Part-of-speech Tagging & Hidden Markov Model Intro
Lecture #10
Introduction to Natural Language Processing
CMPSCI 585, Fall 2007
University of Massachusetts Amherst

Andrew McCallum
Today’s Main Points

• Tips for HW#4
• Summary of course feedback

• Part-of-speech tagging
  – What is it? Why useful?
• Return to recipe for NLP problems
• Hidden Markov Models
  – Definition
  – Generative Model
  – Next time: Dynamic programming with Viterbi algorithm
Class surveys very helpful

• Learning something?
  – Yes! Very edifying!
  – Yes. Lots. Statistical NLP is a lot of fun.
  – Yes! Both theory and practice.
  – Yes, I have been learning a lot. Particularly since the probability class pretty much everything is new to me.
  – Yes. I went to the Google talk on Machine Translation and mostly understood it, based entirely on experience from this class.
  – Yes. My understanding of dynamic programming has greatly increased.
Class Surveys

• Pace and Lectures
  – I like that we cover a large breadth of material and don’t doddle.
  – Balance between theory and applications is great.
  – The slides are really good. I also like when math is demo’ed on the whiteboard.
  – Everything working well.
  – I like the quizzes. Helps me know what I should be learning.
  – In-class exercises very helpful. Let’s have more!
  – Pace: 5 just right, 3 slightly too fast, 3 slightly too slow.

  – Love the in-class exercises and group discussions.
  – Enthusiasm is motivating and contagious. Available after class to offer deeper insights, answer questions, etc.
  – Love hearing about NLP people history lessons
Class Surveys

- **Homeworks**
  - Homework assignments are fantastic, especially the open-ended aspect!
  - The reinforce the learning.
  - Interesting, fun, promotes creativity, very much unlike other homeworks that just “have to be done”. I like particularly that we get a choice... room for doing stuff one finds interesting.
  - Fun because we get to play around; lots of freedom!
  - Helpful that some of the less interesting infrastructure (file reading...) is provided.

- Initially confused about the report format. An example would help. (But comfortable with them now.)
- Make grading rubric / expectations more clear.
- Grading harsh--points off for not going above and beyond, even though the specified requirements were met. Hard to tell how much creativity is enough.
Class Surveys

• **Workload**
  – (No one complaining.)
  – “Work is fun, so it feels like less.”
Class Surveys

• Suggestions & Concerns
  – Would like more exercises and take-home quizzes.
  – Post slides sooner.
  – Make HW grading policy more clear.
HW #4 Tasks

• Naive Bayes
  – document classification (SPAM dataset provided)
  – part-of-speech tagger

• N-gram Language model
  – Train and generate language
    • look for phase changes?
    • experiment with different smoothing methods?
  – Foreign language classifier
  – Rank output of a machine translation system
HW#4 Help
Evaluation

Result of running classifier on a test set:

| filename | trueclass | predclass | p(predclass|doc) |
|----------|-----------|-----------|-------------|
|          |           |           |             |
|          |           |           |             |

... 

<table>
<thead>
<tr>
<th></th>
<th>true spam</th>
<th>true ham</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred spam</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>pred ham</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

Accuracy = (TP+TN) / (TP+TN+FP+FN)
Precision = TP / (TP+FP)
Recall = TP / (TP+FN)
F1 = harmonic mean of Precision & Recall
HW#4 Help

Precision-Recall Curve

Typically if $p(\text{spam}) > 0.5$, then label as spam, but can change 0.5 “threshold.” Each threshold yields a new precision/recall pair. Plot them:
HW#4 Help

Accuracy-Coverage Curve

Result of running classifier on a test set:

| filename | trueclass | predclass | p(predclass | doc) |
|----------|-----------|-----------|-------------|
|          |           |           |             |

... 

<table>
<thead>
<tr>
<th></th>
<th>true spam</th>
<th>true ham</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred spam</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>pred ham</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

Accuracy = \( \frac{TP+TN}{TP+TN+FP+FN} \)

Precision = \( \frac{TP}{TP+FP} \)

Recall = \( \frac{TP}{TP+FN} \)

F1 = harmonic mean of Precision & Recall
HW#4 Help
Working with log-probabilities

\[ p(c|d) \propto p(c) \prod_{i} p(w_i|c) \]

\[ \log(p(c|d)) \propto \log(p(c)) + \sum_{i} \log(p(w_i|c)) \]

- Getting back to \( p(c|d) \)
  - Subtract a constant to make all non-positive
  - \( \exp() \)
HW#4 Help
The importance of train / test splits

• When measuring accuracy, we want an estimate on how well a classifier will do on “future data”.

• “Testing” on the “training data” doesn’t do this.

• Split data. Train on one half. Test on the other half.
Part of Speech Tagging and Hidden Markov Models
Grammatical categories: parts-of-speech

- Nouns: people, animals, concepts, things
- Verbs: expresses action in the sentence
- Adjectives: describe properties of nouns

- The \( \left\{ \begin{array}{l} 
  \text{sad} \\
  \text{intelligent} \\
  \text{green} \\
  \text{fat} \\
  \ldots 
\end{array} \right\} \) one is in the corner.

“Substitution test”
The Part-of-speech Tagging Task

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

• Uses:
  – text-to-speech (how do we pronounce “lead”?)
  – can differentiate word senses that involve part of speech differences (what is the meaning of “interest”)
  – can write regexps like Det Adj* N* over the output (for filtering collocations)
  – can be used as simpler “backoff” context in various Markov models when too little is known about a particular history based on words instead.
  – preprocessing to speed up parser (but a little dangerous)
  – tagged text helps linguists find interesting syntactic constructions in texts (“ssh” used as a verb)
Tagged Data Sets

• Brown Corpus
  – Designed to be a representative sample from 1961
    • news, poetry, …
  – 87 different tags

• Claws5 “C5”
  – 62 different tags

• Penn Treebank
  – 45 different tags
  – Most widely used currently
### Part-of-speech tags, examples

<table>
<thead>
<tr>
<th>PART-OF-SPEECH</th>
<th>TAG</th>
<th>EXAMPLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjective</td>
<td>JJ</td>
<td>happy, bad</td>
</tr>
<tr>
<td>Adjective, comparative</td>
<td>JJR</td>
<td>happier, worse</td>
</tr>
<tr>
<td>Adjective, cardinal number</td>
<td>CD</td>
<td>3, fifteen</td>
</tr>
<tr>
<td>Adverb</td>
<td>RB</td>
<td>often, particularly</td>
</tr>
<tr>
<td>Conjunction, coordination</td>
<td>CC</td>
<td>and, or</td>
</tr>
<tr>
<td>Conjunction, subordinating</td>
<td>IN</td>
<td>although, when</td>
</tr>
<tr>
<td>Determiner</td>
<td>DT</td>
<td>this, each, other, the, a, some</td>
</tr>
<tr>
<td>Determiner, postdeterminer</td>
<td>JJ</td>
<td>many, same</td>
</tr>
<tr>
<td>Noun</td>
<td>NN</td>
<td>aircraft, data</td>
</tr>
<tr>
<td>Noun, plural</td>
<td>NNS</td>
<td>women, books</td>
</tr>
<tr>
<td>Noun, proper, singular</td>
<td>NNP</td>
<td>London, Michael</td>
</tr>
<tr>
<td>Noun, proper, plural</td>
<td>NNPS</td>
<td>Australians, Methodists</td>
</tr>
<tr>
<td>Pronoun, personal</td>
<td>PRP</td>
<td>you, we, she, it</td>
</tr>
<tr>
<td>Pronoun, question</td>
<td>WP</td>
<td>who, whoever</td>
</tr>
<tr>
<td>Verb, base present form</td>
<td>VBP</td>
<td>take, live</td>
</tr>
</tbody>
</table>
Closed, Open

• Closed Set tags
  – Determiners
  – Prepositions
  – ...

• Open Set tags
  – Noun
  – Verb
Why is this such a big part of NLP?

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task
  - Can be done well with methods that look at local context
  - (Though should “really” do it by parsing!)
Fed raises interest rates 0.5% in effort to control inflation

NY Times headline 17 May 2000
Fed raises interest rates 0.5% in effort to control inflation
Degree of Supervision

- **Supervised**: Training corpus is tagged by humans
- **Unsupervised**: Training corpus isn’t tagged
- **Partly supervised**: E.g. Training corpus isn’t tagged, but you have a dictionary giving possible tags for each word

- We’ll start with the supervised case (in later classes we may move to lower levels of supervision).
Current Performance

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

• Using state-of-the-art automated method, how many tags are correct?
  – About 97% currently
  – But baseline is already 90%
    • Baseline is performance of simplest possible method:
      • Tag every word with its most frequent tag
      • Tag unknown words as nouns
Recipe for solving an NLP task

Input: the lead paint is unsafe
Output: the/Det lead/N paint/N is/V unsafe/Adj

1) **Data**: Notation, representation
2) **Problem**: Write down the problem in notation
3) **Model**: Make some assumptions, define a parametric model (often generative model of the data)
4) **Inference**: How to search through possible answers to find the best one
5) **Learning**: How to estimate parameters
6) **Implementation**: Engineering considerations for an efficient implementation
Work out several alternatives on the board...
(Hidden) Markov model tagger

• View sequence of tags as a Markov chain. Assumptions:
  – Limited horizon \( P(x_{t+1}|x_1, \ldots x_t) = P(x_{t+1}|x_t) \)
  
  – Time invariant (stationary) \( P(x_{t+1}|x_t) = P(x_2|x_1) \)

  – We assume that a word’s tag only depends on the previous tag (limited horizon) and that his dependency does not change over time (time invariance)

  – A state (part of speech) generates a word. We assume it depends only on the state.

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

• A stochastic process has the **Markov property** if the conditional probability distribution of future states of the process, given the current state, depends only upon the current state, and conditionally independent of the past states (the *path* of the process) given the current state.

• A process with the Markov property is usually called a **Markov process**, and may be described as *Markovian*.

\[
\Pr[X(t+h) = y \mid X(s) = x(s), s \leq t] = \Pr[X(t+h) = y \mid X(t) = x(t)], \quad \forall h > 0.
\]
HMM as Finite State Machine

transitions

\[ P(x_{t+1}|x_t) \]

emissions

for above in …

\[ P(o_t|x_t) \]
HMM as Bayesian Network

- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations (words)
Applications of HMMs

• NLP
  – Part-of-speech tagging
  – Word segmentation
  – Information extraction
  – Optical Character Recognition (OCR)

• Speech recognition
  – Modeling acoustics

• Computer Vision
  – gesture recognition

• Biology
  – Gene finding
  – Protein structure prediction

• Economics, Climatology, Communications, Robotics…
(One) Standard HMM formalism

- \((X, O, x_s, A, B)\) are all variables. Model \(\mu = (A, B)\)
- \(X\) is state sequence of length \(T\); \(O\) is observation seq.
- \(x_s\) is a designated start state (with no incoming transitions). (Can also be separated into \(\pi\) as in book.)
- \(A\) is matrix of transition probabilities (each row is a conditional probability table (CPT))
- \(B\) is matrix of output probabilities (vertical CPTs)

\[
P(X, O|\mu) = \prod_{t=1}^{T} a[x_t|x_{t-1}] b[o_t|x_t]
\]

- HMM is a probabilistic (nondeterministic) finite state automaton, with probabilistic outputs (from vertices, not arcs, in the simple case)
Three fundamental questions for an HMM:

1) Compute the probability of a given observation sequence, when tag sequence is hidden (language modeling)

2) Given an observation sequence, find the most likely hidden state sequence (tagging)  **DO THIS NEXT**

3) Given observation sequence(s) and a set of states, find the parameters that would make the observations most likely (parameter estimation)
Most likely hidden state sequence

- Given $O = (o_1, \ldots, o_T)$ and model $\mu = (A, B)$
- We want to find

$$
\arg \max_X P(X|O, \mu) = \arg \max_X \frac{P(X, O|\mu)}{P(O|\mu)} = \arg \max_X P(X, O|\mu)
$$

- $P(O, X| \mu) = P(O|X, \mu) P(X| \mu)$
- $P(O|X, \mu) = b[x_1|o_1] b[x_2|o_2] \ldots b[x_T|o_T]$
- $P(X| \mu) = a[x_1|x_2] a[x_2|x_3] \ldots a[x_{T-1}|x_T]$
- $\arg \max_X P(O, X| \mu) = \arg \max x_1, x_2, \ldots x_T$
- Problem: $\arg \max$ is exponential in sequence length!
Representation for Paths: Trellis

States

X1
x2
x3
x4

Time 1 2 3 4 ... T
Representation for Paths: Trellis

States

X1

x2

x3

x4

Time

1

2

3

4

... 

T
$\delta_i(t) = \text{Probability of most likely path that ends at state } i \text{ at time } t.$
Finding Probability of Most Likely Path using Dynamic Programming

- Efficient computation of max over all states
- Intuition: Probability of the first $t$ observations is the same for all possible $t+1$ length sequences.
- Define forward score:
  \[
  \delta_i(t) = \max_{x_1 \ldots x_{t-1}} P(o_1 o_2 \ldots o_t, x_1 \ldots x_{t-1}, x_t = i | \mu)
  \]
  \[
  \delta_j(t + 1) = \max_{i=1 \ldots N} \delta_i(t) a[x_j | x_i] b[o_{t+1} | x_j]
  \]
- Compute it recursively from the beginning
- (Then must remember best paths to get arg max.)
Finding the Most Likely State Path with the Viterbi Algorithm

[Viterbi 1967]

• Used to efficiently find the state sequence that gives the highest probability to the observed outputs

• Maintains two dynamic programming tables:
  – The probability of the best path (max)
    \[ \delta_j(t+1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j] \]
  – The state transitions of the best path (arg)
    \[ \psi_j(t+1) = \arg \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j] \]

• Note that this is different from finding the most likely tag for each time \( t \)!
Viterbi Recipe

• Initialization

\[ \delta_j(0) = 1 \text{ if } x_j = x_s, \quad \delta_j(0) = 0 \text{ otherwise.} \]

• Induction

\[ \delta_j(t + 1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j] \]

Store backtrace

\[ \psi_j(t + 1) = \text{arg} \max_{i=1..N} \delta_i(t) a[x_j|x_i] b[o_{t+1}|x_j] \]

• Termination and path readout

\[ \hat{x}_T = \arg \max_{i=1..N} \delta_i(T) \]
\[ \hat{x}_t = \psi \hat{x}_{t+1}(t + 1) \]

Probability of entire best seq.

\[ P(\hat{X}) = \max_{i=1..N} \delta_i(T) \]