6.863J Natural Language Processing
Lecture 5: Finite state machines & part-of-speech tagging

Instructor: Robert C. Berwick
The Menu Bar

- Administrivia:
  - Schedule alert: Lab1 due next Weds (Feb 24)
  - Lab 2, handed out Feb 24 (look for it on the web as laboratory2.html; due the Weds after this – March 5

- Agenda:
  - Part of speech ‘tagging’ (with sneaky intro to probability theory that we need)
  - Ch. 6 & 8 in Jurafsky; see ch. 5 on Hidden Markov models
Two finite-state approaches to tagging

1. Noisy Channel Model (statistical)

2. Deterministic baseline tagger composed with a cascade of fixup transducers

• PS: how do we *evaluate* taggers? (and such statistical models generally?)

• 1, 2, & evaluation = Laboratory 2
The real plan...

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

Find \( x \) that maximizes this quantity
transducer: scores candidate tag seqs on their joint probability with obs words; we should pick best path
What’s the big picture? Why NLP?

Computers would be a lot more useful if they could handle our email, do our library research, talk to us ...

But they are fazed by natural human language.

How can we tell computers about language? (Or help them learn it as kids do?)
What is NLP for, anyway?

• If we could do it perfectly, we could pass the Turing test (more on this below)
• Two basic ‘engineering’ tasks – and third scientific one
• Text-understanding
• Information extraction
• ?What about how people ‘process’ language???  [psycholinguistics]
Some applications...

- Spelling correction, grammar checking ...
- Better search engines
- Information extraction
- Language identification (English vs. Polish)
- Psychotherapy; Harlequin romances; etc.
- And: plagiarism detection - www.turnitin.com
- For code: www.cs.berkeley.edu/~aiken/moss.html
- New interfaces:
  - Speech recognition (and text-to-speech)
  - Dialogue systems (USS Enterprise onboard computer)
  - Machine translation (the Babel fish)
Text understanding is very hard

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

- NL *relies on* ambiguity! (Why?)
- “We haven’t had a sale in 40 years”
What’s hard about the story?

John stopped at the *donut* store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

To get a donut (spare tire) for his car?
What’s hard?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

store where donuts shop? or is run by donuts? or looks like a big donut? or made of donut? or has an emptiness at its core?

(Think of five other issues...there are lots)
John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

Describes where the store is? Or when he stopped?
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

Well, actually, he stopped there from hunger and exhaustion, not just from work.
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

At that moment, or habitually?
(Similarly: Mozart composed music.)
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

That’s how often he thought it?
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

But actually, a coffee only stays good for about 10 minutes before it gets cold.
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

Similarly: In America a woman has a baby every 15 minutes. Our job is to find that woman and stop her.
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

the particular coffee that was good every few hours? the donut store? the situation?
What’s hard about this story?

John stopped at the donut store on his way home from work. He thought a coffee was good every few hours. But it turned out to be too expensive there.

too expensive for what? what are we supposed to conclude about what John did?

how do we connect “it” to “expensive”?
Example tagsets

- 87 tags - Brown corpus
- Three most commonly used:
  1. Small: 45 Tags - Penn treebank (Medium size: 61 tags, British national corpus)
  2. Large: 146 tags

Big question: have we thrown out the right info? Impoverished? How?
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 Homer Simpson algorithm:
      • Tag every word with its most frequent tag
      • Tag unknown words as nouns

• How well do people do?
Knowledge-based (rule-based) vs. Statistically-based systems
A picture: the statistical, noisy channel view

Wreck a nice beach?  
Reckon eyes peach?  
Recognize speech?

\[ x(\text{speech}) \xrightarrow{\text{Acoustic Model} \ P(x|y)} y(\text{text}) \xrightarrow{\text{Language Model} \ P(y)} \]
Language models, probability & info

• Given a string $w$, a language model gives us the probability of the string $P(w)$, e.g.,
  • $P(\text{the big dog}) > (\text{dog big the}) > (\text{dgo gib eth})$
  • Easy for humans; difficult for machines
  • Let $P(w)$ be called a language model
Language models – statistical view

- Application to speech recognition (and parsing, generally)
  - x = Input (speech)
  - y = output (text)
  - We want to find \( \max P(y|x) \) Problem: we don’t know this!
  - Solution: We have an estimate of \( P(y) \) [the language model] and \( P(x|y) \) [the prob. of some sound given text = an acoustic model]
  - From Bayes’ law, we have,
    \[
    \max P(y|x) = \max P(x|y) \cdot P(y) = \max Pr \text{ acoustic model } x \text{ lang model}
    \]
    (hold \( P(x) \) fixed, i.e., \( P(x|y) \cdot P(y) / P(x) \), but max is same for both)
Can be generalized to Turing test...!

If we could solve this (the?) NLP problem, we could solve the Turing test.... (but not the Twain test)! Language modeling solves the Turing test!

- Interrogator
- q?
- a!
- Human
- q
- P(a|q) = \text{pr human answers } q \text{ with } a = \frac{P(q,a)}{P(q)}
- \text{b!}
- Computer
- q?
- P(b|q) = \text{pr computer answers } q \text{ with } b = \frac{P(q,b)}{P(q)}

If P(q,a) = P(q,b) then computer successfully imitates human
Some applications...

- Spelling correction, grammar checking ...
- Better search engines
- Information extraction
- Language identification (English vs. Polish)
- Psychotherapy; Harlequin romances; etc.
- And: plagiarism detection - www.turnitin.com
- For code: www.cs.berkeley.edu/~aiken/moss.html
- New interfaces:
  - Speech recognition (and text-to-speech)
  - Dialogue systems (USS Enterprise onboard computer)
  - Machine translation (the Babel fish)
What’s this stuff *for* anyway?
Information extraction

- Information extraction involves processing text to identify selected information:
  - particular types of names
  - specified classes of events.
  - For names, it is sufficient to find the name in the text and identify its type
  - for events, we must extract the critical information about each event (the agent, objects, date, location, etc.) and place this information in a set of templates (data base)
SANTIAGO, 18 MAY 90 (RADIO COOPERATIVA NETWORK) -- [REPORT] [JUAN ARAYA] [TEXT]
EDMUNDO VARGAS CARRENO, CHILEAN FOREIGN MINISTRY UNDER SECRETARY, HAS STATED THAT THE BRYANT TREATY WITH THE UNITED STATES WILL BE APPLIED IN THE LETELIER CASE ONLY TO COMPENSATE THE RELATIVES OF THE FORMER CHILEAN FOREIGN MINISTER MURDERED IN WASHINGTON AND THE RELATIVES OF HIS U.S. SECRETARY, RONNIE MOFFIT. THE CHILEAN FOREIGN UNDER SECRETARY MADE THIS STATEMENT IN REPLY TO U.S. NEWSPAPER REPORTS STATING THAT THE TREATY WOULD BE PARTIALLY RESPECTED.

FOLLOWING ARE VARGAS CARRENO'S STATEMENTS AT A NEWS CONFERENCE HE HELD IN BUENOS AIRES BEFORE CONCLUDING HIS OFFICIAL VISIT TO ARGENTINA:
Extracted info – names, events

0. MESSAGE: ID
   TST1-MUC3-0011

1. MESSAGE: TEMPLATE
   1

2. INCIDENT: DATE
   18 MAY 90

3. INCIDENT: LOCATION
   UNITED STATES: WASHINGTON D.C. (CITY)

4. INCIDENT: TYPE
   ATTACK

5. INCIDENT: STAGE OF EXECUTION
   ACCOMPLISHED

6. INCIDENT: INSTRUMENT ID

7. INCIDENT: INSTRUMENT TYPE

8. PERP: INCIDENT CATEGORY
   STATE-SPONSORED VIOLENCE

9. PERP: INDIVIDUAL ID

10. PERP: ORGANIZATION ID
    "CHILEAN GOVERNMENT"

11. PERP: ORGANIZATION
    CONFIDENCE REPORTED AS FACT: "CHILEAN GOVERNMENT"

12. PHYS TGT: ID

13. PHYS TGT: TYPE

14. PHYS TGT: NUMBER

15. PHYS TGT: FOREIGN NATION

16. PHYS TGT: EFFECT OF INCIDENT

17. PHYS TGT: TOTAL NUMBER

18. HUM TGT: NAME
    "ORLANDO LETELIER"
    "RONNIE MOFFIT"

19. HUM TGT: DESCRIPTION
    "FORMER CHILEAN FOREIGN MINISTER": "ORLANDO LETELIER"
    "U.S. SECRETARY" / "ASSISTANT" / "SECRETARY": "RONNIE MOFFIT"

20. HUM TGT: TYPE
    GOVERNMENT OFFICIAL: "ORLANDO LETELIER"
    CIVILIAN: "RONNIE MOFFIT"

21.
Text understanding vs. Info extraction

- For information extraction:
  - generally only a fraction of the text is relevant; for example, in the case of the MUC-4 terrorist reports, probably only about 10% of the text was relevant;
  - information is mapped into a predefined, relatively simple, rigid target representation; this condition holds whenever entry of information into a database is the task;
  - the subtle nuances of meaning and the writer's goals in writing the text are of at best secondary interest.
Properties of message understanding task

- Simple, fixed definition of the information to be sought.
- Much, or even most, of the text is irrelevant to the information extraction goal.
- Large volumes of text need to be searched.
Text understanding

- the aim is to make sense of the entire text;
- the target representation must accommodate the full complexities of language;
- one wants to recognize the nuances of meaning and the writer's goals.
Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan. The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and "metal wood" clubs a month.

TIE-UP-1:
Relationship: TIE-UP
Entities: "Bridgestone Sports Co."
"a local concern"
"a Japanese trading house"
Joint Venture Company: "Bridgestone Sports Taiwan Co." Activity: ACTIVITY-1
Amount: NT$20000000

ACTIVITY-1:
Activity: PRODUCTION
Company: "Bridgestone Sports Taiwan Co."
Product: "iron and `metal wood' clubs"
Start Date: DURING: January 1990
Applications

- **IR tasks:**
  - Routing queries to prespecified topics
  - Text classification/routing
- **Summarization**
  - Highlighting, clipping
  - NL generation from formal output representation
- **Automatic construction of knowledge bases from text**
Message understanding tasks

Business News: Joint Ventures (English and Japanese),
Labor Negotiations, Management Succession
Geopolitical News: Terrorist Incidents
Military Messages: DARPA Message Handler
Legal English: Document Analysis Tool
Integration with OCR
Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.
Shallow parsing or ‘chunking’ – Lab 3

Bridgestone Sports Co. said Friday it has set up a joint venture

in Taiwan with a local concern and a Japanese trading house
to produce golf clubs to be shipped to Japan.
Recognizing domain patterns

- Match domain patterns against complex phrase heads
  - <company> <form> <joint venture> with <company>
  - <company> capitalized at <currency>

"Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan. "The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990."

<table>
<thead>
<tr>
<th>Relationship:</th>
<th>TIE-UP</th>
</tr>
</thead>
</table>
| Entities:     | "Bridgestone Sports Co."  
                "a local concern"  
                "a Japanese trading house" |
| JV Company:   | --     |
| Capitalization: | --     |

<table>
<thead>
<tr>
<th>Relationship:</th>
<th>TIE-UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities:</td>
<td>--</td>
</tr>
<tr>
<td>JV Company:</td>
<td>&quot;Bridgestone Sports Taiwan Co.&quot;</td>
</tr>
<tr>
<td>Capitalization:</td>
<td>20000000 TWD</td>
</tr>
</tbody>
</table>
What about part of speech tagging here?

- **Advantages**
  - Ambiguity can be potentially reduced (but we shall see in our laboratory if this is true)
  - Avoid errors due to incorrect categorization of rare senses e.g., “has been” as noun

- **Disadvantages**
  - Errors taggers make often those you’d most want to eliminate
  - High performance requires training on similar genre
  - Training takes time
Proper names...

- Proper names are particularly important for extraction systems.
- Because typically one wants to extract events, properties, and relations about some particular object, and that object is usually identified by its name.
...A challenge...

- Problems though...
  - proper names are huge classes and it is difficult, if not impossible to enumerate their members
  - Hundreds of thousands of names of locations around the world
  - Many of these names are in languages other than the one in which the extraction system is designed
How are names extracted?

- (Hidden) Markov Model
- Hypothesized that there is an underlying finite state machine (not directly observable, hence hidden) that changes state with each input element
- Probability of a recognized constituent is conditioned not only on the words seen, but the state that the machine is in at that moment
- “John” followed by “Smith” is likely to be a person, while “John” followed by “Deere” is likely to be a company (a manufacturer of heavy farming and construction equipment).
HMM statistical name tagger
HMM

- Whether a word is part of a name or not is a random event with an estimable probability.
- The probability of name versus non-name readings can be estimated from a training corpus in which the names have been annotated.
- In a Hidden Markov model, it is hypothesized that there is an underlying finite state machine (not directly observable, hence hidden) that changes state with each input element.
- The probability of a recognized constituent is conditioned not only on the words seen, but the state that the machine is in at that moment.
- “John” followed by “Smith” is likely to be a person, while “John” followed by “Deere” is likely to be a company (a manufacturer of heavy farming and construction equipment).
HMM construction

- Hidden state transition model governs word sequences
- Transitions probabilistic
- Estimate transition probabilities from an annotated corpus
  - \( P(s_j \mid s_{j-1}, w_j) \)
  - Based just on prior state and current word seen (hence Markovian assumption)
- At runtime, find maximum likelihood path through the network, using a max-flow algorithm (Viterbi)
How much data is needed?

- System performance bears a roughly log-linear relationship to the training data quantity, at least up to about 1.2 million words.
- Obtaining 1.2 million words of training data requires transcribing and annotating approximately 200 hours of broadcast news programming, or if annotating text, this would amount to approximately 1,777 average-length Wall Street Journal articles.
If you think name recog is not relevant, then...

- Microsoft announced plans to include “Smart Tags” in its browser and other products. This is a feature that automatically inserts hyperlinks from concepts in text to related web pages chosen by Microsoft.
- The best way to make automatic hyperlinking unbiased is to base it on an unbiased source of web pages, such as Google.
How to do this?

- The main technical problem is to find pieces of text that are good concept anchors… like names!
- So: Given a text, find the starting and ending points of all the names. Depending on our specific goals, we can include the names of people, places, organizations, artifacts (such as product names), etc.
- Once we have the anchor text, we can send it to a search engine, retrieve a relevant URL (or set of URLs, once browsers can handle multi-way hyperlinks), and insert them into the original text on the fly.
OK, back to the tagging task

- This will illustrate all the issues with name recognition too
# Brown/Upenn corpus tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordin. Conjunction</td>
<td><em>and, but, or</em></td>
<td>SYM</td>
<td>Symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td><em>one, two, three</em></td>
<td>TO</td>
<td>“to”</td>
<td>to</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td><em>a, the</em></td>
<td>UH</td>
<td>Interjection</td>
<td><em>ah, oops</em></td>
</tr>
<tr>
<td>EX</td>
<td>Existential ‘there’</td>
<td><em>there</em></td>
<td>VB</td>
<td>Verb, base form</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
<td><em>mea culpa</em></td>
<td>VBD</td>
<td>Verb, past tense</td>
<td><em>ate</em></td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/sub-conj</td>
<td><em>of, in, by</em></td>
<td>VBG</td>
<td>Verb, gerund</td>
<td><em>eating</em></td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td><em>yellow</em></td>
<td>VBN</td>
<td>Verb, past participle</td>
<td><em>eaten</em></td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td><em>bigger</em></td>
<td>VBP</td>
<td>Verb, non-3sg pres</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>JJS</td>
<td>Adj., superlative</td>
<td><em>wildest</em></td>
<td>VBJ</td>
<td>Verb</td>
<td></td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td><em>1, 2, One</em></td>
<td>VBG</td>
<td>Verb, 3sg pres</td>
<td><em>eats</em></td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td><em>can, should</em></td>
<td>WDT</td>
<td>Wh-determiner</td>
<td><em>which, that</em></td>
</tr>
<tr>
<td>NN</td>
<td>Noun, sing. or mass</td>
<td><em>llama</em></td>
<td>WP</td>
<td>Wh-pronoun</td>
<td><em>what, who</em></td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td><em>llamas</em></td>
<td>WP$</td>
<td>Possessive wh-</td>
<td><em>whose</em></td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td><em>IBM</em></td>
<td>WRB</td>
<td>Wh-adverb</td>
<td><em>how, where</em></td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
<td><em>Carolinias</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
<td><em>all, both</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td><em>’s</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP</td>
<td>Personal pronoun</td>
<td><em>I, you, he</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP$</td>
<td>Possessive pronoun</td>
<td><em>your, one’s</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td><em>quickly, never</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
<td><em>faster</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
<td><em>fastest</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
<td><em>up, off</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**J. text, p. 297**

**Fig 8.6**

1M words

60K tag counts
Bill directed a cortege of autos through the dunes.

Correct tags:

- **PN** Bill
- **Adj** directed
- **Det** a
- **Noun** cortege
- **Prep** of
- **Noun** autos
- **Prep** through
- **Det** the
- **Noun** dunes

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 ...
Each unknown tag is *constrained* by its word and by the tags to its immediate left and right. But those tags are unknown too ...
Ok, what should we look at?

Correct tags:

Bill directed a cortege of autos through the dunes.

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 ...
Ok, what *should* we look at?

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

- Noishy Chunnel Muddle (statistical)

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

want to recover $X$ from $Y$
Noisy channel – and prob intro

real language $X$

noisy channel $X \rightarrow Y$

yucky language $Y$

$p(X)$

* $p(Y \mid X)$

= $p(X,Y)$

choose sequence of tags $X$ that \textit{maximizes} $p(X \mid Y)$ [oops... this isn't \textit{quite} correct... need 1 more step]
Two approaches

- Fix up Homer Simpson idea with more than unigrams – look at tag sequence
- Go to more powerful Hidden Markov model
What are unigrams and bigrams?

- Letter or word frequencies: 1-grams
  - useful in solving cryptograms: ETAOINSHRDLU...
- If you know the previous letter: 2-grams
  - “h” is rare in English (4%; 4 points in Scrabble)
  - but “h” is common after “t” (20%)
- If you know the previous 2 letters: 3-grams
  - “h” is really common after “ ” “t” etc. ...
In our case

- Most likely word? Most likely tag $t$ given a word $w$? $= P(\text{tag}|\text{word})$
- Task of predicting the next word
- Woody Allen:
  “I have a gub”
In general: predict the $N^{\text{th}}$ word (tag) from the preceding $N-1$ word (tags) aka $\text{N-gram}$
Homer Simpson: just use the current word (don’t look at context) = unigram (1-gram)
Where do these probability info estimates come from?

- Use tagged corpus e.g. “Brown corpus” 1M words (fewer token *instances*); many others – Celex 16M words
- Use *counts* (relative frequencies) as estimates for probabilities (various issues w/ this, these so-called *Maximum-Likelihood estimates* – don’t work well for low numbers)
- Train on texts to get estimates – use on new texts
General probabilistic decision problem

- E.g.: data = bunch of text
  - label = language
  - label = topic
  - label = author
- E.g.2: (sequential prediction)
  - label = translation or summary of entire text
  - label = part of speech of current word
  - label = identity of current word (ASR) or character (OCR)
Formulation, in general

\[
Label = \arg \max_{Label} \Pr(Label \mid Data)
\]
How far should we go?

• “long distance___”
• Next word? Call?
• $p(w_n|w$
• Consider special case above
• Approximation says that
  $|long\ distance\ call|/|distance\ call| \approx |distance\ call|/|distance|$
• If context 1 word back = bigram
But even better approx if 2 words back: long distance___
  Not always right: long distance runner/long distance call
Further you go: collect long distance______
Bigrams, fsa’s, and Markov models – take two

• We approximate \( p(\text{tag} | \text{all previous tags}) \)
  Instead of
  \( p(\text{rabbit} | \text{Just then the white...}) \) we use:
  \( P(\text{rabbit} | \text{white}) \)

• This is a **Markov assumption** where past memory is limited to immediately previous state – just 1 state corresponding to the previous word or tag
Forming classes

• “n-gram” = sequence of n “words”
  • unigram
  • bigram
  • trigram
  • four-gram

• In language modeling, the conditioning variables are sometimes called the “history” or the “context.”

• The Markov assumption says that the prediction is conditionally independent of ancient history, given recent history.

• I.e., we divide all possible histories into equiv. classes based on the recent history.
3-gram

[Genmetheyesse orils of Ted you doorder [6], the Grily Capiduantent pildred and For thy werarme: nomiterst halt i, what production the Covers, in calt cations on wile ars, was name conch rom the exce of the man, Winetwentagaint up, and and All. And of Ther so i hundal panite days th the res of th rand ung into the forD six es, wheralf the hie soulsee, frelatche rigat. And the LOperact camen unismelfaret fammedied: and nople,
[1] By the returall benefit han every familitant of all thou go? And At the eld to parises of the nursed by thy way of all histantly be the ~aciedfag . to the narre gread abrasa of thing, and vas these conwuning clann com to one language; all Lah, which for the greath othey die. -
[Gen 3:1] In the called up history of its opposition of bourgeoIS AND Adam to rest, that the existing of heaven; and land the bourgeoIS ANger anything but concealed, the land whethere had doth know ther: bury thy didst of Terature their faces which went masses the old society [2] is the breaks out of oppressor of all which, the prolETARiat goest, unto German pleast twelves applied in manner with these, first of this polities have all
[Gen 4:25] And Adam gave names to all feudal, patriarchal, idyllic relations. It has but established new classes, new conditions of oppression, new form of struggle in place of the West? The bourgeoisie keeps more and more splitting up into two great lights; the greater light to rule the day of my house is this Eliezer of Damascus.

How far can we go??
Shakespeare in lub...
The unkindest cut of all

- Shakespeare: 884,647 words or *tokens* (Kucera, 1992)
- 29,066 *types* (incl. proper nouns)
- So, \# bigrams is $29,066^2 > 844$ million. 1 million word training set doesn’t cut it – only 300,000 different bigrams appear
- Use *backoff* and *smoothing*
- So we can’t go very far...
Reliability vs. discrimination

“large green ____________”

tree? mountain? frog? car?

“swallowed the large green ____________”

pill? broccoli?
Reliability vs. discrimination

- larger n: more information about the context of the specific instance (greater discrimination)

- smaller n: more instances in training data, better statistical estimates (more reliability)
Choosing $n$

Suppose we have a vocabulary $(V) = 20,000$ words

<table>
<thead>
<tr>
<th>$n$</th>
<th>Number of bins</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (bigrams)</td>
<td>400,000,000</td>
</tr>
<tr>
<td>3 (trigrams)</td>
<td>8,000,000,000,000,000</td>
</tr>
<tr>
<td>4 (4-grams)</td>
<td>$1.6 \times 10^{17}$</td>
</tr>
</tbody>
</table>
Statistical estimators

Example:

Corpus: five Jane Austen novels
N = 617,091 words
V = 14,585 unique words

Task: predict the next word of the trigram “inferior to ________”

from test data, *Persuasion*:

“[In person, she was] inferior to both [sisters.]”
Instances in the Training Corpus:
“inferior to ________”

<table>
<thead>
<tr>
<th>Word</th>
<th>C(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>2</td>
</tr>
<tr>
<td>her</td>
<td>1</td>
</tr>
<tr>
<td>cherries</td>
<td>1</td>
</tr>
<tr>
<td>Maria</td>
<td>1</td>
</tr>
<tr>
<td>word 5</td>
<td>1</td>
</tr>
<tr>
<td>word 6</td>
<td>1</td>
</tr>
<tr>
<td>word 7</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Maximum likelihood estimate
Actual probability distribution

C(w) vs. words: the, her, cherries, Maria, word 5, word 6, word 7, ...
Comparison
Smoothing

- Develop a model which decreases probability of seen events and allows the occurrence of previously unseen n-grams
- a.k.a. “Discounting methods”
- “Cross-Validation” – Smoothing methods which utilize a second batch of data.
How to use $Pr(\text{Label} \mid \text{Data})$?

- Assume we have a training corpus, i.e. a bunch of Data with their Labels.
- Then we can compute Maximum Likelihood Estimates (MLEs) of the probabilities as relative frequencies:
  \[
  Pr(\text{Label} \mid \text{Data}) = \frac{\text{count}(\text{Label, Data})}{\text{count}(\text{Data})}
  \]
- To predict a Label for a new bunch of Data, we just look up $Pr(\text{Label} \mid \text{Data})$ for all possible Labels and output the argmax.
Practical limitations

• Equivalently,

\[
\text{Pr}(Label \mid Data) = \frac{\text{Pr}(Label, Data)}{\text{Pr}(Data)}
\]

• Let \( Data = (w_1, w_2, \ldots, w_d) \).

• Let \( \nu = \text{size of vocabulary} \).

• Then we need \( \nu^d \) probability estimates!
We can use n-grams for tagging

- Replace ‘words’ with ‘tags’
- Find best maximum likelihood estimates
- Estimates calculated this way:
  - \( P(\text{noun}|\text{det}) = \frac{p(\text{det, noun})}{p(\text{det})} \) replace:
  - \( \approx \frac{\text{count(\text{det at position i-1 & noun at i})}}{\text{count(\text{det at position i-1})}} \)
  - Correction: include frequency of context word
    - \( \approx \frac{\text{count(\text{det at position i-1 & noun at I})}}{\text{count(\text{det at position i-1})}*\text{count noun at i})} \)

Find optimal path – highest \( p \), using dynamic programming algorithm, approx. linear in length of tag sequence
Example

The guy still saw her
Det NN NN NN PPO
VB VB VBD PP$
RB

Table 2 from DeRose (1988)
Det=determiner, NN=noun, VB=verb, RB=adverb, VBD=past-tense-verb, PPO=object pronoun and PP$=possessive pronoun
Find the Max likelihood estimate (MLE) path through this ‘trellis’
## Transitional probability estimates from counts

<table>
<thead>
<tr>
<th></th>
<th>DT</th>
<th>NN</th>
<th>PPO</th>
<th>PP$</th>
<th>RB</th>
<th>VB</th>
<th>VBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>0</td>
<td>186</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>NN</td>
<td>40</td>
<td>1</td>
<td>3</td>
<td>40</td>
<td>9</td>
<td>66</td>
<td>186</td>
</tr>
<tr>
<td>PPO</td>
<td>7</td>
<td>3</td>
<td>16</td>
<td>164</td>
<td>109</td>
<td>16</td>
<td>313</td>
</tr>
<tr>
<td>PP$</td>
<td>176</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>RB</td>
<td>5</td>
<td>3</td>
<td>16</td>
<td>164</td>
<td>109</td>
<td>16</td>
<td>313</td>
</tr>
<tr>
<td>VB</td>
<td>22</td>
<td>694</td>
<td>146</td>
<td>98</td>
<td>9</td>
<td>1</td>
<td>59</td>
</tr>
<tr>
<td>VBD</td>
<td>11</td>
<td>584</td>
<td>143</td>
<td>160</td>
<td>2</td>
<td>1</td>
<td>91</td>
</tr>
</tbody>
</table>
Tagging search tree (trellis)

The guy still saw her

Step 1. \( c(\text{DT-NN}) = 186 \)
\( c(\text{DT-VB}) = 1 \)
Keep both paths. (Why?)

Step 2. Pick \( \max \) to each of the tags NN, VB, RB
need keep only the \( \max \). Why?
The guy still saw her
Smoothing

- We don’t see many of the words in English (unigram)
- We don’t see the huge majority of bigrams in English
- We see only a tiny sliver of the possible trigrams
- So: most of the time, bigram model assigns $p(0)$ to bigram:
  
  $$p(\text{food}|\text{want}) = \frac{|\text{want food}|}{|\text{want}|} = 0/\text{whatever}$$

  But means event can’t happen – we aren’t warranted to conclude this... therefore, we must adjust...how?
Simplest idea: add-1 smoothing

- Add 1 to every cell of
- \( P(\text{food} \mid \text{want}) = \frac{|\text{want to}|}{|\text{want}|} = \frac{1}{2931} = .0003 \)
Laplace’s law (add 1)
Laplace’s Law (add 1)
Laplace’s Law
Initial counts – Berkeley restaurant project

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>8</td>
<td>1087</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>0</td>
<td>786</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>860</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>Chinese</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>19</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>.0023</td>
<td>.32</td>
<td>0</td>
<td>.0038</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>.0025</td>
<td>0</td>
<td>.65</td>
<td>0</td>
<td>.0049</td>
<td>.0066</td>
<td>.0049</td>
</tr>
<tr>
<td>to</td>
<td>.00092</td>
<td>0</td>
<td>.0031</td>
<td>.26</td>
<td>.00092</td>
<td>0</td>
<td>.0037</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>.0021</td>
<td>0</td>
<td>.020</td>
<td>.0021</td>
<td>.055</td>
</tr>
<tr>
<td>Chinese</td>
<td>.0094</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.56</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>.013</td>
<td>0</td>
<td>.011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>.0087</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.0022</td>
<td>0</td>
</tr>
</tbody>
</table>
### Old vs. New Table

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I</strong></td>
<td>0.0023</td>
<td>0.32</td>
<td>0</td>
<td>0.0038</td>
<td>0</td>
<td>0.0049</td>
<td>0</td>
</tr>
<tr>
<td><strong>want</strong></td>
<td>0.0025</td>
<td>0</td>
<td>0.65</td>
<td>0</td>
<td>0.0049</td>
<td>0.0066</td>
<td>0.0049</td>
</tr>
<tr>
<td><strong>to</strong></td>
<td>0.0092</td>
<td>0</td>
<td>0.0031</td>
<td>0.26</td>
<td>0.0092</td>
<td>0</td>
<td>0.0037</td>
</tr>
<tr>
<td><strong>eat</strong></td>
<td>0</td>
<td>0</td>
<td>0.0021</td>
<td>0</td>
<td>0.020</td>
<td>0.0021</td>
<td>0.055</td>
</tr>
<tr>
<td><strong>Chinese</strong></td>
<td>0.0094</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.56</td>
<td>0.0047</td>
</tr>
<tr>
<td><strong>food</strong></td>
<td>0.013</td>
<td>0</td>
<td>0.011</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>lunch</strong></td>
<td>0.0087</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0022</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>Chinese</th>
<th>food</th>
<th>lunch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I</strong></td>
<td>0.0018</td>
<td>0.22</td>
<td>0.0020</td>
<td>0.0028</td>
<td>0.0020</td>
<td>0.0020</td>
<td>0.0020</td>
</tr>
<tr>
<td><strong>want</strong></td>
<td>0.0014</td>
<td>0.0035</td>
<td>0.28</td>
<td>0.0035</td>
<td>0.0025</td>
<td>0.0032</td>
<td>0.0025</td>
</tr>
<tr>
<td><strong>to</strong></td>
<td>0.00082</td>
<td>0.0021</td>
<td>0.0023</td>
<td>0.18</td>
<td>0.0082</td>
<td>0.0021</td>
<td>0.0027</td>
</tr>
<tr>
<td><strong>eat</strong></td>
<td>0.00039</td>
<td>0.0039</td>
<td>0.0012</td>
<td>0.0039</td>
<td>0.0078</td>
<td>0.012</td>
<td>0.021</td>
</tr>
<tr>
<td><strong>Chinese</strong></td>
<td>0.0016</td>
<td>0.0055</td>
<td>0.0055</td>
<td>0.0055</td>
<td>0.0055</td>
<td>0.066</td>
<td>0.0011</td>
</tr>
<tr>
<td><strong>food</strong></td>
<td>0.0064</td>
<td>0.0032</td>
<td>0.0058</td>
<td>0.0032</td>
<td>0.0032</td>
<td>0.0032</td>
<td>0.0032</td>
</tr>
<tr>
<td><strong>lunch</strong></td>
<td>0.0024</td>
<td>0.0048</td>
<td>0.0048</td>
<td>0.0048</td>
<td>0.0048</td>
<td>0.0096</td>
<td>0.0048</td>
</tr>
</tbody>
</table>
Changes

- All non-zero probs went down
- Sometimes probs don’t change much
- Some predictable events become less predictable (P(to|want) [0.65 to 0.22])
- Other probs change by large factors (P(lunch|Chinese) [0.0047 to 0.001])
- Conclusion: generally good idea, but effect on nonzeroes not always good – blur original model – too much prob to the zeros, we want less ‘weight’ assigned to them (zero-sum game, ‘cause probs always sum to 0)
So far, then...

- n-gram models are a.k.a. Markov models/chains/processes.
- They are a model of how a sequence of observations comes into existence.
- The model is a probabilistic walk on a FSA.
- \( Pr(a/b) = \) probability of entering state \( a \), given that we’re currently in state \( b \).
How well does this work for tagging?

- 90% accuracy (for unigram) pushed up to 96%
- So what?
- How *good* is this? Evaluation!
Evaluation of systems

• The principal measures for information extraction tasks are recall and precision.

• Recall is the number of answers the system got right divided by the number of possible right answers
  • It measures how complete or comprehensive the system is in its extraction of relevant information

• Precision is the number of answers the system got right divided by the number of answers the system gave
  • It measures the system's correctness or accuracy
  • Example: there are 100 possible answers and the system gives 80 answers and gets 60 of them right, its recall is 60% and its precision is 75%.
A better measure - **Kappa**

- Takes baseline & complexity of task into account – if 99% of tags are Nouns, getting 99% correct no great shakes
- Suppose no “Gold Standard” to compare against?
- $P(A) = \text{proportion of times hypothesis agrees with standard} \ (\% \text{ correct})$
- $P(E) = \text{proportion of times hypothesis and standard would be expected to agree by chance} \ (\text{computed from some other knowledge, or actual data})$
Kappa [p. 315 J&M text]

\[ \kappa = \frac{P(A) - P(E)}{1 - P(E)} \]

- Note \( \kappa \) ranges between 0 (no agreement, except by chance; to complete agreement, 1)
- Can be used even if no ‘Gold standard’ that everyone agrees on
- \( \kappa > 0.8 \) is good
Kappa

\[ \kappa = \frac{P(A) - P(E)}{1 - P(E)} \]

- \(A\) = actual agreement; \(E\) = expected agreement
- consistency is more important than “truth”
- methods for raising consistency
  - style guides (often have useful insights into data)
  - group by task, not chronologically, etc.
  - annotator acclimatization
Statistical Tagging Methods

- Simple bigram – ok, done
- Combine bigram and unigram
Markov chain...

note: $Pr(h/h) = 0$

$$\forall y \sum_x Pr(x \mid y) = 1$$
First-order Markov (bigram) model as fsa
Markov inferences

1. Given a Markov model $M$, we can compute the (MLE) probability of a given observation sequence.

2. Given the states of a Markov model, and an observation sequence, we can estimate the state transition probabilities.

3. Given a pair of Markov models and an observation sequence, we can choose the more likely model.
From Markov models to Hidden Markov models (HMMs)

- The HMM of how an observation sequence comes into existence adds one step to a (simple/visible) Markov model.
- Instead of being observations, the states now probabilistically emit observations.
- The relationship between states and observations is, in general, many-to-many.
- We can’t be sure what sequence of states emitted a given sequence of observations.
HMM for ice-cream consumption

- Predict Boston weather given my consumption (observable) of ice-creams, 1, 2, 3.
- Underlying state is either H(ot) or C(old)
- Prob of moving between states, and also prob of eating 1, 2, 3, given in state H or C. Formally:
• Start in state $S_i$ with probability $p_i$.
• While ($t++$) do:
  • Move from $S_i$ to $S_j$ with probability $a_{ij}$ (i.e. $X_{t+1}=j$)
  • Emit observation $o_t=k$ with probability $b_{jk}$. 
Noisy channel maps well to our fsa/fst notions

- What’s p(X)?
- Ans: p(tag sequence) – i.e., some finite state automaton
- What’s p(Y|X)?
- Ans: transducer that takes tags $\rightarrow$ words
- What’s P(X,Y)?
- The joint probability of the tag sequence, given the words (well, gulp, almost... we will need one more twist – why? What is Y?)
The plan modeled as composition (x-product) of finite-state machines

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

Note \(p(x,y)\) sums to 1.
Cross-product construction for fsa’s (or fst’s)
Pulled a bit of a fast one here...

- So far, we have a plan to compute $P(X,Y)$ – but is this correct?
- $Y =$ all the words in the world
- $X =$ all the tags in the world (well, for English)
- What we get to see as input is $y \in Y$ not $Y$!
- What we want to compute is REALLY this:
  
  \[
  \text{want to recover } x \in X \text{ from } y \in Y \\
  \text{choose } x \text{ that } \text{maximizes } p(X \mid y) \text{ so...}
  \]
The real plan...

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

Find \( x \) that maximizes this quantity.
transducer: scores candidate tag seqs on their joint probability with obs words; we should pick best path

\[ p(X) \]

\[ p(Y | X) \]

\[ p(y | Y) \]

\[ = \]

\[ p(X, y) \]
This is an HMM for tagging

- Each hidden tag state produces a word in the sentence
- Each word is
  - Uncorrelated with all the other words and their tags
  - Probabilistic depending on the N previous tags only
The plan modeled as composition (product) of finite-state machines

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

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

Suppose \( y=\text{“C”} \); what is best “x”?
We need to factor in one more machine that models the actual word sequence, y.

\[ p(X) \]
\[ \ast \]
\[ p(Y \mid X) \]
\[ \ast \]
\[ p(y \mid Y) \]

restrict just to paths compatible with output “C”

find x to maximize \( p(X, y) \)
The statistical view, in short:

- We are modeling $p(\text{word seq, tag seq})$
- The tags are *hidden*, but we see the words
- What is the most likely tag sequence?
- Use a finite-state automaton, that can emit the observed words
- FSA has limited memory
- AKA this Noisy channel model is a "Hidden Markov Model"
Put the punchline before the joke

Bill directed a cortege of autos through the dunes

Recover tags
Bill directed a cortege of autos through the dunes

Find tag sequence X that maximizes probability product
Punchline – ok, where do the pr numbers come from?

The tags are not observable & they are states of some fsa. We estimate transition probabilities between states. We also have ‘emission’ pr’s from states. En tout: a **Hidden Markov Model** (HMM).
Our model uses both **bigrams** & **unigrams**: 

```
<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>Det</th>
<th>Noun</th>
<th>Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td></td>
<td>0.4</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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

This only shows the best path... how do we find it?
Submenu for probability theory – redo n-grams a bit more formally

- Define all this $p(X)$, $p(Y|X)$, $P(X,Y)$ notation
- $p$, event space, conditional probability & chain rule;
- Bayes’ Law
- (Eventually) how do we estimate all these probabilities from (limited) text? (Backoff & Smoothing)
Rush intro to probability

\[ p(\text{Paul Revere wins} \mid \text{weather’s clear}) = 0.9 \]
What’s this mean?

\[ p(\text{Paul Revere wins } | \text{ weather’s clear}) = 0.9 \]

- Past performance?
  - Revere’s won 90% of races with clear weather
- Hypothetical performance?
  - If he ran the race in many parallel universes ...
- Subjective strength of belief?
  - Would pay up to 90 cents for chance to win $1
- Output of some computable formula?
  - But then which formulas should we trust?
    \[ p(X | Y) \text{ versus } q(X | Y) \]
p is a function on event sets

\[ p(\text{win} \mid \text{clear}) \equiv \frac{p(\text{win}, \text{clear})}{p(\text{clear})} \]
p is a function on event sets

\[ p(\text{win} \mid \text{clear}) \equiv \frac{p(\text{win, clear})}{p(\text{clear})} \]

syntactic sugar  |  logical conjunction of predicates  |  predicate selecting races where weather’s clear

p measures total probability of a set of events.
Commas in \( p(x,y) \) mean conjunction – on the left...

\[
p(\text{Paul Revere wins, Valentine places, Epitaph shows } \mid \text{ weather’s clear})
\]

what happens as we add conjuncts to left of bar?

- probability can only decrease
- numerator of historical estimate likely to go to zero:
  
  # times Revere wins AND Val places... AND weather’s clear
  
  # times weather’s clear
Commas in \( p(x,y) \)...on the right

\[ p(\text{Paul Revere wins} \mid \text{weather’s clear, ground is dry, jockey getting over sprain, Epitaph also in race, Epitaph was recently bought by Gonzalez, race is on May 17, ...} ) \]

what happens as we add conjuncts to right of bar?

- probability could increase or decrease
- probability gets more relevant to our case (less bias)
- probability estimate gets less reliable (more variance)

\# times Revere wins AND weather clear AND ... it’s May 17
\# times weather clear AND ... it’s May 17
Backing off: simplifying the right-hand side...

\[ p(\text{Paul Revere wins} \mid \text{weather’s clear, ground is dry, jockey getting over sprain, Epitaph also in race, Epitaph was recently bought by Gonzalez, race is on May 17, ...}) \]

not exactly what we want but at least we can get a reasonable estimate of it!

try to *keep* the conditions that we suspect will have the most influence on whether Paul Revere wins

Recall ‘backing off’ in using just \( p(\text{rabbit} \mid \text{white}) \) instead of \( p(\text{rabbit} \mid \text{Just then a white}) \) – so this is a general method
What about simplifying the left-hand side?

\[ p(\text{Paul Revere wins, Valentine places, Epitaph shows } | \text{ weather’s clear}) \]

NOT ALLOWED!

but we can do something similar to help …

We can FACTOR this information – the so-called “Chain Rule”
Chain rule: factoring lhs

\[ p(\text{Revere}, \text{Valentine}, \text{Epitaph} | \text{weather’s clear}) = p(\text{Revere} | \text{Valentine}, \text{Epitaph}, \text{weather’s clear}) \times p(\text{Valentine} | \text{Epitaph}, \text{weather’s clear}) \times p(\text{Epitaph} | \text{weather’s clear}) \]

True because numerators cancel against denominators
Mak[es] perfect sense when read from bottom to top
Moves material to right of bar so it can be ignored

If this prob is unchanged by backoff, we say Revere was CONDITIONALLY INDEPENDENT of Valentine and Epitaph (conditioned on the weather’s being clear). Often we just ASSUME conditional independence to get the nice product above.
The plan: summary so far

automaton: \( p(\text{tag sequence}) \) 
\[ \ast \]

“Markov Model”

transducer: tags \( \rightarrow \) words

“Unigram Replacement”
\[ \ast \]

automaton: the observed words

“straight line”
\[ = \]

transducer: scores candidate tag seqs on their joint probability with obs words; pick best path
First-order Markov (bigram) model as fsa
Add in transition probs - sum to 1
Same as bigram

\[ P(\text{Noun}|\text{Det}) = 0.7 \equiv \]
Add in start & etc.
Markov Model

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

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

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

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

\[ \text{Start Det Adj Adj Noun Stop} = 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2 \]
Add ‘output tags’ (transducer) $p(\text{tag seq})$

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

\[
\phantom{\text{Start}} \xrightarrow{\text{Det Adj Adj Noun Stop}} 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2
\]
Tag bigram picture

$p$(tag seq)

$\text{Start} \rightarrow \text{Det} \rightarrow \text{Adj} \rightarrow \text{Adj} \rightarrow \text{Noun} \rightarrow \text{Stop}$

$= 0.8 \times 0.3 \times 0.4 \times 0.5 \times 0.2$
Our plan

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

"Markov Model"

* 

transducer: tags \( \rightarrow \) words

"Unigram Replacement"

* 

automaton: the observed words

"straight line"

= 

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

\[ p(X) \] 

* 

\[ p(Y \mid X) \] 

* 

\[ p(y \mid Y) \] 

= 

\[ p(X, y) \]
Cartoon form again

transducer: scores candidate tag seqs on their joint probability with obs words; we should pick best path
Next up: unigram replacement model

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

\[ \text{Det: the/0.4} \]
\[ \text{Det: a/0.6} \]
\[ \text{Noun: autos/0.001} \]
\[ \text{Noun: Bill/0.002} \]
\[ \text{Adj: cool/0.003} \]
\[ \text{Adj: directed/0.0005} \]
\[ \text{Adj: cortege/0.000001} \]
\[ \text{Noun: cortege/0.000001} \]

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
p(word seq, tag seq) = p(tag seq) * p(word seq | 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:

Start Det Adj Adj Noun Stop = 0.32 \times 0.0009 \times 0.00020 \ldots
the cool directed autos

\[ p(\text{word seq, tag seq}) = p(\text{tag seq}) \times p(\text{word seq} \mid \text{tag seq}) \]
But...how do we find this ‘best’ path???
All paths together form ‘trellis’

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

The best path:

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

the cool directed autos

WHY?
Cross-product construction forms trellis

So all paths here must have 5 words on output side

All paths here are 5 words

So all paths here must have 5 words on output side
Trellis isn’t complete

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

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

The best path:

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

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

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
And missing some states...

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

Lattice is missing some states; why?

The best path:

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

the cool directed autos
Finding the 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 algorithm
- Faster if some arcs/states are absent
Method: Viterbi algorithm

- For **each** path reaching state $s$ at step (word) $t$, we compute a path probability. We call the **max** of these $\text{viterbi}(s,t)$
- **[Base step]** Compute $\text{viterbi}(0,0)=1$
- **[Induction step]** Compute $\text{viterbi}(s',t+1)$, assuming we know $\text{viterbi}(s,t)$ for all $s$
Viterbi recursion

\[
\text{path-prob}(s'|s,t) = v^t(s,t) \times a(s,s')
\]

probability of path to s' through s
max path score * transition p for state s at time t

\[
v^t(s',t+1) = \max_{s \in \text{STATES}} \text{path-prob}(s'|s,t)
\]
This is *almost* correct... but again, we need to factor in the unigram prob of a state $s'$ given an observed surface word $w$.

So the correct formula for the path prob is:

$$\text{path-prob}(s'|s,t) = \text{viterbi}(s,t) \times a[s,s'] \times b_{s'}(o_t)$$

**bigram  unigram**
function VITERBI(observations of len $T$, state-graph) returns best-path

$num-states \leftarrow$ NUM-OF-STATES(state-graph)
Create a path probability matrix $viterbi[num-states+2, T+2]$
$viterbi[0,0] \leftarrow 1.0$
for each time step $t$ from 0 to $T$ do
    for each state $s$ from 0 to num-states do
        for each transition $s'$ from $s$ specified by state-graph
            $new-score \leftarrow viterbi[s, t] \times a[s,s'] \times b_{s'}(o_t)$
            if $((viterbi[s', t+1] = 0) \lor (new-score > viterbi[s', t+1]))$
                then
                    $viterbi[s', t+1] \leftarrow new-score$
                    $back-pointer[s', t+1] \leftarrow s$
        Backtrace from highest probability state in the final column of $viterbi[]$ and return path
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”:

$$\text{Start} \rightarrow \text{PN} \rightarrow \text{Verb} \rightarrow \text{Det} \rightarrow \text{Noun} \rightarrow \text{Prep} \rightarrow \text{Noun} \rightarrow \text{Prep}$$

- Find $X$ that maximizes probability product
Evaluation of systems...redux

- The principal measures for information extraction tasks are recall and precision.
- **Recall** is the number of answers the system got right divided by the number of possible right answers
  - It measures how complete or comprehensive the system is in its extraction of relevant information
- **Precision** is the number of answers the system got right divided by the number of answers the system gave
  - It measures the system's correctness or accuracy
  - Example: there are 100 possible answers and the system gives 80 answers and gets 60 of them right, its recall is 60% and its precision is 75%.
A better measure - **Kappa**

- Takes baseline & complexity of task into account – if 99% of tags are Nouns, getting 99% correct no great shakes
- Suppose no “Gold Standard” to compare against?
- \( P(A) = \) proportion of times hypothesis *agrees* with standard (% correct)
- \( P(E) = \) proportion of times hypothesis and standard would be *expected* to agree by chance (computed from some other knowledge, or actual data)
Kappa [p. 315 J&M text]

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

- Note $K$ ranges between 0 (no agreement, except by chance; to complete agreement, 1)
- Can be used even if no ‘Gold standard’ that everyone agrees on
- $K > 0.8$ is good
Two approaches

1. Noisy Channel Model (statistical) – what’s that?? (we will have to learn some statistics)

2. Deterministic baseline tagger composed with a cascade of fixup transducers

These two approaches will be the guts of Lab 2 (lots of others: decision trees, ...)
Fixup approach: Brill tagging
Another FST Paradigm: Successive Fixups

- Like successive markups but *alter*
- Morphology
- Phonology
- Part-of-speech tagging
- ...

Diagram:
- Initial annotation
- Fixup 1
- Fixup 2
- Fixup 3

Input: Initial annotation
Output: Fixup 3
Transformation-Based Tagging
(Brill 1995)
Transformations Learned

<table>
<thead>
<tr>
<th>#</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td>Previous tag is <em>TO</em></td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td>One of the previous three tags is <em>MD</em></td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td>One of the previous two tags is <em>MD</em></td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td>One of the previous two tags is <em>DT</em></td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous three tags is <em>VBZ</em></td>
</tr>
<tr>
<td>6</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is <em>PRP</em></td>
</tr>
<tr>
<td>7</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is <em>NNP</em></td>
</tr>
<tr>
<td>8</td>
<td>VBD</td>
<td>VBN</td>
<td>Previous tag is <em>VBD</em></td>
</tr>
<tr>
<td>9</td>
<td>VBP</td>
<td>VB</td>
<td>Previous tag is <em>TO</em></td>
</tr>
<tr>
<td>10</td>
<td>POS</td>
<td>VBZ</td>
<td>Previous tag is <em>PRP</em></td>
</tr>
<tr>
<td>11</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is <em>NNS</em></td>
</tr>
<tr>
<td>12</td>
<td>VBD</td>
<td>VBN</td>
<td>One of previous three tags is <em>VBP</em></td>
</tr>
<tr>
<td>13</td>
<td>IN</td>
<td>WDT</td>
<td>One of next two tags is <em>VB</em></td>
</tr>
<tr>
<td>14</td>
<td>VBD</td>
<td>VBN</td>
<td>One of previous two tags is <em>VB</em></td>
</tr>
<tr>
<td>15</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is <em>PRP</em></td>
</tr>
<tr>
<td>16</td>
<td>IN</td>
<td>WDT</td>
<td>Next tag is <em>VBZ</em></td>
</tr>
<tr>
<td>17</td>
<td>IN</td>
<td>DT</td>
<td>Next tag is <em>NN</em></td>
</tr>
<tr>
<td>18</td>
<td>JJ</td>
<td>NNP</td>
<td>Next tag is <em>NNP</em></td>
</tr>
<tr>
<td>19</td>
<td>IN</td>
<td>WDT</td>
<td>Next tag is <em>VBD</em></td>
</tr>
<tr>
<td>20</td>
<td>JJR</td>
<td>RBR</td>
<td>Next tag is <em>JJ</em></td>
</tr>
</tbody>
</table>

BaselineTag*

NN @→ VB // TO _
VBP @→ VB // ... _

etc.

Compose this cascade of FSTs.

Get a big FST that does the initial tagging and the sequence of fixups “all at once.”
# Initial Tagging of OOV Words

<table>
<thead>
<tr>
<th>#</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>NNS</td>
<td>Has suffix -s</td>
</tr>
<tr>
<td>2</td>
<td>NN</td>
<td>CD</td>
<td>Has character .</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>JJ</td>
<td>Has character -</td>
</tr>
<tr>
<td>4</td>
<td>NN</td>
<td>VBN</td>
<td>Has suffix -ed</td>
</tr>
<tr>
<td>5</td>
<td>NN</td>
<td>VBG</td>
<td>Has suffix -ing</td>
</tr>
<tr>
<td>6</td>
<td>??</td>
<td>RB</td>
<td>Has suffix -ly</td>
</tr>
<tr>
<td>7</td>
<td>??</td>
<td>JJ</td>
<td>Adding suffix -ly results in a word.</td>
</tr>
<tr>
<td>8</td>
<td>NN</td>
<td>CD</td>
<td>The word $ can appear to the left.</td>
</tr>
<tr>
<td>9</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -al</td>
</tr>
<tr>
<td>10</td>
<td>NN</td>
<td>VB</td>
<td>The word would can appear to the left.</td>
</tr>
<tr>
<td>11</td>
<td>NN</td>
<td>CD</td>
<td>Has character 0</td>
</tr>
<tr>
<td>12</td>
<td>NN</td>
<td>JJ</td>
<td>The word be can appear to the left.</td>
</tr>
<tr>
<td>13</td>
<td>NNS</td>
<td>JJ</td>
<td>Has suffix -us</td>
</tr>
<tr>
<td>14</td>
<td>NNS</td>
<td>VBZ</td>
<td>The word it can appear to the left.</td>
</tr>
<tr>
<td>15</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ble</td>
</tr>
<tr>
<td>16</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ic</td>
</tr>
<tr>
<td>17</td>
<td>NN</td>
<td>CD</td>
<td>Has character 1</td>
</tr>
<tr>
<td>18</td>
<td>NNS</td>
<td>NN</td>
<td>Has suffix -ss</td>
</tr>
<tr>
<td>19</td>
<td>??</td>
<td>JJ</td>
<td>Deleting the prefix un- results in a word</td>
</tr>
<tr>
<td>20</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ive</td>
</tr>
</tbody>
</table>
Laboratory 2

• Goals:
  1. Use both HMM and Brill taggers
  2. Find errors that both make
  3. Compare performance – use of kappa & ‘confusion matrix’
  4. All the slings & arrows of corpora – use Wall Street Journal excerpts