Outline

- What is part-of-speech (POS) and POS tagging?
- Hidden Markov Model (HMM) for POS tagging
- Learning an HMM
- Prediction with an learned HMM (inference)

Parts of Speech

- Perhaps starting with Aristotle in the West (384–322 BCE), there was the idea of having parts of speech (POS)
  - a.k.a lexical categories, word classes, “tags”
  - Lowest level of syntactic analysis

English Parts of Speech (POS) Tagsets

- Original Brown corpus used a large set of 87 POS tags.
- Most common in NLP today is the Penn Treebank set of 45 tags.
  - Tagset used in the slides.
  - Reduced from the Brown set for use in the context of a parsed corpus (i.e. Penn Treebank).

English Parts of Speech

- Noun (person, place or thing)
  - Singular (NN): dog, fork
  - Plural (NNS): dogs, forks
  - Proper (NNP, NNPS): John, Springfields
  - Personal pronoun (PRP): I, you, he, she, it
  - Wh-pronoun (WP): who, what

- Verb (actions and processes)
  - Base, infinitive (VB): eat
  - Past tense (VBD): ate
  - Gerund (VBG): eating
  - Past participle (VBN): eaten
  - 3rd person singular present tense (VBP): eats
  - Modal (MD): should, can
  - To (TO): to (to eat)

Adjective (modify nouns)

- Basic (JJ): red, tall
- Comparative (JRR): redder, taller
- Superlative (JJR): reddest, tallest

Adverb (modify verbs)

- Basic (RB): quickly
- Comparative (RBR): quicker
- Superlative (RBS): quickest

Preposition (IN): on, in, by, to, with

Determiner:

- Basic (DT): a, an, the
- WH-determiner (WDT): which, that

Coordinating Conjunction (CC): and, but, or

Particle (RP): off (took off), up (put up)
Open vs. Closed classes

- Open vs. Closed classes
  - Closed:
    - determiners: a, an, the
    - pronouns: she, he, I
    - prepositions: on, under, over, near, by, ...
  - Why "closed"?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

Ambiguity in POS Tagging

- "Like" can be a verb or a preposition
  - I like/VBP candy.
  - Time flies like/IN an arrow.
- "Around" can be a preposition, particle, or adverb
  - I bought it at the shop around/IN the corner.
  - I never got around/RP to getting a car.
  - A new Prius costs around/RB $25K.

POS Tagging

- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS tagging performance

- How many tags are correct? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
      - Take an annotated corpus (or a dictionary), tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (the, a, etc.) and for punctuation marks!

- Input: plays well with others
- Ambiguity: NNS/VBZ UH/JJ/NN/RB IN NNS
- Output: Plays/VBZ well/RB with/IN others/NNS
- Uses:
  - Text-to-speech (how do we pronounce "lead"?)
  - Can write regexps over the output for phrase extraction
  - Noun phrase: (Det) Adj* N+
  - As input to or to speed up a full parser
How difficult is POS tagging?

- Word types: roughly speaking, unique words
- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - *I know that he is honest* = IN (preposition)
  - *Yes, that play was nice* = DT (determiner)
  - *You can't go that far* = RB (adverb)
- 40% of the word tokens are ambiguous

Sources of information

- What are the main sources of information for POS tagging? “Bill saw that man yesterday”
  - **Contextual**: Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VBD IN VB NN
  - **Local**: Knowledge of word probabilities
    - *man* is rarely used as a verb...
- The latter proves the most useful, but the former also helps
- Sometimes these preferences are in conflict:
  - *The trash can is in the garage*

More and Better Features ➔

Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word: the: the → DT
  - Lowercased word: Importantly: importantly → RB
  - Prefixes: unfathomable: un- → JJ
  - Suffixes: Importantly: -ly → RB
  - Capitalization: Meridian: CAP → NNP
  - Word shapes: 35-year: d-x → JJ

POS Tagging Approaches

- **Rule-Based**: Human crafted rules based on lexical and other linguistic knowledge.
- **Learning-Based**: Trained on human annotated corpora like the Penn Treebank.
  - **Statistical models**: Hidden Markov Model (HMM) – this lecture, Maximum Entropy Markov Model (MEMM), Conditional Random Field (CRF)
  - **Rule learning**: Transformation Based Learning (TBL)
  - **Neural networks**: Recurrent networks like Long Short Term Memory (LSTMs)
- Generally, learning-based approaches have been found to be more effective overall, taking into account the total amount of human expertise and effort involved.

Outline

- What is part-of-speech (POS) and POS tagging?
  ➔ **Hidden Markov Model (HMM) for POS tagging**
  - Learning an HMM
  - Prediction with an learned HMM (inference)

Hidden Markov Model
Markov Model / Markov Chain
• A finite state machine with probabilistic state transitions.
• Makes Markov assumption that next state only depends on the current state and independent of previous history.

Sample Markov Model for POS (a finite state machine)

Hidden Markov Model
• Probabilistic generative model for sequences.
• Assume an underlying set of hidden (unobserved) states in which the model can be (e.g. part-of-speech).
• Assume probabilistic transitions between states over time (e.g. transition from POS to another POS as sequence is generated).
• Assume a probabilistic generation of tokens from states (e.g. words generated for each POS).
Sample HMM Generation

Formally, Markov Sequences

- Consider a sequence of random variables $X_1, X_2, \ldots, X_m$ where $m$ is the length of the sequence.
- Each variable $X_i$ can take any value in $\{1, 2, \ldots, k\}$.
- How do we model the joint distribution
  \[ P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) \]

Homogeneous Markov Chains

- In a homogeneous Markov chain, we make an additional assumption, that for $j = 2 \ldots m$,
  \[ P(X_j = x_j | X_{j-1} = x_{j-1}) = q(x_j | x_{j-1}) \]
  where $q(x'|x)$ is some function.
- Idea behind this assumption: the transition probabilities do not depend on the position in the Markov chain (do not depend on the index $j$).
Homogeneous Markov Chains

- In a homogeneous Markov chain, we make an additional assumption, that for \( j = 2 \ldots m \),
  \[
P(X_j = x_j | X_{j-1} = x_{j-1}) = q(x_j | x_{j-1})
\]
  where \( q(x' | x) \) is some function

- Idea behind this assumption: the transition probabilities do not depend on the position in the Markov chain (do not depend on the index \( j \))

"the Markov Chains follows the Markov assumption"

Markov Models

- Our model is then as follows:
  \[
p(x_1, x_2, \ldots, x_m; \theta) = q(x_1)^m \prod_{j=2}^m q(x_j | x_{j-1})
\]

- Parameters in the model:
  \[
  q(x) \text{ for } x = \{1, 2, \ldots, k\}
  \]
  Constraints: \( q(x) \geq 0 \) and \( \sum_{x=1}^k q(x) = 1 \)
  \[
  q(x' | x) \text{ for } x = \{1, 2, \ldots, k\} \text{ and } x' = \{1, 2, \ldots, k\}
  \]
  Constraints: \( q(x' | x) \geq 0 \) and \( \sum_{x=1}^k q(x' | x) = 1 \)

Probabilistic Models for Sequence Pairs – words and POS tags

- We have two sequences of random variables:
  \( X_1, X_2, \ldots, X_m \) and \( S_1, S_2, \ldots, S_m \)

- Intuitively, each \( X_i \) corresponds to an "observation" and each \( S_i \) corresponds to an underlying "state" that generated the observation. Assume that each \( S_i \) is in \( \{1, 2, \ldots, k\} \), and each \( X_i \) is in \( \{1, 2, \ldots, o\} \)

- How do we model the joint distribution
  \[
P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)
\]

Probabilistic Models for Sequence Pairs – words and POS tags

- We have two sequences of random variables:
  \( X_1, X_2, \ldots, X_m \) and \( S_1, S_2, \ldots, S_m \)

  Words
  \[
  \text{Part-of-Speech tags}
\]

  Words
  \[
  \text{Part-of-Speech tags}
\]

- Intuitively, each \( X_i \) corresponds to an "observation" and each \( S_i \) corresponds to an underlying "state" that generated the observation. Assume that each \( S_i \) is in \( \{1, 2, \ldots, k\} \), and each \( X_i \) is in \( \{1, 2, \ldots, o\} \)

- How do we model the joint distribution
  \[
P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)
\]

Hidden Markov Models (HMMs)

- In HMMs, we assume that:
  \[
P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)
  = P(S_1 = s_1) \prod_{j=2}^m P(S_j = s_j | S_{j-1} = s_{j-1}) \prod_{j=1}^m P(X_j = x_j | S_j = s_j)
\]

- Firstly, why would we want to model the joint distribution?

\[
P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)
\]
Independence Assumptions in HMMs

- By the chain rule, the following equality is exact:
\[
P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m) = P(S_1 = s_1, \ldots, S_m = s_m) \times P(X_1 = x_1, \ldots, X_m = x_m | S_1 = s_1, \ldots, S_m = s_m)
\]

- Assumption 1: the state sequence forms a Markov chain

  e.g. Part-of-Speech tags

\[
P(S_1 = s_1, \ldots, S_m = s_m) = P(S_1 = s_1) \prod_{j=2}^m P(S_j = s_j | S_{j-1} = s_{j-1})
\]

Formally

- The model takes the following form:
\[
p(x_1 \ldots x_m, s_1, \ldots, s_m; \theta) = t(s_1) \prod_{j=2}^m t(s_j | s_{j-1}) \prod_{j=1}^m e(x_j | s_j)
\]

- Parameters in the model:
  1. Initial state parameters \( t(s) \) for \( s \in \{1, 2, \ldots, k\} \)
  2. Transition parameters \( t(s' | s) \) for \( s, s' \in \{1, 2, \ldots, k\} \)
  3. Emission parameters \( e(x | s) \) for \( s \in \{1, 2, \ldots, k\} \) and \( x \in \{1, 2, \ldots, n\} \)

Outline

- What is part-of-speech (POS) and POS tagging?
- Hidden Markov Model (HMM) for POS tagging
  - Learning an HMM
  - Prediction with an learned HMM (inference)

HMM

- Parameter estimation
  - Learning the probabilities from training data
  - \( P(\text{verb}|\text{noun})? \), \( P(\text{apples}|\text{noun})? \)

- Inference: Viterbi algorithm (dynamic programming)
  - Given a new sentence, what are the POS tags for the words?
Parameter Estimation with Fully Observed Data

- We’ll now discuss parameter estimates in the case of **fully observed data**. For \( i = 1 \ldots n \), we have pairs of sequences \( x_{ij} \) for \( j = 1 \ldots m \) and \( s_{ij} \) for \( j = 1 \ldots m \). (i.e., we have \( n \) training examples, each of length \( m \).)

**Parameter Estimation: Transition Parameters**

- \( P(\text{verb} | \text{noun}) \)

**Parameter Estimation: Emission Parameters**

- \( P(\text{apples} | \text{noun}) \)

---

- Assume we have fully observed data: for \( i = 1 \ldots n \), we have pairs of sequences \( x_{ij} \) for \( j = 1 \ldots m \) and \( s_{ij} \) for \( j = 1 \ldots m \).
- Define \( \text{count}(i, s \rightarrow s') \) to be the number of times state \( s' \) follows state \( s \) in the \( i \)th training example. More formally:

\[
\text{count}(i, s \rightarrow s') = \sum_{j=1}^{m-1} [s_{ij} = s \land s_{i,j+1} = s']
\]

(We define \([x]\) to be 1 if \( x \) is true, 0 otherwise.)
- The maximum-likelihood estimates of transition probabilities are then

\[
t(s'|s) = \frac{\sum_{i=1}^{n} \text{count}(i, s \rightarrow s')}{\sum_{s'} \sum_{i=1}^{n} \text{count}(i, s \rightarrow s')}
\]

---

- Assume we have fully observed data: for \( i = 1 \ldots n \), we have pairs of sequences \( x_{ij} \) for \( j = 1 \ldots m \) and \( s_{ij} \) for \( j = 1 \ldots m \).
- Define \( \text{count}(i, s \rightarrow x) \) to be the number of times state \( s \) is paired with emission \( x \). More formally:

\[
\text{count}(i, s \rightarrow x) = \sum_{j=1}^{m} [s_{ij} = s \land x_{ij} = x]
\]
- The maximum-likelihood estimates of emission probabilities are then

\[
e(x|s) = \frac{\sum_{i=1}^{n} \text{count}(i, s \rightarrow x)}{\sum_{s} \sum_{i=1}^{n} \text{count}(i, s \rightarrow x)}
\]

---

**Parameter Estimation: Initial State Parameters**

- Assume we have fully observed data: for \( i = 1 \ldots n \), we have pairs of sequences \( x_{ij} \) for \( j = 1 \ldots m \) and \( s_{ij} \) for \( j = 1 \ldots m \).
- Define \( \text{count}(i, s) \) to be 1 if state \( s \) is the initial state in the sequence, and 0 otherwise:

\[
\text{count}(i, s) = [s_{i,1} = s]
\]
- The maximum-likelihood estimates of initial state probabilities are:

\[
t(s) = \frac{\sum_{i=1}^{n} \text{count}(i, s)}{n}
\]
Outline

• What is part-of-speech (POS) and POS tagging?
• Hidden Markov Model (HMM) for POS tagging
• Learning an HMM
  • Prediction with an learned HMM (inference)

HMM

• Parameter estimation
• Inference: Viterbi algorithm (dynamic programming)

The Viterbi Algorithm

• Goal: for a given input sequence \( x_1, \ldots, x_m \), find
  
  \[
  \arg \max_{s_1, \ldots, s_m} p(x_1, \ldots, x_m, s_1, \ldots, s_m; \theta)
  \]

• This is the most likely state sequence \( s_1 \ldots s_m \) for the given input sequence \( x_1 \ldots x_m \)

Most Likely State Sequence

• Given an observation sequence, \( X \), and a model, what is the most likely state sequence, \( S=s_1, s_2, \ldots, s_m \), that generated this sequence from this model?
• Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.

Most Likely State Sequence

• Given an observation sequence, \( X \), and a model, what is the most likely state sequence, \( S=s_1, s_2, \ldots, s_m \), that generated this sequence from this model?
• Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.
Most Likely State Sequence

- Given an observation sequence, \( X \), and a model, what is the most likely state sequence, \( S = s_1,s_2,\ldots,s_m \), that generated this sequence from this model?

- Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.

The Viterbi Algorithm

- Goal: for a given input sequence \( x_1,\ldots,x_m \), find

$$
\arg\max_{s_1,\ldots,s_m} p(x_1,\ldots,x_m, s_1\ldots,s_m; \theta)
$$

- The Viterbi algorithm is a dynamic programming algorithm. Basic data structure:

$$
\tau[j,s]
$$

will be a table entry that stores the maximum probability for any state sequence ending in state \( s \) at position \( j \). More formally: \( \tau[1,s] = t(s|x_1) \), and for \( j > 1 \),

- The goal: find a path with highest probability
The Viterbi Algorithm

- **Goal:** for a given input sequence \( x_1, \ldots, x_m \), find

\[
\arg \max_{s_1, \ldots, s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \theta)
\]

- The **Viterbi algorithm** is a dynamic programming algorithm. Basic data structure: \( \pi[j, s] \) Why do we need this data structure?

will be a table entry that stores the maximum probability for any state sequence ending in state \( s \) at position \( j \). More formally: \( \pi[1, s] = t(s)e(x_1|s) \), and for \( j > 1 \).

The Viterbi Algorithm

- **Initialization:** for \( s = 1 \ldots k \)

\[
\pi[1, s] = t(s)e(x_1|s)
\]

- For \( j = 2 \ldots m \), \( s = 1 \ldots k \):

\[
\pi[j, s] = \max_{s' \in \{1, \ldots, k\}} [\pi[j-1, s'] \times t(s'|s) \times e(x_j|s)]
\]

and

\[
bp[j, s] = \arg \max_{s' \in \{1, \ldots, k\}} [\pi[j-1, s'] \times t(s'|s) \times e(x_j|s)]
\]

- The \( bp \) entries are backpointers that will allow us to recover the identity of the highest probability state sequence

Viterbi Backpointers

- Most likely Sequence: \( s_0 \ s_1 \ s_2 \ \ldots \ s_2 \ s_F \)

Viterbi Backtrace

- Highest probability for any sequence of states is

\[
\max_s \pi[m, s]
\]

- To recover identity of highest-probability sequence:

\[
s_m = \arg \max_s \pi[m, s]
\]

and for \( j = m \ldots 2 \),

\[
s_{j-1} = bp[j, s_j]
\]

- The sequence of states \( s_1 \ldots s_m \) is then

\[
\arg \max_{s_1, \ldots, s_m} p(x_1 \ldots x_m, s_1 \ldots s_m; \theta)
\]
Homework

• Reading J&M Ch5.1-5.5, Ch6.1-6.5

• HMM notes

• Start thinking about course project and find a team.