{p}(\mathbf{y} | \mathbf{Y})$

$\mathbf{p}(\mathbf{X}, \mathbf{y})$

$\mathbf{p}(\mathbf{Y} | \mathbf{X})$

$\mathbf{p}(\text{tag sequence})$

“Markov Model”

$\mathbf{.o.}$
transducer: scores candidate tag seqs
on their joint probability with obs words;
we should pick best path
Unigram Replacement Model

\[ p(\text{word seq} \mid \text{tag seq}) \]

\[ \text{sums to 1} \]

\[ \text{sums to 1} \]
Compose

$p(\text{tag seq})$
Compose

\[ p(\text{word seq}, \text{tag seq}) = p(\text{tag seq}) \times p(\text{word seq} | \text{tag seq}) \]
Observed Words as Straight-Line FSA

word seq

\[ \text{the} \rightarrow \text{cool} \rightarrow \text{directed} \rightarrow \text{autos} \]
Compose with

\[ p(\text{word seq}, \text{tag seq}) = p(\text{tag seq}) \ast p(\text{word seq} | \text{tag seq}) \]
Compose with

\[ p(\text{word seq, tag seq}) = p(\text{tag seq}) \times p(\text{word seq | tag seq}) \]
p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)

Compose with

why did this loop go away?

Adj: directed 0.00020

Adj: cool 0.0009

N: autos
The best path:

Start  Det  Adj  Adj  Noun  Stop  = 0.32 * 0.0009 ...
the  cool  directed  autos

\[
p(\text{word seq, tag seq}) = p(\text{tag seq}) \times p(\text{word seq | tag seq})
\]
In Fact, Paths Form a “Trellis”

\[ p(\text{word seq, tag seq}) \]

The best path:

\[ \text{Start} \quad \text{Det} \quad \text{Adj} \quad \text{Adj} \quad \text{Noun} \quad \text{Stop} = 0.32 \times 0.0009 \ldots \]

the cool directed autos
In Fact, Paths Form a “Trellis”

\[ p(\text{word seq, tag seq}) \]

The best path:

Start Det Adj Adj Noun Stop = 0.32 * 0.0009 ...

the cool directed autos
In Fact, Paths Form a “Trellis”

\[ p(\text{word seq, tag seq}) \]

The best path:

**Start** Det Adj Adj Noun **Stop** = 0.32 * 0.0009 ...

the cool directed autos
The Trellis Shape Emerges from the Cross-Product Construction for

\[ 0, 0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \]
The Trellis Shape Emerges from the Cross-Product Construction for

\[
\begin{align*}
0 & \rightarrow 1 \\
1 & \rightarrow 2 \\
2 & \rightarrow 3 \\
3 & \rightarrow 4
\end{align*}
\]

= 

\[
\begin{align*}
0,0 & \rightarrow 1,1 \\
1,1 & \rightarrow 1,2 \\
2,1 & \rightarrow 1,3 \\
3,1 & \rightarrow 1,4 \\
2,2 & \rightarrow 2,3 \\
3,2 & \rightarrow 2,4 \\
3,3 & \rightarrow 3,4 \\
4,4
\end{align*}
\]
The Trellis Shape Emerges from the Cross-Product Construction for

All paths here are 4 words

= All paths here are 4 words
The Trellis Shape Emerges from the Cross-Product Construction for

All paths here are 4 words

So all paths here must have 4 words on output side
Actually, Trellis Isn’t Complete

\[ p(\text{word seq, tag seq}) \]

The best path:

**Start** Det Adj Adj Noun **Stop** = 0.32 * 0.0009 ... 
the cool directed autos
Actually, Trellis Isn’t Complete

\[ p(\text{word seq}, \text{tag seq}) \]

Trellis has no Det \( \rightarrow \) Det or Det \( \rightarrow \) Stop arcs; why?

The best path:

\textbf{Start} Det Adj Adj Noun \textbf{Stop} = 0.32 \times 0.0009 \ldots

the cool directed autos
Actually, Trellis Isn’t Complete

\[
p(\text{word seq, tag seq})
\]

Lattice is missing some other arcs; why?

The best path:

\[
\text{Start Det Adj Adj Noun Stop} = 0.32 \times 0.0009 \ldots
\]

the cool directed autos
Actually, Trellis Isn’t Complete

\[ p(\text{word seq, tag seq}) \]

Lattice is missing some states; why?

The best path:

Start Det Adj Adj Noun Stop \( = 0.32 * 0.0009 \ldots \)
the cool directed autos
Find best path from Start to Stop
Find best path from Start to Stop

- Use dynamic programming:
  - What is best path from Start to each node?
  - Work from left to right
  - Each node stores its best path from Start (as probability plus one backpointer)
Use dynamic programming:
- What is best path from Start to each node?
- Work from left to right
- Each node stores its best path from Start (as probability plus one backpointer)

Special acyclic case of Dijkstra’s shortest-path alg.
Find best path from Start to Stop

- Use dynamic programming:
  - What is best path from Start to each node?
  - Work from left to right
  - Each node stores its best path from Start (as probability plus one backpointer)
- Special acyclic case of Dijkstra’s shortest-path alg.
- Faster if some arcs/states are absent
In Summary

- We are modeling $p(\text{word seq}, \text{tag seq})$
In Summary

- We are modeling $p(\text{word seq, tag seq})$
- The tags are hidden, but we see the words
In Summary

- We are modeling $p(\text{word seq, tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
In Summary

- We are modeling $p(\text{word seq, tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence $X$ likely with these words?
- Noisy channel model is a “Hidden Markov Model”:
In Summary

- We are modeling $p(\text{word seq, tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a “Hidden Markov Model”:

```
<table>
<thead>
<tr>
<th>Start</th>
<th>PN</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
<th>Prep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>Stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bill directed a cortege of autos through
In Summary

- We are modeling $p(\text{word seq}, \text{tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a “Hidden Markov Model”:

```
Start  PN   Verb    Det    Noun    Prep    Noun    Prep    Noun    Stop
Bill  directed  a  cortege  of  autos  through
```

0.4  0.6  0.001
In Summary

- We are modeling $p(\text{word seq, tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a “Hidden Markov Model”:

```
Start  PN  Verb  Det  Noun  Prep  Noun  Prep  Noun
Noun  Stop
Bill  directed  a  cortege  of  autos  through
```
In Summary

- We are modeling $p(\text{word seq}, \text{tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence $X$ likely with these words?
- Noisy channel model is a “Hidden Markov Model”:

```
<table>
<thead>
<tr>
<th>Start</th>
<th>PN</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Stop</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.001</td>
</tr>
<tr>
<td>Bill</td>
<td>directed</td>
<td>a</td>
<td>cortege of</td>
<td>autos</td>
<td>through</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
In Summary

- We are modeling $p(\text{word seq, tag seq})$
- The tags are hidden, but we see the words
- Is tag sequence $X$ likely with these words?
- Noisy channel model is a “Hidden Markov Model”:

Find $X$ that maximizes probability product
Another Viewpoint
Another Viewpoint

- We are modeling $p(\text{word seq}, \text{tag seq})$
Another Viewpoint

- We are modeling $p(\text{word seq, tag seq})$
- Why not use chain rule + some kind of backoff?
Another Viewpoint

- We are modeling $p(\text{word seq}, \text{tag seq})$
- Why not use chain rule + some kind of backoff?
- Actually, we are!
Another Viewpoint

- We are modeling $p(\text{word seq, tag seq})$
- Why not use chain rule + some kind of backoff?
- Actually, we are!

$$p(\text{Start PN Verb Det ...} \mid \text{Bill directed a ...}) = p(\text{Start}) \cdot p(\text{PN} \mid \text{Start}) \cdot p(\text{Verb} \mid \text{Start PN}) \cdot p(\text{Det} \mid \text{Start PN Verb}) \cdot \ldots \cdot p(\text{Bill} \mid \text{Start PN Verb ...}) \cdot p(\text{directed} \mid \text{Bill, Start PN Verb Det ...}) \cdot \ldots \cdot p(\text{a} \mid \text{Bill directed, Start PN Verb Det ...}) \cdot \ldots$$
Another Viewpoint

- We are modeling \( p(\text{word seq, tag seq}) \)
- Why not use chain rule + some kind of backoff?
- Actually, we are!

\[
p(\text{Start PN Verb Det ... Bill directed a ...}) = \]
\[
p(\text{Start}) \times p(\text{PN | Start}) \times p(\text{Verb | Start PN}) \times p(\text{Det | Start PN Verb}) \times ...
\]
\[
\times p(\text{Bill | Start PN Verb ...}) \times p(\text{directed | Bill, Start PN Verb Det ...})
\]
\[
\times p(\text{a | Bill directed, Start PN Verb Det ...}) \times ...
\]
Another Viewpoint

- We are modeling $p(\text{word seq}, \text{tag seq})$
- Why not use chain rule + some kind of backoff?
- Actually, we are!

$$p(\text{Start PN Verb Det ...}) = p(\text{Start}) \times p(\text{PN} \mid \text{Start}) \times p(\text{Verb} \mid \text{Start PN}) \times p(\text{Det} \mid \text{Start PN Verb}) \times ... \times p(\text{Bill} \mid \text{Start PN Verb ...}) \times p(\text{directed} \mid \text{Bill, Start PN Verb Det ...}) \times p(\text{a} \mid \text{Bill directed, Start PN Verb Det ...}) \times ...$$
Another Viewpoint

- We are modeling $p(\text{word seq, tag seq})$
- Why not use chain rule + some kind of backoff?
- Actually, we are!

\[
p(\text{Start PN Verb Det ... Bill directed a ...}) = p(\text{Start}) \times p(\text{PN | Start}) \times p(\text{Verb | Start PN}) \times p(\text{Det | Start PN Verb}) \times ... \times p(\text{Bill | Start PN Verb ...}) \times p(\text{directed | Bill, Start PN Verb Det ...}) \times p(\text{a | Bill directed, Start PN Verb Det ...}) \times ...
\]

Start PN Verb Det Noun Prep Noun Prep Det Noun Stop
Bill directed a cortege of autos through the dunes
Variations
Variations

- Multiple tags per word
Variations

- Multiple tags per word
  - Transformations to knock some of them out
Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?
Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?
Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?
- Use the above for partly supervised learning
Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

- Use the above for partly supervised learning
  - Supervised: You have a tagged training corpus
Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

Use the above for partly supervised learning
  - Supervised: You have a tagged training corpus
  - Unsupervised: You have an untagged training corpus
Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

- Use the above for *partly supervised* learning
  - **Supervised:** You have a tagged training corpus
  - **Unsupervised:** You have an untagged training corpus
  - **Here:** You have an untagged training corpus and a dictionary giving possible tags for each word
Applications of HMMs

- NLP
  - Part-of-speech tagging
  - Word segmentation
  - Information extraction
  - Optical character recognition
- Speech recognition
  - Modeling acoustics, with continuous emissions
- Computer Vision
  - Gesture recognition
- Biology
  - Gene finding
  - Protein structure prediction
- Economics, Climatology, Robotics, etc.
A More Traditional View of HMMs
Recipe for NLP

Input:  the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

1) **Data**: Notation, representation
2) **Problem**: Write down the problem in notation
3) **Model**: Make some assumptions, define a parametric model (often generative model of the data)
4) **Inference**: How to search through possible answers to find the best one
5) **Learning**: How to estimate parameters
6) **Implementation**: Engineering considerations for an efficient implementation
An HMM Tagger

- View sequence of tags as a Markov chain. Assumptions:
  - Limited horizon \( P(x_{t+1}|x_1, \ldots x_t) = P(x_{t+1}|x_t) \)
  - Time invariant (stationary) \( P(x_{t+1}|x_t) = P(x_2|x_1) \)
  - We assume that a word’s tag only depends on the previous tag (limited horizon) and that his dependency does not change over time (time invariance)
  - A state (part of speech) generates a word. We assume it depends only on the state.
    \[
    P(o_t|x_1, \ldots x_T, o_1, \ldots o_{t-1}) = P(o_t|x_t)
    \]
The Markov Property

- A stochastic process has the **Markov property** if the conditional probability distribution of future states of the process, given the current state, depends only upon the current state, and conditionally independent of the past states (the *path* of the process) given the current state.

- A process with the Markov property is usually called a **Markov process**, and may be described as *Markovian*.

\[
\Pr[X(t+h) = y \mid X(s) = x(s), s \leq t] = \Pr[X(t+h) = y \mid X(t) = x(t)], \quad \forall h > 0.
\]
HMM w/State Emissions

transitions

$P(x_{t+1}|x_t)$

emissions

for above in

$P(o_t|x_t)$
HMM as Bayes Net

- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations (words)
(One) Standard HMM Formalism

- \( (X, O, x_s, A, B) \) are all variables. Model \( \mu = (A, B) \)
- \( X \) is state sequence of length \( T \); \( O \) is observation seq.
- \( x_s \) is a designated start state (with no incoming transitions). (Can also be separated into \( \pi \) as in book.)
- \( A \) is matrix of transition probabilities (each row is a conditional probability table (CPT))
- \( B \) is matrix of output probabilities (vertical CPTs)
- HMM is a probabilistic (nondeterministic) finite state automaton, with probabilistic outputs (from vertices, not arcs, in the simple case)

\[
P(X, O|\mu) = \prod_{t=1}^{T} a[x_t|x_{t-1}] \cdot b[o_t|x_t]
\]
HMM Inference Problems

- Given an observation sequence, find the most likely state sequence (tagging)
- Compute the probability of observations when state sequence is hidden (language modeling)
- Given observations and (optionally) a their corresponding states, find parameters that maximize the probability of the observations (parameter estimation)
Most Likely State Sequence

- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B)$
- We want to find

$$\arg \max_X P(X|O, \mu) = \arg \max_X \frac{P(X, O|\mu)}{P(O|\mu)} = \arg \max_X P(X, O|\mu)$$

- $P(O,X|\mu) = P(O|X, \mu) P(X|\mu)$
- $P(O|X, \mu) = b[x_1|o_1] b[x_2|o_2] \ldots b[x_T|o_T]$  
- $P(X|\mu) = a[x_1|x_2] a[x_2|x_3] \ldots a[x_{T-1}|x_T]$  
- $\arg \max_X P(O,X|\mu) = \arg \max x_1, x_2, \ldots x_T$
- Problem: $\arg \max$ is exponential in sequence length!
Paths in a Trellis

States

X1

x2

x3

x4

Time 1 2 3 4 ... T

Paths in a Trellis
Paths in a Trellis

States

X1

x2

x3

x4

Time 1 2 3 4 ... T
Paths in a Trellis

\[ \delta_i(t) = \text{Probability of most likely path that ends at state } i \text{ at time } t. \]
Dynamic Programming

• Efficient computation of max over all states
• Intuition: Probability of the first $t$ observations is the same for all possible $t+1$ length sequences.
• Define forward score:
  $$\delta_i(t) = \max_{x_1\ldots x_{t-1}} P(o_1 o_2 \ldots o_t, x_1 \ldots x_{t-1}, x_t = i | \mu)$$
  $$\delta_j(t+1) = \max_{i=1\ldots N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j]$$
• Compute it recursively from the beginning
• (Then must remember best paths to get arg max.)
The Viterbi Algorithm (1967)

- Used to efficiently find the state sequence that gives the highest probability to the observed outputs
- Maintains two dynamic programming tables:
  - The probability of the best path (max)
    \[
    \delta_j(t + 1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j]
    \]
  - The state transitions of the best path (arg)
    \[
    \psi_j(t + 1) = \arg \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j]
    \]
- Note that this is different from finding the most likely tag for each time \( t \)!
Viterbi Recipe

- Initialization
  \[ \delta_j(0) = 1 \text{ if } x_j = x_s. \quad \delta_j(0) = 0 \text{ otherwise.} \]

- Induction
  \[ \delta_j(t + 1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j] \]
  Store backtrace
  \[ \psi_j(t + 1) = \arg \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j] \]

- Termination and path readout
  \[ \hat{x}_T = \arg \max_{i=1..N} \delta_i(T) \]
  Probability of entire best seq.
  \[ P(\hat{X}) = \max_{i=1..N} \delta_i(T) \]

  \[ \hat{x}_t = \psi_{\hat{x}_{t+1}}(t + 1) \]
HMMs:
Maxing and Summing
Markov vs. Hidden Markov Models

Fed raises interest rates
Markov vs. Hidden Markov Models
Markov vs. Hidden Markov Models

Fed raises interest rates...
Markov vs. Hidden Markov Models

Fed raises interest rates... raises rates...

interest...

raises rates...

NN NNS VBZ VB NNP
Markov vs. Hidden Markov Models

Fed raises interest rates

interest raises rates

raises...
Markov vs. Hidden Markov Models

Fed raises interest rates...
Markov vs. Hidden Markov Models
Unrolled into a Trellis

<table>
<thead>
<tr>
<th>NN</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NNS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBZ</td>
<td>Fed</td>
<td>raises</td>
<td>interest</td>
<td>rates</td>
</tr>
</tbody>
</table>
HMM Inference Problems

• Given an observation sequence, find the most likely state sequence (tagging)

• Compute the probability of observations when state sequence is hidden (language modeling)

• Given observations and (optionally) a their corresponding states, find parameters that maximize the probability of the observations (parameter estimation)
Tagging

Given an observation sequence, find the most likely state sequence.

$$\arg \max_x P(X \mid O, \mu) = \arg \max_x \frac{P(X, O \mid \mu)}{P(O \mid \mu)} = \arg \max_x P(X, O \mid \mu)$$

$$\arg \max_{x_1, x_2, \ldots, x_T} P(x_1, x_2, \ldots, x_T, O \mid \mu)$$

Last time: Use dynamic programming to find highest-probability sequence (i.e. best path, like Dijkstra’s algorithm)
Language Modeling

Compute the probability of observations when state sequence is hidden.

\[
P(X, O \mid \mu) = P(O \mid X, \mu)P(X \mid \mu)
\]

Therefore

\[
P(O \mid \mu) = \sum_X P(O \mid X, \mu)P(X \mid \mu)
\]

\[
\sum_{x_1, x_2, \ldots, x_T} P(x_1, x_2, \ldots, x_T, O \mid \mu)
\]

Suspiciously similar to

\[
\max_{x_1, x_2, \ldots, x_T} P(x_1, x_2, \ldots, x_T, O \mid \mu)
\]
Viterbi Algorithm (Tagging)

Fed raises interest rates

NN  NNS  NNP  VB  VBZ

Fed   raises  interest  rates
Viterbi Algorithm (Tagging)

Fed raises interest rates
Viterbi Algorithm (Tagging)

NN  
NNS 
NNP 
VB  
VBZ 

Fed raises interest rates
Viterbi Algorithm (Tagging)

Fed raises interest rates
Viterbi Algorithm (Tagging)

Fed raises interest rates
Viterbi Algorithm (Tagging)

Fed raises interest rates
Viterbi Algorithm (Tagging)

NN

NNS

NNP

VB

VBZ

Fed raises interest rates
Viterbi Algorithm (Tagging)

NN

δ_{NN}(2) \ a[VB|NN]\ b[interest|VB]

NNS

δ_{NNS}(2) \ a[VB|NNS]\ b[interest|VB]

NNP

δ_{NNP}(2) \ a[VB|NNP]\ b[interest|VB]

VB

δ_{VB}(2) \ a[VB|VB]\ b[interest|VB]

VBZ

δ_{VBZ}(2) \ a[VB|VBZ]\ b[interest|VB]

Fed raises interest rates
Forward Algorithm (LM)

NN  NNS  NNP  VB  VBZ  Fed  raises  interest  rates
Forward Algorithm (LM)

Fed raises interest rates
Forward Algorithm (LM)

NN  NNP  NNS  VB  VBZ

Fed raises interest rates
Forward Algorithm (LM)

Fed raises interest rates
Forward Algorithm (LM)

Fed raises interest rates

NN NNS NNP VB VBZ

a[VB]|NNS|b[interest|VB]

a[VB]|NNP|b[interest|VB]

a[VB]|VB|b[interest|VB]

a[VB]|VBZ|b[interest|VB]
Forward Algorithm (LM)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td><img src="image" alt="NN" /></td>
</tr>
<tr>
<td>NNS</td>
<td><img src="image" alt="NNS" /></td>
</tr>
<tr>
<td>NNP</td>
<td><img src="image" alt="NNP" /></td>
</tr>
<tr>
<td>VB</td>
<td><img src="image" alt="VB" /></td>
</tr>
<tr>
<td>VBZ</td>
<td><img src="image" alt="VBZ" /></td>
</tr>
</tbody>
</table>

Fed raises interest rates
Forward Algorithm (LM)

NN  α_{NN}(2) a[VB|NN] b[interest|VB]  
NNS α_{NNS}(2) a[VB|NNS] b[interest|VB]  
NNP α_{NNP}(2) a[VB|NNP] b[interest|VB]  
VB  α_{VB}(2) a[VB|VB] b[interest|VB]  
VBZ α_{VBZ}(2) a[VB|VBZ] b[interest|VB]  
Fed raises interest rates
### Forward Algorithm (LM)

<table>
<thead>
<tr>
<th>Part of Speech</th>
<th>Text</th>
<th>Forward Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>NNP</td>
<td>$\alpha_{NN}(2)$</td>
</tr>
<tr>
<td>NNS</td>
<td>NNS</td>
<td>$\alpha_{NN}(2)$</td>
</tr>
<tr>
<td>NNP</td>
<td>NNP</td>
<td>$\alpha_{NN}(2)$</td>
</tr>
<tr>
<td>VB</td>
<td>VB</td>
<td>$\alpha_{VB}(2)$</td>
</tr>
<tr>
<td>VBZ</td>
<td>VBZ</td>
<td>$\alpha_{VB}(2)$</td>
</tr>
</tbody>
</table>

**Sentence: Fed raises interest rates**

**Equation:**

$$\text{sum} = \alpha_{VB}(3)$$
What Do These Greek Letters Mean?

\[
\delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu)
\]

\[
\alpha_j(t) = \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu)
\]

\[
= P(o_1 \cdots o_{t-1}, x_t = j \mid \mu)
\]
What Do These Greek Letters Mean?

\[ \delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j | \mu) \]

Probability of the best path from the beginning to word \( t \) such that word \( t \) has tag \( j \)

\[ \alpha_j(t) = \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j | \mu) \]

\[ = P(o_1 \cdots o_{t-1}, x_t = j | \mu) \]
What Do These Greek Letters Mean?

\[ \delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \]

Probability of the best path from the beginning to word \( t \) such that word \( t \) has tag \( j \)

\[ \alpha_j(t) = \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \]

Probability of all paths from the beginning to word \( t \) such that word \( t \) has tag \( j \)

\[ = P(o_1 \cdots o_{t-1}, x_t = j \mid \mu) \]
What Do These Greek Letters Mean?

\[ \delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \]

Probability of the best path from the beginning to word \( t \) such that word \( t \) has tag \( j \)

\[ \alpha_j(t) = \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \]

Probability of all paths from the beginning to word \( t \) such that word \( t \) has tag \( j \)

\[ = P(o_1 \cdots o_{t-1}, x_t = j \mid \mu) \]

NOT the probability of tag \( j \) at time \( t \)
HMM Language Modeling

• Probability of observations, summed over all possible ways of tagging that observation:

\[ \sum_{i} \alpha_i(T) \]

• This is the sum of all path probabilities in the trellis
HMM Parameter Estimation

- Supervised
  - Train on tagged text, test on plain text
  - Maximum likelihood (can be smoothed):
    - $a[VBZ \mid NN] = \frac{C(NN,VBZ)}{C(NN)}$
    - $b[rates \mid VBZ] = \frac{C(VBZ,rates)}{C(VBZ)}$
- Unsupervised
  - Train and test on plain text
  - What can we do?
Forward-Backward Algorithm

NN
NNS
NNP
VB
VBZ

Fed raises interest rates

\( \alpha_{NN}(2) \)
\( \alpha_{NNS}(2) \)
\( \alpha_{NNP}(2) \)
\( \alpha_{VB}(2) \)
\( \alpha_{VBZ}(2) \)
Forward-Backward Algorithm

Fed raises interest rates

NN
NNS
NNP
VB
VBZ

\( \alpha_{NN}(2) \)
\( \alpha_{NNS}(2) \)
\( \alpha_{NNP}(2) \)
\( \alpha_{VB}(2) \)
\( \alpha_{VBZ}(2) \)
Forward-Backward Algorithm

Fed raises interest rates

α_{NN}(2) \quad \alpha_{NNS}(2) \quad \alpha_{NNP}(2) \quad \alpha_{VB}(2) \quad \alpha_{VBZ}(2)

α_{NN}(2) \quad \alpha_{NNS}(2) \quad \alpha_{NNP}(2) \quad \alpha_{VB}(2) \quad \alpha_{VBZ}(2)
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

NN
NNS
NNP
VB
VBZ

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates

αNN(2) a[VB|NN]b[interest|VB] a[NN|VB]b[rates|NN] βNN(4)

αNNS(2) a[VB|NNS]b[interest|VB] a[NNS|VB]b[rates|NNS] βNNS(4)

αNNP(2) a[VB|NNP]b[interest|VB] a[NNP|VB]b[rates|NNP] βNNP(4)

αVB(2) a[VB|VB]b[interest|VB] a[VB|VB]b[rates|VB] βVB(4)

αVBZ(2) a[VB|VBZ]b[interest|VB] a[VBZ|VB]b[rates|VBZ] βVBZ(4)
Forward-Backward Algorithm

\[ P(o_1 \cdots o_{t-1}, x_t = j \mid \mu) = \alpha_j(t) \]

\[ P(o_t \cdots o_T \mid x_t = j, \mu) = \beta_j(t) \]

\[ P(o_1 \cdots o_T, x_t = j \mid \mu) = \alpha_j(t)\beta_j(t) \]

\[ P(x_t = j \mid O, \mu) = \frac{P(x_t = j, O \mid \mu)}{P(O \mid \mu)} = \frac{\alpha_j(t)\beta_j(t)}{\alpha_\#(T)} \]

\[ P(x_t = i, x_{t+1} = j \mid O, \mu) = \frac{P(x_t = i, x_{t+1} = j, O \mid \mu)}{P(O \mid \mu)} = \frac{\alpha_i(t)a[j \mid i]b[o_t \mid j]\beta_j(t + 1)}{\alpha_\#(T)} \]
Expectation Maximization (EM)

• Iterative algorithm to maximize likelihood of observed data in the presence of hidden data (e.g., tags)

• Choose an initial model $\mu$

• **Expectation step**: find the expected value of hidden variables given current $\mu$

• **Maximization step**: choose new $\mu$ to maximize probability of hidden and observed data

• Guaranteed to increase likelihood

• Not guaranteed to find global maximum
## Supervised vs. Unsupervised

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotated training text</td>
<td>Plain text</td>
</tr>
<tr>
<td>Simple count/normalize</td>
<td>EM</td>
</tr>
<tr>
<td>Fixed tag set</td>
<td>Set during training</td>
</tr>
<tr>
<td>Training reads data once</td>
<td>Training needs multiple passes</td>
</tr>
</tbody>
</table>
Logarithms for Precision

\[ P(Y) = p(y_1)p(y_2)\cdots p(y_T) \]

\[ \log P(Y) = \log p(y_1) + \log p(y_2)\cdots + \log p(y_T) \]

Increased dynamic range of \([0,1]\) to \([\infty,0]\)
# Semirings

<table>
<thead>
<tr>
<th></th>
<th>Set</th>
<th>⊕</th>
<th>⊗</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prob</td>
<td>$\mathbb{R}^+$</td>
<td>$+$</td>
<td>$\times$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Max</td>
<td>$\mathbb{R}^+$</td>
<td>max</td>
<td>$\times$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Log</td>
<td>$\mathbb{R} \cup {\pm \infty}$</td>
<td>log+</td>
<td>$+$</td>
<td>$-\infty$</td>
<td>0</td>
</tr>
<tr>
<td>“Tropical”</td>
<td>$\mathbb{R} \cup {\pm \infty}$</td>
<td>max</td>
<td>$+$</td>
<td>$-\infty$</td>
<td>0</td>
</tr>
<tr>
<td>Shortest path</td>
<td>$\mathbb{R} \cup {\pm \infty}$</td>
<td>min</td>
<td>$+$</td>
<td>$\infty$</td>
<td>0</td>
</tr>
<tr>
<td>Boolean</td>
<td>${0, 1}$</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>String</td>
<td>$\Sigma^* \cup {\infty}$</td>
<td>longest common prefix</td>
<td>concat</td>
<td>$\infty$</td>
<td>$\varepsilon$</td>
</tr>
</tbody>
</table>