HMM and Part of Speech Tagging
Adam Meyers
New York University
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:
  – http://repository.upenn.edu/cgi/viewcontent.cgi?article=1603&context=cis_reports
    • Provides list

• To map several POS tagsets to each other, see Table 1 in:
  – http://nlp.cs.nyu.edu/meyers/Annotation%20Compatibility%20Working%20Group%20Report%202006.html

• POS tags:
  – Assume Particular Tokenizations, e.g., *Mary's* → *Mary + 's*
  – Distinguish inflections: e.g., *eat/VB, eat/VBP, eats/VBZ, ate/VBD*
  – Different instances of the same string can have different tags
    • *She wants to eat/VB; They eat/VBP. He eats/VBZ, Those are good eats/NNS*

• Annotators & POS taggers assign tags to each token in a sentence, no exceptions
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)
- **Interrogatives**: 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/VBZ 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:
    • `!@#$%^&*()-_=]|{\}|:;"'"<>./`
    – Punctuation followed by a space, other punctuation, or at the end of line should be separated from words:
      • ...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 2:
  – Period after abbreviations should not be separate from words
    • Most cases covered by list of 100 items (or if sentence end is known)
  – Final periods are not duplicated after abbreviations (consistency issues)
    • These periods serve 2 functions simultaneously (argument for duplication)
    • These periods occupy a single character position
      – argument against duplication – difficulty with calculating character offsets
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
    - Untensed: VB (base form), VBN & VBG (past & present participles)

- 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
Probability

- Estimate of probability of future event based on past observations

\[ P(\text{event}) = \frac{\text{num of events}}{\text{num of trials}} \]

- Conditional Probability: probability of \( X \) given \( Y \)

\[ P( X \mid Y ) = \frac{P( X, Y )}{P( Y )} \]

- Examples relating to POS tags (previous examples with word N-grams):
  - Out of 200 \( DT \) tags, 150 of them are tagging the word \textit{the}
    - If a word is tagged \( DT \), there is a 75% chance that word is \textit{the}
    - Example of likelihood probability
  - The POS after a \( DT \) is \textit{NN} 120 times and \textit{JJ} 60 times:
    - A word following \( DT \) is
      - \( 120/200 = 60\% \) likely to be a singular noun (\textit{NN})
      - \( 60/200 = 30\% \) likely to be a base adjective (\textit{JJ})
    - Examples of transition probability
More Math Terminology

• N instances of a variable looked at individually:
  \( X_1^n \) is the same as \( \{X_1, X_2, X_3, \ldots, X_n\} \) in sequence

• The product of instances of X from 1 to n
  \[ \prod_{i=1}^{n} P(X_i) \]

• Max = the maximum number in a set

• Argmax = the formula that maximizes a particular argument of the formula
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.,
    • Assume that we can estimate probability of whole tag sequence by multiplying simpler probabilities, e.g., sequences of 2 consecutive tags
Probabilistic Assumptions of HMM Tagging

- \[ \hat{t} = \arg \max_{t_1^n} P(t_1^n | w_1^n) \]
  - 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 \( x \), the probability of \( x \) and the probability of \( y \)

- Applying Bayes Rule to Tag Probability
  - \[ \hat{t} = \arg \max_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)} \]
Simplifying Assumptions for HMMs

• Simplification: Drop the denominator
  – Denominator is same for all the tag sequences (the word sequence is given)
    – \( \hat{t} = \arg\max_{t_1^n} P(w_1^n|t_1^n) P(t_1^n) \)
  – 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,
    • \( P(w_i^n|t_i^n) \approx \prod_{i=1}^n P(w_i|t_i) \)
  – 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:
  \( \hat{t} \approx \arg\max_{t_1^n} \prod_{i=1}^n P(w_i|t_i) P(t_i|t_{i-1}) \)

• HMM taggers are fast and achieve precision/recall scores of about 93-95%
Estimating Probability of $\hat{t}$

- We assume that:
  $$\hat{t} \approx \arg\max_t \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

- 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 **VBPs**), and 1 instance of book (.005) as **VB** (the corpus has 500 **VBs**).
    - 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*
      - $\hat{t} = .61$, **DT NN = .53**, **NN VBZ = .44**, **VBZ IN = .12**, **IN DT = .05**, **DT NN = .53** **NN E .31**
      - $\prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1}) = (.4 \times .61)(.001 \times .53)(.02 \times .44)(.1 \times .12)(.4 \times .05)(.005 \times .53)(1 \times .31) \approx 2.4 \times 10^{-13}$

Computational Linguistics
Lecture 4
2017
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: $q_0$ (start state), …, $q_F$ (final state)
  – A = transition probability matrix of $n \times n$ probabilities of transitioning between any pair of $n$ states ($n = F+1$). Called: *prior probability* or *transition probability* of a tag sequence
  – O = sequence of $T$ observations (*words*) from a vocabulary $V$
  – B = sequence of observation likelihoods (probability of observation generated at state) – Called *likelihood* (of word sequence given tag sequence), aka *emission probability*
Example HMM

START
Q0

.20

DT
Q1

IN
Q4

JJ
Q2

NN
Q3

VBZ
Q5

END
QF

the: .4
an: .05
a: .3
these: .07
...

is: .02
sees: .0012
hates: .002
sells: .004
...

of: .2
in: .11
on: .1
before: .001
...

book: .001
table: .0005
fish: .0002
orange: .00001
...

angry: .0005
blue: .0011
perfect: .003
orange: .0015
...

the: .4
an: .05
a: .3
these: .07
...

is: .02
sees: .0012
hates: .002
sells: .004
...

of: .2
in: .11
on: .1
before: .001
...

book: .001
table: .0005
fish: .0002
orange: .00001
...
Go to Ralph's Viterbi Demo for Fish Sleep
Viterbi Algorithm for HMM

Observed Words = \( w_1 \ldots w_T \)

- States = \( q_0, q_1 \ldots q_N, q_F \)

  \( A = N \times N \) 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 POS \( i \) is realized as word \( t \)

  \( \text{viterbi} = (N+2) \times T \) matrix \# columns are states, rows are words

  \( \text{backpointer} = (N+2) \times T \) matrix \# highest scoring previous cells for viterbi

for states \( q \) from 1 to \( N \):

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

  initialize \( \text{backpointer}[q,1] \) to 0 (start state)

for word \( w \) from 2 to \( T \):

  for state \( q \) from 1 to \( N \):

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

    \( \text{backpointer}[q,w] \leftarrow \max_{q'=1}^{N} \text{viterbi}[q',t-1] \ast a_{q',q} \) \# backpointer = maximum previous

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

backpointer[\( qF,T \)] ← \( \max_{q=1}^{N} \text{viterbi}[q,T] \ast a_{q,qF} \) \# backpointer = maximum previous

return(best_path) \# derive by following backpointers from \( (qF,T) \) to \( q_0 \)
**Walk Through: The orange is on the table. (ignoring period)**

\[
1 \times .4 \times .61 \times .00001 \times .53 \times .02 \times .33 \times .1 \times .12 \times .4 \times .6 \times .54 \times .0005 \times .33 \times 1 = 2.19 \times 10^{-15}
\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1: The</th>
<th>2: orange</th>
<th>3: is</th>
<th>4: on</th>
<th>5: the</th>
<th>6: table</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td></td>
<td>.4 \times .61</td>
<td></td>
<td></td>
<td></td>
<td>.4 \times .6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.1 \times .12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJ</td>
<td></td>
<td>.0015 \times .47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBZ</td>
<td></td>
<td>.00001 \times .53</td>
<td>.02 \times .44</td>
<td></td>
<td></td>
<td></td>
<td>.53 \times .0005</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.00001 \times .53</td>
<td>.53 \times .0005</td>
<td></td>
</tr>
<tr>
<td>End</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.33 \times 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comments on Viterbi Trace

• 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 \(0.4 \times 0.61 = 0.244\) (this is also the only possibility)

• The adjective sense of *orange* is more likely locally, but leads to a dead end

• The transitions from B and the transition to E are necessary parts of the process.
Unknown (OOV) Words

• Possibility 1
  – Assume all POS tags have the same probability (e.g., 1/1000)
  – In effect, only use transitions to predict the correct tag
• Possibility 2
  – Use morphology (prefixes, suffixes), orthography (uppercase/lowercase), hyphenation
• Possibility 3:
  – Words occurring once in corpus = instances of UNKNOWN_WORD
  – Distribution of UNKNOWN_WORD used for OOV words
• Possibility 4: Some combination
  – Example: divide UNKNOWN_WORD into morphological classes like UNKNOWN_WORD_ENDING_IN_S
Homework

- [http://cs.nyu.edu/courses/spring17/CSCI-UA.0480-009/homework4.html](http://cs.nyu.edu/courses/spring17/CSCI-UA.0480-009/homework4.html)
- Guidance on Program – Next few slides
- This assignment can be completed alone or with a partner
  - A slightly more elaborate system will be expected if you choose to work with a partner.
Implement Simple version of training stage first

- Data 2 fields (separated by tab): word and POS
  - Start of file = begin of sentence
  - Blank line = begin and end of sentence
  - End of file = end of sentence
- Make 2 hash tables (e.g., Python dictionaries)
  1. POS → list of frequencies of words that occur with that POS
     - Example: likelihood['DT'] → {'the':1500,'a':200,'an':100, …}
     - Hash table of POSs with each value a hash table from words to frequencies
  2. STATE → list of frequencies of following states
     - Example: Transition['Begin_Sent'] → {'DT':1000,'NNP':500,'VB':200, …}
     - Example: Transition['DT'] → {'NN':500,'NNP':200,'VB':30, …}
     - Hash table of states with each a value a hash table from states to frequencies
     - States = Begin_Sent, End_Sent and all POSs
- Go through the data one line at a time
  - Record frequencies for both 1 and 2
  - Loop thru hash table and convert frequencies into probabilities
    - freq/total = probability
Simple Version of Transducer

- Make a 2 dimensional array (or equivalent)
  - columns represent positions in the text
    - 0 = start of sentence
    - N = Nth token
    - Length+1 = end of sentence
  - Rows represent all possible states
    - Start symbol
    - End symbol
    - All POSs (NN, JJ, …) found in the corpus
- Traverse the chart as per the algorithm (fish sleep slides, etc.)
  - Score all possible transitions from position = 0, state = Start
  - For all states at position 1 with non-zero scores, score all possible transitions from Start
  - Do subsequent transitions until arriving at end of the string.
  - At each position in the chart record:
    - the highest score
    - the previous state that resulted in this score
Calculating Probabilities

- The probability of each transition to state $N$ for token $T$ is assumed to be the product of 3 factors
  - Probability that state $N$ occurs with token $T$
    - There is 100% chance that the start state will be at the beginning of the sentence
    - There is 100% chance that the end state will be at the end of the sentence
    - If a token was observed in the training corpus, look up probability from table
    - For Out of Vocabulary words, there are several strategies
      - Simple strategy (for first implementation): 100% divided by number of states
      - Other strategies are a separate discussion
  - Probability that state $N$ occurs, previous state
    - Look up in table, calculate for every possible previous state
    - Highest Probability of previous state (calculate for each previous state)
    - For each new state, choose the highest score (this is the bigram model)
- Choose the POS tag sequence resulting in the highest score in the end state
OOV Strategies from slide 20

- Default (use until other parts of program are debugged)
  - Assume all POS tags have the same probability (e.g., 1/1000)
  - In effect, only use transitions to predict the correct tag
- Morphology
  - Use prefixes, suffixes, uppercase/lowercase, hyphenation, to predict POS classes of OOV words
  - Assign “made up” values based on these features?
- Computer probability of UNKNOWN_WORD
  - Treat words occurring once in training collectively as UNKNOWN_WORD
    - don't record them separately
  - UNKNOWN_WORD probability used for OOV words by transducer
- Combination:
  - UNKNOWN_ending_in_s, UNKNOWN_ending_in_ed, UNKNOWN_with_capital_letter, ...
How you Might Improve your Score

• Do error analysis on development corpus and base changes on what you find.

• Implement a trigram algorithm
  – See Jurafsky and Martin (p. 149)
  – 4-gram is a waste of time for this size corpus

• A particularly clever OOV system

• Manual rule system using constraints, e.g., slide 7.
  – For words with frequency>1, assume the disjunction of observed labels is possible
  – Rule out possibilities according to constraints
  – Run this and compare results with HMM system
  – Figure out way of combining results based on error analysis
    • Voting, weighted combinations, etc.
Grading

• 1 Person can get a 9 or 10 with:
  – a bigram system and an implementation of an OOV system based on words occurring once
  – an accuracy score above 94 on the test corpus

• 2 Person system can get a 9 or 10 with
  – Same as one person system plus at least one extra interesting attempt, even if unsuccessful

• Include a short write-up of what you did, so it is easier to evaluate.
2 Person Collaboration

- Indicate on your submission documents that you are collaborating and indicate who you are collaborating with.
- Collaborators should submit the same documents twice on NYUClasses to make sure there is no confusion.
- Indicate who did what, e.g.,
  - Person 1: create the tables with the probabilities
  - Person 2: create the initial version of Viterbi and a very simple OOV strategy (assume all POS have equal probability)
  - Person 1: OOV strategy based on words occurring once
  - Person 2: error analysis on development corpus to determine next improvements
  - Person 1 and 2: Manual Rule based system
  - Etc.