Noisy Channel and Hidden Markov Models

Natural Language Processing
CS 4120/6120—Spring 2017
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

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
    - ance $\rightarrow$ ence (deliverance, ...)
    - e $\rightarrow$ $\varepsilon$ (deliverance, ...)
    - $\varepsilon$ $\rightarrow$ e $\rightarrow$ Cons _ Cons (athlete, ...)
    - rr $\rightarrow$ r (embarrass occurrence, ...)
    - ge $\rightarrow$ 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 $Y$
Noisy Channel Model

real language \( X \)

noisy channel \( X \rightarrow Y \)

yucky language \( Y \)

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

(delete spaces)

(text w/o spaces)

(lexicon space)*
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$

(lexicon space)*

pronunciation

speech
*Noisy Channel Model*

real language \(X\)  
\[
\begin{align*}
\Downarrow \text{language model} \\
\text{noisy channel} \quad X \rightarrow Y
\end{align*}
\]  
\[
\Downarrow \text{acoustic model} \\
\text{yucky language} \quad Y
\]

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 $Y$
Noisy Channel Model

real language $X$

language model $\downarrow$

noisy channel $X \rightarrow Y$

translation model $\downarrow$

yucky language $Y$

“target” language

translation

“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 $X \rightarrow Y$

deprecated everything but terminals

text

tree

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

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 \( y \in Y \)
Noisy Channel Model

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

$p(X)$
*p
$p(Y \mid X)$

$= p(X, Y)$

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

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

\[ p(X) \]

\[ p(Y | X) \]

\[ p(X,Y) \]

\[ a:a/0.7 \]

\[ b:b/0.3 \]

\[ a:C/0.1 \]

\[ b:C/0.8 \]

\[ a:D/0.9 \]

\[ b:D/0.2 \]

\[ p(X) \]

\[ * \]

\[ p(Y | X) \]

\[ = \]

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

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

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

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

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

\begin{align*}
\text{p}(X) * \\
\text{p}(Y | X) = \\
\text{p}(X,Y)
\end{align*}

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

Suppose \( y=\text{“C”} \); what is best \( x \)?
Noisy Channel Model

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

\[ p(Y \mid X) = a:a/0.7, a:C/0.07, b:b/0.3, b:D/0.06 \]

\[ p(X, Y) = a:D/0.63, b:D/0.06 \]

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

\[ p(X, y) = a:a/0.7 \quad b:b/0.3 \]
\[ a:C/0.1 \quad b:C/0.8 \quad a:D/0.9 \quad b:D/0.2 \]

\[ p(X) \quad \ast \quad p(Y | X) \]

\[ = \]

\[ p(X, y) \]

\[ a:C/0.07 \quad b:C/0.24 \]
Noisy Channel Model

\[ p(X) \times p(Y \mid X) = p(X, y) \]

restrict just to paths compatible with output "C"
Noisy Channel Model

\[ p(X) \]

\[ p(Y | X) \]

\[ p(X, y) \]

\[ \ast \]

\[ a:D/0.9 \]

\[ a:C/0.1 \]

\[ b:C/0.8 \]

\[ b:D/0.2 \]

\[ a:a/0.7 \]

\[ b:b/0.3 \]

\[ a:a/0.07 \]

\[ b:b/0.24 \]

\[ \ast \]

restrict just to paths compatible with output “C”

\[ (Y=y)? \]

\[ = \]

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

\[ p(X) \]

\[ p(Y | X) \]

\[ (Y=y)? \]

\[ p(X, y) \]

restrict just to paths compatible with output “C”

best path
Morpheme Segmentation

- 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: *uygarlaştıramadıklarımızdanmışsınızcasına*
    = *uygar+la+şt+ı+ra+ma+dı+k+la+rı+mı+z+da+n+miş+sınız+ca+si+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
- $O(k^2)$ substitution arcs
- $O(k)$ insertion arcs
- $O(k)$ no-change arcs
Stochastic Edit Distance Transducer

Likely edits = high-probability arcs
Stochastic Edit Distance Transducer

clara

\[ \text{.0.} \]

a: \( \varepsilon \)
b: \( \varepsilon \)
a:b

\( \text{.0.} \)
caca

a: \( \varepsilon \)
b: \( \varepsilon \)
a:a

b: \( \varepsilon \)
b:a

\( \text{.0.} \)

\( = \)

\[ \begin{array}{cccccc}
  c: \varepsilon & l: \varepsilon & a: \varepsilon & r: \varepsilon & a: \varepsilon \\
  c: \varepsilon & c: \varepsilon & l: \varepsilon & a: \varepsilon & r: \varepsilon & a: \varepsilon \\
  c: \varepsilon & c: \varepsilon & c: \varepsilon & l: \varepsilon & a: \varepsilon & r: \varepsilon & a: \varepsilon \\
  c: \varepsilon & c: \varepsilon & c: \varepsilon & c: \varepsilon & l: \varepsilon & a: \varepsilon & r: \varepsilon & a: \varepsilon \\
  c: \varepsilon & c: \varepsilon & c: \varepsilon & c: \varepsilon & c: \varepsilon & l: \varepsilon & a: \varepsilon & r: \varepsilon & a: \varepsilon
\end{array} \]
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**: \(p(\text{phone seq} \mid \text{word seq})\)
- **acoustic model**: \(p(\text{acoustics} \mid \text{phone seq})\)
Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model \( p(\text{word seq}) \)

pronunciation model \( p(\text{phone seq} \mid \text{word seq}) \)

acoustic model \( p(\text{acoustics} \mid \text{phone seq}) \)

observed acoustics
Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model

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

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

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

CAT: \( k \, \text{æ} \, t \)
Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model

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

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

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

Tonya Harding
トーニャ・ハーディング
(toonya haadingu)

ramp
ランプ
(ranpu)

lamp
ランプ
(ranpu)

casual fashion
カジュアルファッション
(kajyuaruhasshyon)

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.
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
jumped
```

```
the
brown
fox
quick
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

\[
\begin{align*}
\text{The} & \quad \text{sad} \\
\text{intelligent} & \quad \text{green} \\
\text{fat} & \quad \text{...}
\end{align*}
\]

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
# Penn Treebank POS 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**

<table>
<thead>
<tr>
<th>Fed</th>
<th>raises</th>
<th>interest rates</th>
<th>0.5 %</th>
<th>in effort to control inflation</th>
</tr>
</thead>
</table>

Part-of-speech tags:
- NNP: Proper noun
- VBZ: Verb, 3rd person singular present
- VB: Verb, base form
- NNS: Singular or plural noun
- CD: Cardinal number
- NN: Singular or plural noun
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**

<table>
<thead>
<tr>
<th>PN</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
<th>Prep</th>
<th>Det</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill directed a cortege of autos through the dunes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PN</th>
<th>Adj</th>
<th>Det</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
<th>Prep</th>
<th>Det</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>some possible tags for each word (maybe more)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verb</th>
<th>Verb</th>
<th>Noun</th>
<th>Verb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adj</td>
<td>Prep</td>
<td>...?</td>
<td></td>
</tr>
</tbody>
</table>
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?

Bill directed a cortege of autos through the dunes.

Correct tags

PN Verb Det Noun Prep Noun Prep Det Noun

Bill directed a cortege of autos through the dunes.

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)
...

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
d

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**

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  some  possible  tags  for

Prep  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
Prep  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)

![Diagram of noisy channel model]

- real language $X$
- noisy channel $X \rightarrow Y$
- yucky language $Y$

want to recover $X$ from $Y$
Review: Noisy Channel

real language \( X \)

noisy channel \( X \rightarrow Y \)

yucky language \( Y \)

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

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

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

$p(X)$

$\ast$

$p(Y | X)$

$=$

$p(X,Y)$
Review: Noisy Channel

Real language $X$ 

Noisy channel $X \rightarrow Y$

Yucky language $Y$

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 for Tagging

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

"Markov Model"

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

"Unigram Replacement"

**acceptor:** the observed words

"straight line"

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

\[
p(X) = \sum_a p(a) \cdot p(Y \mid X) \cdot \Pr(Y = y)
\]

\[
p(X, y) = \sum_a p(a) \cdot p(Y \mid X) \cdot \Pr(Y = y)
\]
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)

Start

Det → Adj

Prep → Noun

Verb → Noun

Noun → Stop
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model (bigrams)
Markov Model

Start

Det

Verb

Adj

Prep

Noun

Stop
Markov Model

Start

Det

0.3

Adj

0.7

Noun

Verb

Prep

Stop
Markov Model

Start

Det

Adj

Noun

Verb

Prep

Stop

0.3

0.7

0.4

0.5

0.1
Markov Model

Diagram of a Markov model showing transitions between different parts of speech, such as DET (determiner), Adj (adjective), Noun, Verb, Prep (preposition), and Stop.
Markov Model

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

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

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

Start

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

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

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

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

\[ \text{Start} \xrightarrow{\text{Det} 0.8} \text{Det} \xrightarrow{\text{Adj} 0.3} \text{Adj} \xrightarrow{\text{Adj} 0.4} \text{Noun} \xrightarrow{\varepsilon 0.1} \text{Noun} \xrightarrow{\varepsilon 0.2} \text{Stop} \]

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

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

\[
\begin{align*}
\text{Start} & \quad \text{Det} 0.8 \quad \text{Adj} 0.3 \\
\text{Adj} 0.4 & \quad \rightarrow \quad \text{Adj} \\
\text{Adj} 0.3 & \quad \rightarrow \quad \text{Det} \quad \epsilon 0.2 \\
\text{Det} & \quad \rightarrow \quad \text{Adj} \quad \rightarrow \quad \text{Noun} 0.5 \\
\text{Noun} & \quad \rightarrow \quad \text{Stop} \\
\text{Stop} & \quad = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2
\end{align*}
\]
Noisy Channel for Tagging

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

“Markov Model”

\( p(X) \)

transducer: \( \text{tags} \rightarrow \text{words} \)

“Unigram Replacement”

\( p(Y | X) \)

automaton: \( \text{the observed words} \)

“straight line”

\( p(y | Y) \)

transducer: scores candidate tag seqs on their joint probability with obs words; pick best path

\( p(X, y) \)
Noisy Channel for Tagging

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

* \[ = \]

\[ p(y | Y) \]

\[ \text{transducer: scores candidate tag seqs on their joint probability with obs words; we should pick best path} \]
Unigram Replacement Model

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

Det: the/0.4
Det: a/0.6
Noun: Bill/0.002
Noun: autos/0.001
Noun: cortege/0.000001
Adj: cool/0.003
Adj: directed/0.0005
Adj: cortege/0.000001
...

sums to 1

sums to 1
Compose

p(tag seq)
**Compose**

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

word seq

the → cool → directed → autos →
Compose with

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

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

\[ p(\text{word seq, tag seq}) = p(\text{tag seq}) \times p(\text{word seq} | \text{tag seq}) \]
The best path:

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

\[
\text{the} \quad \text{cool} \quad \text{directed} \quad \text{autos}
\]

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

$p(\text{word seq, tag seq})$

The best path:

Start Det Adj Adj Noun Stop $= 0.32 \times 0.0009 \ldots$

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

The best path:
Start Det Adj Adj Noun Stop = 0.32 * 0.0009 ...
the cool directed autos
In Fact, Paths Form a “Trellis”

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

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

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

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

Trellis has no Det → Det or Det → Stop arcs; why?

The best path:

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

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

Lattice is missing some other arcs; why?

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
Actually, Trellis Isn’t Complete

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

Lattice is missing some states; why?

The best path:

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

\[ \text{the} \quad \text{cool} \quad \text{directed} \quad \text{autos} \]
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
In Summary

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

```
Bill  directed  a  cortege  of  autos  through
```

```
Start  PN  Verb  Det  Noun  Prep  Noun  Prep  Noun  Prep

Noun  Stop

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”:

```
<table>
<thead>
<tr>
<th>PN</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
<th>Prep</th>
<th>Det</th>
</tr>
</thead>
</table>
```

```
Bill directed a cortege of autos through the dunes
```

```
probs from tag bigram 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>Stop</th>
<th>0.4</th>
<th>0.6</th>
<th>0.001</th>
</tr>
</thead>
</table>

```
Bill directed a cortege of autos through the dunes
```
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>PN</th>
<th>Verb</th>
<th>Det</th>
<th>Noun</th>
<th>Prep</th>
<th>Noun</th>
<th>Prep</th>
<th>Det</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill directed a cortege of autos through the dunes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

probs from tag bigram model

probs from unigram replacement
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”:

```plaintext
probs from tag bigram model

| Start | PN | Verb | Det | Noun | Prep | Noun | Prep | Noun | BILL | directed | a | cortege | of | autos | through |
|-------|----|------|-----|------|------|------|------|------|------|-----------|---|---------|    |       |          |
|       |    |      |     |      |      |      |      |      | 0.4  |           | 0.6 |         |    |       |          |
|       |    |      |     |      |      |      |      |      |      | 0.001      |    |         |    |       |          |
| STOP  |    |      |     |      |      |      |      |      |      |             |    |         |    |       |          |

Find \( X \) that maximizes probability product
Another Viewpoint
Another Viewpoint

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

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

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

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

\[
\begin{array}{cccccccc}
\text{Start} & \text{PN} & \text{Verb} & \text{Det} & \text{Noun} & \text{Prep} & \text{Noun} & \text{Prep} & \text{Det} & \text{Noun} & \text{Stop} \\
\text{Bill} & \text{directed} & \text{a} & \text{cortege of} & \text{autos} & \text{through} & \text{the} & \text{dunes} \\
\end{array}
\]
Another Viewpoint

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

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

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

$$p(\text{Start PN Verb Det ... \text{Bill directed a ...}}) = p(\text{Start}) \cdot p(\text{PN | Start}) \cdot p(\text{Verb | Start PN}) \cdot p(\text{Det | Start PN Verb}) \cdot ...$$

* $p(\text{Bill | Start PN Verb ...}) \cdot p(\text{directed | Bill, Start PN Verb Det ...}) \cdot p(\text{a | Bill directed, Start PN Verb Det ...}) \cdot ...$

```
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 \( P(x_{t+1}|x_1, ... x_t) = P(x_{t+1}|x_t) \)
  
  - 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.

\[
P(o_t|x_1, ... x_T, o_1, ... o_{t-1}) = P(o_t|x_t)
\]
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)
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

\begin{align*}
\text{States} & \quad \text{Time} \\
x1 & \quad 1 \\
x2 & \quad 2 \\
x3 & \quad 3 \\
x4 & \quad 4 \\
\cdots & \quad \ldots \\
T & \quad T
\end{align*}
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
  \[ \hat{x}_T = \arg \max_{i=1..N} \delta_i(T) \]
  \[ \hat{x}_t = \psi_{\hat{x}_{t+1}}(t + 1) \quad \text{Probability of entire best seq.} \]
  \[ P(\hat{X}) = \max_{i=1..N} \delta_i(T) \]
HMMs: Maxing and Summing
Markov vs. Hidden Markov Models

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

Fed raises interest rates

NN NNS VBZ raises interest ...

interest ...

raises rates ...

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

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

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

interest...
Markov vs. Hidden Markov Models
# Unrolled into a Trellis

| NN  | Fedora raises interest rates |
| NNS | Fedora raises interest rates |
| NNP | Fedora raises interest rates |
| VB  | Fedora raises interest rates |
| VBZ | Fedora raises interest rates |
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) 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 Dijsktra’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)

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)

Fed raises interest rates
Viterbi Algorithm (Tagging)

Fed raises interest rates
Viterbi Algorithm (Tagging)

Fed raises interest rates
Viterbi Algorithm (Tagging)

\[
\max = \delta_{VB}(3)
\]

Fed raises interest rates
Forward Algorithm (LM)

Fed raises interest rates
Forward Algorithm (LM)

Fed raises interest rates
Forward Algorithm (LM)

Fed raises interest rates
### Forward Algorithm (LM)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
Fed raises interest rates

[Diagram with Forward Algorithm (LM)]
Forward Algorithm (LM)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Fed</th>
<th>raises</th>
<th>interest</th>
<th>rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Forward Algorithm (LM)

Fed raises interest rates
Forward Algorithm (LM)

Fed raises interest rates
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?

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

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

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

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

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

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

- Probability of all paths from the beginning to word \( t \) such that word \( t \) has tag \( j \)
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):
    • $a_{\text{VBZ} | \text{NN}} = \frac{C(\text{NN,VBZ})}{C(\text{NN})}$
    • $b_{\text{rates} | \text{VBZ}} = \frac{C(\text{VBZ,rates})}{C(\text{VBZ})}$

• Unsupervised
  • Train and test on plain text
  • What can we do?
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

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

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

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

NN
NNS
NNP
VB
VBZ
Fed raises interest rates
Forward-Backward Algorithm

NN

NNS

NNP

VB

VBZ

Fed raises interest rates
Forward-Backward Algorithm

NN
NNS
NNP
VB
VBZ

Fed raises interest rates

α_{NN}(2) a[VB|NN]b[interest|VB] a[NN|VB]b[rates|NN] β_{NN}(4)

α_{NNS}(2) a[VB|NNS]b[interest|VB] a[NNS|VB]b[rates|NNS]

α_{NNP}(2) a[VB|NNP]b[interest|VB] a[NNP|VB]b[rates|NNP]

α_{VB}(2) a[VB|VB]b[interest|VB] a[VB|VB]b[rates|VB]

α_{VBZ}(2) a[VB|VBZ]b[interest|VB] a[VBZ|VB]b[rates|VBZ]
Forward-Backward Algorithm

Fed raises interest rates

NN
NNS
NNP
VB
VBZ

α(2)

β(4)

α NN(2)
α NNS(2)
α NNP(2)
α VB(2)
α VBZ(2)
Forward-Backward Algorithm

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

\[ \beta_{NN}(4) \quad \beta_{NNS}(4) \quad \beta_{NNP}(4) \quad \beta_{VB}(4) \quad \beta_{VBZ}(4) \]

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates
Forward-Backward Algorithm

Fed raises interest rates

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

\[ \beta_{NN}(4) \quad \beta_{NNS}(4) \quad \beta_{NNP}(4) \quad \beta_{VB}(4) \quad \beta_{VBZ}(4) \]
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 absence 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>×</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Max</td>
<td>$\mathbb{R}^+$</td>
<td>max</td>
<td>×</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>-∞</td>
<td>0</td>
</tr>
<tr>
<td>“Tropical”</td>
<td>$\mathbb{R} \cup {\pm \infty}$</td>
<td>max</td>
<td>+</td>
<td>-∞</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>${F, T}$</td>
<td>V</td>
<td>∧</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>
Search as Deduction

Axioms \( path(\text{Start}, 0), \text{word}('the', 0, 1), \text{emit}(\text{DT}, 'the'), \ldots \)

Inference rule

\[
\forall A, B \in T; W \in V; 0 \leq i, j \leq n \\
\text{path}(B, j) \iff \text{path}(A, i) \land \text{word}(W, i, j) \\
\quad \land \text{emit}(B, W) \land \text{trans}(A, B)
\]

In Prolog

\begin{verbatim}
path(B,J) :-
    path(A,I), word(W,I,J), emit(B,W), trans(A,B).
path("Start",0).
word("the",0,1).
word("cool",1,2).
...
emit("DT","the").
...
\end{verbatim}
Search as Deduction

Axioms \[ \text{path(Start, 0), word(the, 0, 1), emit(DT, the), \ldots} \]

Inference rule

\[ \forall B, j : \text{path}(B, j) = \bigvee_{A, W, i} \text{path}(A, i) \land \text{word}(W, i, j) \land \text{emit}(B, W) \land \text{trans}(A, B) \]

In Prolog

\begin{verbatim}
path(B,J) :-
    path(A,I), word(W,I,J), emit(B,W), trans(A,B).
p(path("Start",0).
word("the",0,1).
word("cool",1,2).
... emit("DT","the").
...
\end{verbatim}
Search as Deduction

Axioms \( \text{path}(\text{Start}, 0), \text{word}(\text{the}, 0, 1), \text{emit}(\text{DT}, \text{the}), \ldots \) 

Inference rule

\[ \forall B, j : \text{path}(B, j) = \bigvee_{A, W, i} \text{path}(A, i) \land \text{word}(W, i, j) \]

\[ \land \text{emit}(B, W) \land \text{trans}(A, B) \]

Shortest path

\[ \forall B, j : \text{path}(B, j) = \min_{A, W, i} \text{path}(A, i) + \text{word}(W, i, j) \]

\[ + \text{emit}(B, W) + \text{trans}(A, B) \]
Search as Deduction

**Axioms**

\[ \text{path(Start, 0), word(the, 0, 1), emit(DT, the), \ldots} \]

**Shortest path**

\[ \forall B, j : \text{path}(B, j) = \min_{A,W,i} \text{path}(A, i) + \text{word}(W, i, j) \]
\[ + \text{emit}(B, W) + \text{trans}(A, B) \]

**Viterbi algorithm**

\[ \forall B, j : \text{path}(B, j) = \max_{A,W,i} \text{path}(A, i) \cdot \text{word}(W, i, j) \]
\[ \cdot \text{emit}(B, W) \cdot \text{trans}(A, B) \]
Search as Deduction

Axioms \( \text{path}(\text{Start}, 0), \text{word}(\text{the}, 0, 1), \text{emit}(\text{DT, the}), \ldots \)

Viterbi algorithm

\[
\forall B, j : \text{path}(B, j) = \begin{cases} 
\max_{A,W,i} \text{path}(A, i) \cdot \text{word}(W, i, j) \\
\cdot \text{emit}(B, W) \cdot \text{trans}(A, B)
\end{cases}
\]

Viterbi w/log probabilities

\[
\forall B, j : \text{path}(B, j) = \begin{cases} 
\max_{A,W,i} \text{path}(A, i) + \text{word}(W, i, j) \\
+ \text{emit}(B, W) + \text{trans}(A, B)
\end{cases}
\]
Search as Deduction

Axioms \( \text{path(Start, 0), word(the, 0, 1), emit(DT, the), ...} \)

Viterbi algorithm

\[
\forall B, j : \text{path}(B, j) = \max_{A, W, i} \text{path}(A, i) \cdot \text{word}(W, i, j) \\
\cdot \text{emit}(B, W) \cdot \text{trans}(A, B)
\]

Forward algorithm

\[
\forall B, j : \text{path}(B, j) = \sum_{A, W, i} \text{path}(A, i) \cdot \text{word}(W, i, j) \\
\cdot \text{emit}(B, W) \cdot \text{trans}(A, B)
\]
Search as Deduction

Axioms  \( path(\text{Start}, 0), word(\text{the}, 0, 1), emit(\text{DT, the}), \ldots \)

Forward algorithm

\[
\forall B, j : path(B, j) = \sum_{A,W,i} path(A, i) \cdot word(W, i, j) \\
\cdot emit(B, W) \cdot trans(A, B)
\]

Let \( \theta = \) subset of axioms whose weights we wish to optimize

\[
goal = \sum_B path(B, n)
\]

Chain rule

\[
\frac{\partial goal}{\partial \theta} = \sum_B \frac{\partial goal}{\partial path(B, n)} \cdot \frac{\partial path(B, n)}{\partial \theta}
\]
Search as Deduction

Axioms  \( \text{path(Start}, 0), \text{word(the, 0, 1), emit(DT, the), \ldots} \)

Forward algorithm

\[ \forall B, j : \text{path}(B, j) = \sum_{A,W,i} \text{path}(A, i) \cdot \text{word}(W, i, j) \cdot \text{emit}(B, W) \cdot \text{trans}(A, B) \]

Chain rule

\[ \frac{\partial \text{goal}}{\partial \text{path}(A, i)} = \sum_{B,j} \frac{\partial \text{goal}}{\partial \text{path}(B, j)} \cdot \frac{\partial \text{path}(B, j)}{\partial \text{path}(A, i)} \]

\[ \beta_A(i) = \sum_{B,W,j} \beta_B(j) \cdot \text{word}(W, i, j) \cdot \text{emit}(B, W) \cdot \text{trans}(A, B) \]
Reading


• Background: Jurafsky & Martin, ch. 5 and 6.1–6.5