HMM
and
Part of Speech Tagging

Adam Meyers

New York University

Computational Linguistics
Lecture 4
2011-2012

Outline

- Parts of Speech Tagsets
- Rule-based POS Tagging
- HMM POS Tagging
- Transformation-based POS Tagging

Part of Speech Tags Standards

- There is no standard set of parts of speech that is used by all researchers for all languages.
- The most commonly used English tagset is that of the Penn Treebank at the University of Pennsylvania:
  - pdf of POS annotation guidelines at class website
    - http://cs.nyu.edu/courses/spring12/CSCI-GA.2590-001/ptb_tagguide.pdf
- To map several POS tagsets to each other, see Table 1 in:
  - http://nlp.cs.nyu.edu/wiki/corpuswg/AnnotationCompatibilityReport
- POS tagsets:
  - Assume Particular Tokenizations, e.g., ***Mary's*** → ***Mary*** + ***'s***
  - Distinguish inflectional morphology: e.g., *eat*/**VB**, *eats*/**VBZ**, *ate*/**VBD**
  - Are exceptionless – there are tags for all possible tokens

The Penn Treebank II POS tagset

- Verbs: VB, VBP, VBZ, VBD, VBG, VBN
  - base, present-non-3rd, present-3rd, past, -ing, -en
- Nouns: NNP, NNPS, NN, NNS
  - proper/common, singular/plural (singular includes mass + generic)
- Adjectives: JJ, JJR, JJS (base, comparative, superlative)
- Adverbs: RB, RBR, RBS, RP (base, comparative, superlative, particle)
- Pronouns: PRP, PP$ (personal, possessive)
- Interogatives: WP, WP$, WDT, WRB (compare to: PRP, PP$, DT, RB)
- Other Closed Class: CC, CD, DT, PDT, IN, MD
- Punctuation: **# $. , : ( ) “ ” '' ' `**
- Weird Cases: FW(***deja vu***), SYM (**@**), LS (***1, 2, a, b***), TO (***to***), POS(***'s, '***), UH (***no, OK, well***), EX (***it/there***)
- Newer tags: HYPH, PU

Part of Speech Tagging

- POS taggers assign 1 POS tag to each input token
  - *The*/DT *silly*/**JJ** *man*/**NN** *is*/**VBP** *a*/**DT** *professor*/**NN** ./**PU**
- Different ways of breaking down POS tagging:
  - Use separate "tokenizer", program that divides string into list of tokens – POS tagger processes output
  - Incorporate tokenizer into POS tagger
- Different ways of breaking down parsing:
  - Use separate POS tagger – output of tagger is input to parser
  - Assign POS tags as part of parsing (assumed previously)
- Accurate POS tagging is "easier" than accurate parsing
  - POS tags may be sufficient information for some tasks

Some Tokenization Rules for English

- 1) Divide at spaces and hyphens.
- 2) Divide before punctuation that is followed by: a space or the end of the line
  - Define punctuation as any non-letter/non-number:
    - ***`!@#$%^&*()-_+={[}]\|:;'”<,>.?/***
  - A list of punctuation followed by a space or end of line should be broken up into a list of characters:
    - ***...and he left.”) → and he left . ”)***
- 3) Break off the following as separate tokens when followed by a space or end of line:
  - ***'s***, ***n't***, ***'d***, ***'ve***, ***'m***, ***'ll***, ***'re***, … (a short list)
- 4) Abbreviations are exceptions to rule 1:
  - Period after abbreviations should not be separate
    - Most cases covered by list of 100 items (or if sentence end is known)
  - Final periods are not duplicated after abbreviations (consistency issues)

Rule-based POS Tagger

- Method
  - Assign lists of potential POS tags to each word based on dictionary
  - Manual rules for Out of Vocabulary (OOV) words
    - Ex: Non-initial capital → NNP; ends in S → VBZ|NNS; default → NN|JJ; etc.
  - Apply hand-written constraints until each word has only one possible POS
- Sample Constraints:
  - 1) DT cannot immediately precede a verb
  - 2) No verb can immediately precede a tensed verb (VBZ,VBP,VBD)
- Example:
  - The/DT book/{NN|VB|VBP} is/VBZ on/IN the/DT table{NN|VB|VBP}
  - The/DT book/NN is/VBZ on/IN the/DT table/NN
    - DT cannot precede VB or VBP
    - VBZ cannot be preceded by VB or VBP

Probabilistic Models of POS tagging

- For tokens w_1, ..., w_n, find the most probable corresponding sequence of possible tags t_1, ..., t_n
  - We assume that *probable* means something like "most frequently observed in some manually tagged corpus of words".
- Penn Treebank II (a common training corpus)
  - 1 million words from the Wall Street Journal
  - Tagged for POS (and other attributes)
- The specific sequence (sentence) is not in the training corpus
  - Therefore the actual "probability" is 0
  - Common practice: estimate probability given assumptions, e.g.,
    - Choose the most frequent possible sequence independently of how frequently each tag is assigned to each token

Go to Ralph's Viterbi Demo for ***Fish Sleep***

Probabilistic Assumptions of HMM Tagging

- $\hat{t} = \underset{t_1^n}{argmax}\, P(t_1^n | w_1^n)$
  - Choose the tag sequence of length n that is most probable given the input token sequence
- Bayes Rule:
  - $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$
  - Way to derive the probability of x given y when you know the probability of y given x
- Applying Bayes Rule to Tag Probability
  - $\hat{t} = \underset{t_1^n}{argmax}\, \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$

Simplifying Assumptions for HMMs

- Simplification: Drop the denominator
  - Denominator is same for all the tag sequences
  - $\hat{t} = \underset{t_1^n}{argmax}\, P(w_1^n|t_1^n)\,P(t_1^n)$
  - For each tag sequence calculate the product of:
    - The probability of the word sequence given the tag sequence (**likelihood**)
    - The probability of the tag sequence (**prior probability**)
  - Still too hard
- 2 simplifying assumptions make it possible to estimate the probability of tag sequences given word sequences:
  - 1) If the probability of a word is only dependent on its own POS tag,
    - $P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$
  - 2) If the probability of a POS tag is only dependent on the previous POS tag,
    - $P(t^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$
- The result of these assumptions: $\hat{t} \approx \underset{t_1^n}{argmax} \prod_{i=1}^{n} P(w_i|t_i)\,P(t_i|t_{i-1})$
- HMM taggers are fast and achieve precision/recall scores of about 93-95%

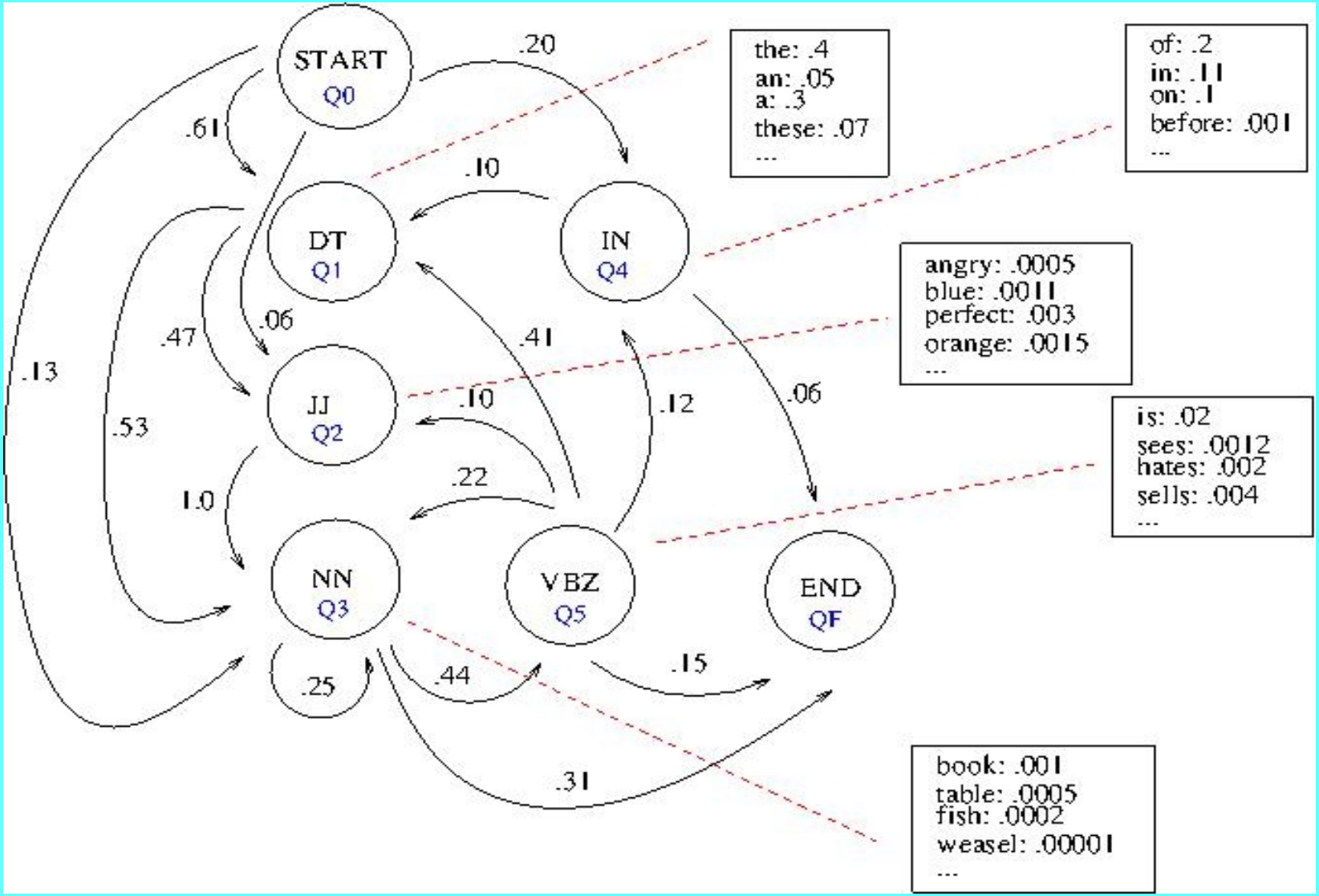
Estimating Probability of $\hat{t}$

- We assume that: $\hat{t} \approx \underset{t_1^n}{argmax} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$
- Acquire frequencies from a training corpus:
  - Word Frequency with given POS
    - suppose ***book*** occurs 14 times in a corpus: 10 times (.001) as **NN (**there are 10000 instances of **NN** in the corpus); 3 times (.003) as **VBP** (the corpus has 1000 **VBP**s**)**, and 1 instance of book (.005) as **VB (**the corpus has 500 **VB**s).
  - Given the previous tag, how often does each tag occur
    - suppose **DT is** followed by **NN** 80,000 times (.53), **JJ** 30,000 times (.2), **NNS** 20,000 times (.13), **VBN** 3,000 (.02) times, … out of a total of 150,000 occurrences of **DT**
- All possible tags for sequence:
  - *The*/**DT** *book*/**{NN|VB|VBP}** *is*/**VBZ** *on*/**IN** *the*/**DT** *table*/**{NN|VB|VBP}**
- Hypothetical probabilities for highest scoring tag sequence:
  - *The*/**DT** *book*/**NN** *is*/**VBZ** *on*/**IN** *the*/**DT** *table*/**NN**
  - *The*/**DT=.4,** *book*/**NN=.001,** *is*/**VBZ=.02,** *on*/**IN=.1,** *the*/**DT=.4**, *table*/**NN=.0005,**
  - **B DT = .61, DT NN = .53, NN VBZ =.44, VBZ IN = .12, IN DT = .05, DT NN = .53 NN E .31**
  - $\prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}) = (.4\times.61)(.001\times.53)(.02\times.44)(.1\times.12)(.4\times.05)(.005\times.53)(1\times.31) \approx 2.4\times10^{-13}$

Defining an HMM

- A Weighted Finite-state Automaton (WFSA)
  - Each transition arc is associated with a probability
  - The sum of all arcs outgoing from a single node is 1
- Markov chain is a WFSA in which an input string uniquely determine path through the Automaton
- Hidden Markov Model (HMM) is a slightly different case because some information (previous POS tags) is unknown (or hidden)
- HMM consists of the following:
  - Q = set of states: q0 (start state), …, qF (final state)
  - A = transition probability matrix of n X n probabilities of transitioning between any pair of n states (n = F+1) – **prior probability (tag sequence)**
  - O = sequence of T observations (**words**) from a vocabulary V
  - B = sequence of observation likelihoods (probability of observation generated at state) – **likelihood (word sequence given tag sequence)**

Example HMM


START
Q0
DT
Q1
JJ
Q2
NN
Q3
IN
Q4
VBZ
Q5
END
QF
.20
.61
.10
.47
.06
.41
.13
.53
.10
.12
.06
.22
1.0
.25
.44
.15
.31
the: .4
an: .05
a: .3
these: .07
...
of: .2
in: .11
on: .1
before: .001
...
angry: .0005
blue: .0011
perfect: .003
orange: .0015
...
is: .02
sees: .0012
hates: .002
sells: .004
...
book: .001
table: .0005
fish: .0002
weasel: .00001
...

Viterbi Algorithm for HMM

Observed_Words = $w_1 \ldots w_T$

- States = q_o, $q_1 \ldots q_N$, q_F

A = N × N matrix such that $a_{i,j}$ is the probability of the transition from q_i to q_j

B = lookup table such that $b_i(w_t)$ is the probability that word t is assigned POS i

viterbi = (N+2) × T matrix # columns are states, rows are words

backpointer = (N+2) × T matrix # highest scoring previous cells for viterbi

for states q from 1 to N:

initialize viterbi[q,1] to $a_{0,q} * b_q(w_1)$ # score transition 0→q given w_1

initialize backpointer[q,1] to 0 (start state)

for word w from 2 to T:

for state q from 1 to N: # for T-1 × N (w,q) pairs

viterbi[q,w] ← $\max_{q'=1}^{N} viterbi[q',t-1] * a_{q',q} * b_q(w_t)$ # score = maximum previous * prior * likelihood

backpointer[q.w] ← $\operatorname{argmax}_{q'=1}^{N} viterbi[q',t-1] * a_{q',q}$ # backpointer = maximum previous

viterbi[qF,T] ← $\max_{q=1}^{N} viterbi[q,T] * a_{q,qF}$ # score = maximum previous * prior * likelihood

backpointer[qF,T] ← $\operatorname{argmax}_{q=1}^{N} viterbi[q,T] * a_{q,qF}$ # backpointer = maximum previous

- return(best_path) # derive by following backpointers from (qF,T) to q_0

Walk Through of HMM from Slide 13

- Initialize scores for first column: transitions from 0 to each possible state given: ***the***
  - The probability of reaching Q1 matching the first item on the tape (***the***) will be .4 X .61 = .244 (this is also the only possibility)
- Set Viterbi(***book***,state) for states 1 to 5 to .61 * prior probability(1,state) * likelihood(word,POS(state))
  - The highest scoring value will be state Q3
    - Score = $(.244 * .53 * .001) \approx .0013$
- And so on until the path includes ***the book is on the table***
- Then we score the final transitions from table in state Q3 to state QF
  - If we had a more complete HMM, we would also need to include a transition from a state marking ***table*** as a VB to state QF.

N-grams

- Prior Probability in our HMM was estimated as:
  - freq(currentPOS, previousPOS) / freq(previousPOS)
  - This is a bigram
- Trigram: freq(POS-2, POS-1, POS) / freq(POS-2,POS-1)
- N-gram: freq(POS-N…POS)/freq(POS-N…POS-1)
- N-grams
  - Used a lot in computational linguistics
  - N-grams can be calculated for: words, POS, other attributes
  - Chapter 4 has more info on N-grams
- J & M show how to extend HMM to tri-grams by calculating the prior based on two previous POS tags instead of one
  - They also show a way of combining unigram, bigram and trigrams

Unknown (OOV) Words

- Possibility 1
  - Assume they are equally ambiguous between all POS
    - Refinements:
      - Assume distribution of unknown = overall POS distribution
      - Assume distribution of unknown = distribution of 1 time words
  - Use N-grams to predict the correct tag
- Possibility 2
  - Use morphology (prefixes, suffixes), orthography (uppercase/lowercase), hyphenation
- Possibility 3: Some combination

Readings

- Jurafsky and Martin, Chapter 5
- NLTK, Chapter 5 (skim, try examples)

Homework # 4: Slide 1

- Download the following file:
  - http://cs.nyu.edu/courses/spring12/CSCI-GA.2590-001/Homework4_corpus.zip
- Unpack the file to produce 2 directories
  - POSData – files in input (text) and output (pos) formats
    - text format:
      - one token (word, punctuation, suffix) per line, sentences end in blank lines
      - Special tokens are replaced: *COMMA* (,); *-LSB-* ([ ); *-RSB-* ( ])
    - pos format: text format plus a POS tag
    - Training vs Dev Corpora
      - training.pos, training.text – use these files to train your system
      - development.pos, development.text – use these files to test your system
  - POSScorer – a scoring program with instructions for use

Homework # 4: Slide 2

- Make a table of the prior probabilities for each of POS tags (assuming bigrams) using the **training** corpus**.**
- Make a likelihood table including all words in the **training** corpus.
- For words not found in the training corpus, the likelihood equals that of the least likely item of that POS. If the least likely NN is .001, then the likelihood of an unknown word given NN is .001.
- Implement an HMM tagger using these tables.
- Test it on the **development** corpus until you are satisfied that it works. Use the scoring software to evaluate your system.
- Submit the resulting system, along with instructions on how to run it and some notes about your results.
- Ang will run it against a **test** corpus which we have set aside.