Today

- Conditional models
- Trained discriminatively
- Motivated by the problem of word sense disambiguation

Objectives

- Review the idea behind joint, generative models, as well as their benefits and limitations.
- Discuss the challenge of word sense disambiguation and approach it as a classification problem.
- Explore knowledge sources (i.e., "features") that might help us do a better job of disambiguating word senses.
- Motivate the need for conditional models, trained discriminatively.
- Prepare to see maximum entropy models.

Back to Verb WSD

- Why are verbs harder?
  - Verbal senses are less topical
  - More sensitive to structure, argument choice

Verb Example: “Serve”

<table>
<thead>
<tr>
<th>Function</th>
<th>The tree stump serves as a table.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Means</td>
<td>The scandal served to increase its popularity.</td>
</tr>
<tr>
<td>Dish</td>
<td>We serve meals for the homeless.</td>
</tr>
<tr>
<td>Military</td>
<td>He served his country.</td>
</tr>
<tr>
<td>Jail</td>
<td>He served six years for embezzlement.</td>
</tr>
<tr>
<td>Tennis</td>
<td>It was Agassi’s turn to serve.</td>
</tr>
<tr>
<td>Legal</td>
<td>He was served by the sheriff.</td>
</tr>
</tbody>
</table>

Weighted Windows with NB

- Distance conditioning
  - Some words are important only when they are nearby
    
    \[
    P(c, w_{-k}, \ldots, w_0, w_1, \ldots, w_k) = P(c) \prod_{i=0}^k P(w_i | c, bin(i))
    \]
- Distance weighting
  - Nearby words should get a larger vote:
    
    \[
    P(c, w_{-k}, \ldots, w_0, w_1, \ldots, w_k) = P(c) \prod_{i=0}^k P(w_i | c)^{\text{bin}(i)}
    \]

Better Features

- There are smarter features:
  - Argument selectional preference:
    - serve NP(food) vs. serve NP(paper) vs. serve NP(country)
  - Subcategorization:
    - (function) serve PP(as)
    - (enable) serve VP[as]
    - (tennis) serve <transitive>
    - (food) serve NP[PP][is]
  - Can capture poorly (but robustly) with local windows
  - ... but we can also use a parser and get these features explicitly
  - Other constraints (Yarowsky, 95)
    - One-sense-per-discourse
      - only true for broad topical distinctions
    - One-sense-per-classification
      - pretty reliable when it kicks vs. manufacturing plant, flowering plant
Knowledge Sources

... point ... court ................. serve ........ game ...

- Can we use these knowledge sources in Naïve Bayes?

\[ P(c, w_1, w_2, ..., w_n) = P(c) \prod P(w_i | c) \]

Complex Features with Naïve Bayes?

- Example: Washington County jail served 11,166 meals last month - a figure that translates to feeding some 120 people three times daily for 31 days.
- So we have a decision to make based on a set of cues:
  - context: jail, context: county, context: feeding, ...
  - local-context: jail, local-context: meals
  - subcat: NP, direct-object-head: meals.
- Not conditionally independent, given class!
- Not clear how to build a generative derivation for these:
  - Choose topic, then decide on having a transitive usage, then pick "meals" to be the object's head, then generate other words?
- How about the words that appear in multiple features?
  - Hard to make this work (though maybe possible)
- No real reason to try

A Discriminative Approach

- View WSD as a discrimination task
- Use a conditional model:
  \[ P(\text{sense} \mid \text{context: jail, context: county, context: feeding, ... local-context: jail, local-context: meals subcat: NP, direct-object-head: meals, ...}) \]
- Have to estimate categorical dist. (over senses) where there are a huge number of things to condition on
- Many feature-based classification techniques exist
- We tend to need and prefer methods that provide distributions over classes (why?)

Feature Representations

\[ \{f_i(w)\} \]

Feature Food Jail Tennis
context:jail -0.5 * 1 1 +1.2 * 1 1 -0.8 * 1
subcat:NP +1.0 * 1 +1.0 * 1 +0.3 * 1
object-head:meals +2.0 * 1 -1.5 * 1 -1.5 * 1
object-head:years -1.8 * 0 +2.1 * 0 -1.1 * 0

TOTAL +3.5 +0.7 -2.6

Linear Classifiers

- For a pair \((c, w)\), we take a weighted vote for each class:
  \[ \text{vote}(c \mid w \text{ in context of document } d) = \sum_{i} \lambda_i(c) f_i(w) \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Food</th>
<th>Jail</th>
<th>Sens</th>
</tr>
</thead>
<tbody>
<tr>
<td>context: jail</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subcat: NP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>object-head:meals</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>object-head:years</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Linear Classifiers

- For a pair \((c, w)\), we take a weighted vote for each class:
  \[ \text{vote}(c \mid w \text{ in context of document } d) = \sum_{i} \lambda_i(c) f_i(w) \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Food</th>
<th>Jail</th>
<th>Sens</th>
</tr>
</thead>
<tbody>
<tr>
<td>context: jail</td>
<td>-0.5</td>
<td>+2.2</td>
<td>+0.8</td>
</tr>
<tr>
<td>subcat: NP</td>
<td>+1.0</td>
<td>+0.3</td>
<td>+1.5</td>
</tr>
<tr>
<td>object-head:meals</td>
<td>-2.0</td>
<td>+1.5</td>
<td>+1.5</td>
</tr>
<tr>
<td>object-head:years</td>
<td>-1.8</td>
<td>-2.7</td>
<td>-1.4</td>
</tr>
<tr>
<td>TOTAL</td>
<td>+3.5</td>
<td>+0.7</td>
<td>-2.6</td>
</tr>
</tbody>
</table>
Linear Classifiers

- For a pair \((c, w)\), we take a weighted vote for each class:
  \[
  \text{vote}(c \mid w \text{ in context of document } d) = \sum f_i(c) f_i(w)
  \]
- There are many ways to set these weights
  - Perceptron:
    - find a currently misclassified example
    - nudge weights in the direction of a correct classification
  - Other discriminative methods usually work in the same way:
    - try out various weights
    - until you maximize some objective that relies on the truth

Why Conditional Models?

Consider this metaphor:
Which would you rather take: a multiple-choice exam or an essay exam?
- Joint, generative models are like essay questions.
- Conditional models are like multiple-choice questions.
- Discuss!

Approaches to Deriving MaxEnt

- **Start with exponential form**
  - Maximize conditional data likelihood
  - Leads to constraints
  - Today’s lecture
- **Start with Occam’s Razor, Entropy, Constraints**
  - Employ the calculus of variations (Lagrange multipliers)
  - Leads to exponential form
  - Read the Berger tutorial, (optional reading)
- **Start with feed-forward neural network**
  - Maximize conditional data likelihood
  - An approximation of gradient descent
  - Or minimize cross-entropy
  - Can approximate training by stochastic gradient descent
- **Start with statistics and regression**
  - Leads to Logistic Regression

Next

- How to estimate those weights
- Maximum Entropy Models