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

Instructor: Robert C. Berwick berwick@ai.mit.edu

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) * $\mathbf{p}(\mathbf{Y} \mid \mathbf{X})$ * **p(y | Y)** Find x that maximizes this quantity **p(X, y)**



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 <u>www.turnitin.com</u>
- 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"

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?

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.

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.)

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?

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.

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.

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?

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?

Knowldege-based (rule-based) vs. Statistically-based systems

A picture: the statistical, noisy channel view



Language models, probability & info

- Given a string w, a language model gives us the probability of the string P(w), e.g.,
 - P(the big dog) > (dog big the) > (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 <u>acoustic model</u>]

 From Bayes' law, we have,
max P(y|x) = max P(x|y) • P(y) = max Pr acoustic model x lang model

(hold P(x) fixed, i.e., P(x|y) • 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! human **q**? P(a|q) = pr human answers q with a =a! Interrogator P(q,a)/P(q)P(b|q) = pr computer answers q with b =P(q,b)/P(q)If P(q,a) = P(q,b) then computer **b!** successfully imitates human computer

Some applications...

- Spelling correction, grammar checking ...
- Better search engines
- Information extraction
- Language identification (English vs. Polish)
- Psychotherapy; Harlequin romances; etc.
- And: plagiarism detection <u>www.turnitin.com</u>
- 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)

Example – "Message understanding" (MUC)

ST1-MUC3-0011

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

1. MESSAGE: TEMPLATE 2. INCIDENT: DATE **3. INCIDENT: LOCATION** 4. INCIDENT: TYPE 5. INCIDENT: STAGE OF EXECUTION 6. INCIDENT: INSTRUMENT ID 7. INCIDENT: INSTRUMENT TYPE 8. PERP: INCIDENT CATEGORY 9. PERP: INDIVIDUAL ID **10. PERP: ORGANIZATION ID 11. PERP: ORGANIZATION CONFIDENCE REPORTED AS FACT:** 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 19. HUM TGT: DESCRIPTION**

20. HUM TGT: TYPE

0. MESSAGE: ID

TST1-MUC3-0011

1 18 MAY 90 UNITED STATES: WASHINGTON D.C. (CITY) ATTACK ACCOMPLISHED

STATE-SPONSORED VIOLENCE

"CHILEAN GOVERNMENT"

"CHILEAN GOVERNMENT"

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

MUC 5 - business

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\$2000000 |
| 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
Typical finite-state system – Fastus (SRI)

From Raw Text to the Analyst's File



Name recognition: a fsa task

<u>Bridgestone Sports Co.</u> c said <u>Friday</u> it had set up a joint venture in <u>Taiwan</u> with a local concern and a Japanese trading house to produce golf clubs to be shipped to <u>Japan</u>.

Shallow parsing or 'chunking' – Lab 3

Bridgestone Sports Co. said Friday it has set up a joint venture Company-Name VG NG VG NG VG NG Complex VG

in Taiwan with a local concern and a Japanese trading house NG Loc P <u>NG Conj NG</u> Complex NG

to produce golf clubs to be shipped to Japan. VG(Inf) NG VG(Inf,Pass) P Loc

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."

| Entities: "Bridgestone Sports Co." "a local concern" "a Japanese trading house" JV Company: Capitalization: | JV Company: "Bridgestone Sports Taiwan Co Capitalization: 2000000 TW |
|---|--|
|---|--|

What about part of speech tagging here?

- Advantages
 - Ambiguity can be <u>potentially</u> 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
 - $P(S_j | S_{j-1}, W_j)$
 - 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 loglinear 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 multiway 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

| | Tag | Description | Example | Tag | Description | Example |
|----------|------|-----------------------|-----------------|------|-----------------------|------------------|
| | CC | Coordin. Conjunction | and, but, or | SYM | Symbol | +,%, & |
| | CD | Cardinal number | one, two, three | ТО | "to" | to |
| | DT | Determiner | a, the | UH | Interjection | ah, oops |
| | EX | Existential 'there' | there | VB | Verb, base form | eat |
| | FW | Foreign word | mea culpa | VBD | Verb, past tense | ate |
| | IN | Preposition/sub-conj | of, in, by | VBG | Verb, gerund | eating |
| | JJ | Adjective | yellow | VBN | Verb, past participle | eaten |
| | JJR | Adj., comparative | bigger | VBP | Verb, non-3sg pres | eat |
| | JJS | Adj., superlative | wildest | VBZ | Verb, 3sg pres | eats |
| l toyt | LS | List item marker | 1, 2, One | WDT | Wh-determiner | which, that |
| J. LEXI, | MD | Modal | can, should | WP | Wh-pronoun | what, who |
| p. 297 | NN | Noun, sing. or mass | llama | WP\$ | Possessive wh- | whose |
| | NNS | Noun, plural | llamas | WRB | Wh-adverb | how, where |
| FIY 0.0 | NNP | Proper noun, singular | IBM | \$ | Dollar sign | \$ |
| 1M words | NNPS | Proper noun, plural | Carolinas | # | Pound sign | # |
| KOK tog | PDT | Predeterminer | all, both | " | Left quote | (' or '') |
| our lay | POS | Possessive ending | 'S | " | Right quote | (' or '') |
| counts | PP | Personal pronoun | I, you, he | (| Left parenthesis | $([, (, \{, <)$ |
| | PP\$ | Possessive pronoun | your, one's |) | Right parenthesis | $(],), \}, >)$ |
| | RB | Adverb | quickly, never | , | Comma | , |
| | RBR | Adverb, comparative | faster | • | Sentence-final punc | (.!?) |
| | RBS | Adverb, superlative | fastest | : | Mid-sentence punc | (:;) |
| | RP | Particle | up, off | | | |



correct tags Det Noun Prep Noun Prep Det Noun PN Verb *directed* cortege of autos through the dunes а Bill Noun Prep Noun Prep Det Noun **D**et erb Noun Verb Adj some possible tags for Prep each word (maybe more) ?

correct tags PN Verb Det Noun Prep Noun Prep Det Noun Bill directed a cortege of autos through the dunes N Adj Det Noun Prep Noun Prep Det Noun Verb Verb Noun Prep Noun Prep Det Noun Adj some possible tags for Prep each word (maybe more) ave

correct tags Det Noun Prep Noun Prep Det Noun PN Verb *directed* cortege of autos through the dunes а Bill Noun Prep Noun Prep Det Noun **D**et erb Noun Verb Adj some possible tags for Prep each word (maybe more) ?

Finite-state approaches

Noishy Chunnel Muddle (statistical)



want to recover X from Y

Noisy channel – and prob intro



choose sequence of tags X that *maximizes* p(X | Y) [oops... this isn't *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(tag|word)
- Task of predicting the next word
- Woody Allen:
 - "I have a gub"

In general: predict the Nth word (tag) from the preceding N-1 word (tags) aka <u>N-gram</u>

Homer Simpson: just use the current word (don't look at context) = <u>unigram (1-gram)</u>

Where do these probability info estimates come from?

- Use tagged <u>corpus</u> e.g. "Brown corpus" 1M words (fewer token *instances*); many others – Celex 16M words
- Use <u>counts</u> (relative frequencies) as estimates for probabilities (various issues w/ this, these so-called <u>Maximum-Likelihood estimates</u> – 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

 Iong distance call |/ | distance call | ≈ | 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(tag| all previous tags) Instead of

> p(rabbit|Just then the white...) we use: P(rabbit|white)

 This is a <u>Markov assumption</u> 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 Capiduatent 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 Al1. 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 unismelight fammedied: and nople,

4-gram

[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 convuning clann com to one language; all Lah, which for the greath othey die. -

5-gram

[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

3-word-gram

[Gen 4:25] And Adam gave naines to ail feudal, patriarchal, idyllic relations. It bas but -established new classes, new conditions of oppression, new forme 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 29,066² > 844 million. 1 million word training set doesn't cut it – only 300,000 difft 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

| n | Number of bins |
|--------------|------------------------|
| 2 (bigrams) | 400,000,000 |
| 3 (trigrams) | 8,000,000,000,000 |
| 4 (4-grams) | 1.6 x 10 ¹⁷ |

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 _____"*



Maximum likelihood estimate



Actual probability distribution



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(Label | 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:

 $Pr(Label \mid Data) = \frac{count(Label, Data)}{count(Data)}$

 To predict a Label for a new bunch of Data, we just look up Pr(Label | Data) for all possible Labels and output the argmax.

Practical limitations

• Equivalently,

$$Pr(Label | Data) = \frac{Pr(Label, Data)}{Pr(Data)}$$

• Let
$$Data = (w_1, w_2, ..., w_d)$$
.

- Let v = size of vocabulary.
- Then we need v^d probability estimates!

We can use n-grams for tagging

- Replace 'words' with 'tags'
- Find best maximum likelihood estimates
- Estimates calculated this way:
 - P(noun|det) = p(det, noun)/p(det) replace:
 - ≈ <u>count(det at position i-1 & noun at i)</u> count(det at position i-1)
 - Correction: include frequency of context word
 ≈ count(det at position i-1 & noun at I) count(det at position i-1)*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

| | DT | NN | PPO | PP\$ | RB | VB | VBD |
|------|-----|-----|-----|------|-----|----|-----|
| DT | 0 | 186 | 0 | 8 | 1 | 8 | 9 |
| NN | 40 | 1 | 3 | 40 | 9 | 66 | 186 |
| PPO | 7 | 3 | 16 | 164 | 109 | 16 | 313 |
| PP\$ | 176 | 0 | 0 | 5 | 1 | 1 | 2 |
| RB | 5 | 3 | 16 | 164 | 109 | 16 | 313 |
| VB | 22 | 694 | 146 | 98 | 9 | 1 | 59 |
| VBD | 11 | 584 | 143 | 160 | 2 | 1 | 91 |

Tagging search tree (trellis)



Step 1. c(DT-NN) = 186
 c(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?

Trellis search



Smoothing

- We don't see many of the words in English (uniqram)
- 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(food|want) = |want food| /|want| = 0/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(food | want) = |want to| ÷ |want| = 1 ÷
 2931 = .0003

Laplace's law (add 1)



Laplace's Law (add 1)



Laplace's Law



Initial counts – Berkeley restaurant project

| | Ι | want | to | eat | Chinese | food | lunch |
|---------|----|------|-----|-----|---------|------|-------|
| Ι | 8 | 1087 | 0 | 13 | 0 | 0 | 0 |
| want | 3 | 0 | 786 | 0 | 6 | 8 | 6 |
| to | 3 | 0 | 10 | 860 | 3 | 0 | 12 |
| eat | 0 | 0 | 2 | 0 | 19 | 2 | 52 |
| Chinese | 2 | 0 | 0 | 0 | 0 | 120 | 1 |
| food | 19 | 0 | 17 | 0 | 0 | 0 | 0 |
| lunch | 4 | 0 | 0 | 0 | 0 | 1 | 0 |

| | Ι | want | to | eat | Chinese | food | lunch |
|---------|--------|------|-------|-------|---------|-------|-------|
| Ι | .0023 | .32 | 0 | .0038 | 0 | 0 | 0 |
| want | .0025 | 0 | .65 | 0 | .0049 | .0066 | .0049 |
| to | .00092 | 0 | .0031 | .26 | .00092 | 0 | .0037 |
| eat | 0 | 0 | .0021 | 0 | .020 | .0021 | .055 |
| Chinese | .0094 | 0 | 0 | 0 | 0 | .56 | .0047 |
| food | .013 | 0 | .011 | 0 | 0 | 0 | 0 |
| lunch | .0087 | 0 | 0 | 0 | 0 | .0022 | 0 |

Old vs.New table

| | Ι | want | to | eat | Chinese | food | lunch |
|---------|--------|------|-------|-------|---------|-------|-------|
| Ι | .0023 | .32 | 0 | .0038 | 0 | 0 | 0 |
| want | .0025 | 0 | .65 | 0 | .0049 | .0066 | .0049 |
| to | .00092 | 0 | .0031 | .26 | .00092 | 0 | .0037 |
| eat | 0 | 0 | .0021 | 0 | .020 | .0021 | .055 |
| Chinese | .0094 | 0 | 0 | 0 | 0 | .56 | .0047 |
| food | .013 | 0 | .011 | 0 | 0 | 0 | 0 |
| lunch | .0087 | 0 | 0 | 0 | 0 | .0022 | 0 |

| | Ι | want | to | eat | Chinese | food | lunch |
|---------|--------|--------|--------|--------|---------|--------|--------|
| Ι | .0018 | .22 | .00020 | .0028 | .00020 | .00020 | .00020 |
| want | .0014 | .00035 | .28 | .00035 | .0025 | .0032 | .0025 |
| to | .00082 | .00021 | .0023 | .18 | .00082 | .00021 | .0027 |
| eat | .00039 | .00039 | .0012 | .00039 | .0078 | .0012 | .021 |
| Chinese | .0016 | .00055 | .00055 | .00055 | .00055 | .066 | .0011 |
| food | .0064 | .00032 | .0058 | .00032 | .00032 | .00032 | .00032 |
| lunch | .0024 | .00048 | .00048 | .00048 | .00048 | .00096 | .00048 |

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 <u>recall</u> and <u>precision</u>.
- <u>Recall</u> 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
- <u>Precision</u> 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]

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

- 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

Kappa
$$\boldsymbol{k} = \frac{P(A) - P(E)}{1 - P(E)}$$

- 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...



First-order Markov (bigram) model as fsa



Markov inferences

- 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.
- 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 <u>being</u> observations, the states now probabilistically <u>emit</u> 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_{ik} .

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 (xproduct) of finite-state machines



Note p(x,y) sums to 1.



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 yÎ Y not Y!
- What we want to compute is REALLY this: want to recover xÎ X from yÎ Y choose x that maximizes p(X | y) SO...

The real plan...

p(X) * $\mathbf{p}(\mathbf{Y} \mid \mathbf{X})$ * **p(y | Y)** Find x that maximizes this quantity **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



Note p(x,y) sums to 1. Suppose y="C"; what is best "x"? We need to factor in one more machine that models the actual word sequence, y



The statistical view, in short:

- We are modeling p(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 "<u>Hidden</u> <u>Markov Model</u>" -

Put the punchline before the joke

Bill directed a cortege of autos through the dunes



Punchline – recovering (words, tags)



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 <u>Hidden Markov Model</u> (HMM)





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(Paul Revere wins | weather's clear) = 0.9

What's this mean?

p(Paul Revere wins | 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) versus q(X | Y)

p is a function on event sets

$p(win | clear) \equiv p(win, clear) / p(clear)$



p is a function on event sets

 $p(win | clear) \equiv p(win, clear) / p(clear)$

syntactic sugar

logical conjunctionpredicate selectingof predicatesraces whereweather's clear



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

p(Paul Revere wins, Valentine places, Epitaph shows | 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: <u># times Revere wins AND Val places... AND weather's clear</u> # times weather's clear

Commas in p(x,y)...on the right

p(Paul Revere wins | 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 <u>right</u> 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 righthand side...

- p(Paul Revere wins | 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(rabbit|white) instead of p(rabbit|Just then a white) – so this is a general method

What about simplifying the lefthand side?

p(Paul Revere wins, Valentine places, Epitaph shows | 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(Revere, Valentine, Epitaph | weather's clear) RVEW/W
= p(Revere | Valentine, Epitaph, weather's clear) = RVEW/VEW
* p(Valentine | Epitaph, weather's clear) * VEW/EW
* p(Epitaph | weather's clear) * EW/W
True because numerators cancel against denominators
Makes 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.



First-order Markov (bigram) model as fsa



Add in transition probs - sum to 1



Same as bigram

$P(Noun|Det)=0.7 \equiv$



Add in start & etc.





Markov model as fsa p(tag seq)



Start Det Adj Adj Noun Stop = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Add 'output tags' (transducer) p(tag seq)



Tag bigram picture p(tag seq)



Start Det Adj Adj Noun Stop = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Our plan



Cartoon form again



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

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








. . .



Compose





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



Observed words as straight-line fsa

word seq



Compose with Othe Cool Odirected autos

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





The best path:Start Det AdjAdjNoun Stop = 0.32 * 0.0009*0.00020...the cool directed autos

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



But...how do we find this 'best' path???

All paths together form 'trellis'



The best path: Start Det Adj Adj Noun Stop = 0.32 * 0.0009 ... the cool directed autos

Cross-product construction forms trellis



Trellis isn't complete p(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(word seq, tag seq)

Lattice is missing some other arcs; why?



The best path:Start Det AdjAdjNoun Stop= 0.32 * 0.0009 ...the cool directed autos

And missing some states... p(word seq, tag seq)

Lattice is missing some states; why?



The best path:Start Det AdjAdjNoun Stop= 0.32 * 0.0009 ...the cool directed autos

Finding the best path from start to stop

Ad

e 0.2

Use dynamic programming

Jet: the

- What is best path from Start to <u>each</u> 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 <u>each</u> path reaching state s at step (word)
 t, we compute a path probability. We call the max of these <u>viterbi(s,t)</u>
- [Base step] Compute viterbi(0,0)=1
- [Induction step] Compute viterbi(s',t+1), assuming we know viterbi(s,t) for all s

Viterbi recursion

path-prob(s'|s,t) =viterbi(s,t) *
$$a[s,s']$$
probability of path tomax path score *transition ps' through sfor state s at time t $s \rightarrow s'$

viterbi(s',t+1) = max_{s in STATES} path-prob(s' | s,t)

Method...

- 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 probis:

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

bigram unigram

Or as in your text...p. 179

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*] * *a*[*s*,*s'*] * *b*_{*s'*}(*o*_{*t*})

if ((*viterbi*[s',t+1] = 0) || (*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(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":



Find X that maximizes probability product

Evaluation of systems...redux

- The principal measures for information extraction tasks are recall and precision.
- <u>Recall</u> 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
- <u>Precision</u> 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]

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

- 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

 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 the guts of Lab 2 (lots of others: decision trees, ...)

Fixup approach: Brill tagging

Another FST Paradigm: Successive Fixups

• Like successive markups but alter

FIXI

output

- Morphology
- Phonology

input

Part-of-speech tagging

Frital ann tainn

Transformation-Based Tagging figure from Brill's thesis (Brill 1995)



figure from Brill's thesis

Transformations Learned

| | Chan | ge Tag | |
|----|------|--------|---|
| # | From | То | Condition |
| 1 | NN | VB | Previous tag is TO |
| 2 | VBP | VB | One of the previous three tags is MD |
| 3 | NN | VB | One of the previous two tags is MD |
| 4 | VB | NN | One of the previous two tags is DT |
| 5 | VBD | VBN | One of the previous three tags is VBZ |
| 6 | VBN | VBD | Previous tag is <i>PRP</i> |
| 7 | VBN | VBD | Previous tag is NNP |
| 8 | VBD | VBN | Previous tag is VBD |
| 9 | VBP | VB | Previous tag is TO |
| 10 | POS | VBZ | Previous tag is <i>PRP</i> |
| 11 | VB | VBP | Previous tag is NNS |
| 12 | VBD | VBN | One of previous three tags is VBP |
| 13 | IN | WDT | One of next two tags is VB |
| 14 | VBD | VBN | One of previous two tags is VB |
| 15 | VB | VBP | Previous tag is <i>PRP</i> |
| 16 | IN | WDT | Next tag is VBZ |
| 17 | IN | DT | Next tag is NN |
| 18 | 11 | NNP | Next tag is NNP |
| 19 | IN | WDT | Next tag is VBD |
| 20 | JJR | RBR | Next tag is JJ |

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."

figure from Brill's thesis

Initial Tagging of OOV Words

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

Laboratory 2

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