Statistical NLP
Fall 2017

Lecture 5:
Advanced Part-of-Speech Tagging

Ankur Parikh – Google
Reminder: HMMs

\[ P(s, w) = \prod_i P(s_i | s_{i-1}) P(w_i | s_i) \]

- **transition matrix**
- **emission matrix**
(Vanilla) HMMs

- Easy to learn
  - Counting/smoothing

- Possible to find top scoring sequence efficiently

- Not easy to add arbitrary features

- Generative:
  - Explicitly models distribution over words
Adding Arbitrary Features

- “Featurize” emission matrix:

\[
P(w_i | s_i) = \frac{\exp(\sum_k \alpha_k f_k(w_i, s_i))}{\sum_w \exp(\sum_k \alpha_k f_k(w_i, s_i))}
\]

- Can also featuring transitions by splitting the state space

- Learning now requires gradient descent just like max-ent.
MEMMs

- Discriminative but locally normalized.

\[ P(s|w) = \prod_i P_{ME}(s_i|w, s_{i-1}, s_{i-2}) \]

max-ent (linear) model with lots of features
Label Bias Problem

Example from Ramesh Nallapati
When in $s_1$, it is more likely to go to $s_2$ than to stay in $s_1$

When in $s_2$ it is more likely to stay in $s_2$ than to go anywhere else
Label Bias Problem

- So we would think the most likely sequence is $s_1 \rightarrow s_2$ $\rightarrow s_2 \rightarrow s_2$
- But it is not
Label Bias Problem

\[ P(s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_2) = 0.6 \times 0.3 \times 0.3 = 0.054 \]
Label Bias Problem

\[ P(s_1 \rightarrow s_1 \rightarrow s_1 \rightarrow s_1) = 0.4 \times 0.45 \times 0.5 = 0.09 \]
Label Bias Problem

What happened?

\[ P(s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_2) = 0.6 \times 0.3 \times 0.3 = 0.054 \]

\[ P(s_1 \rightarrow s_1 \rightarrow s_1 \rightarrow s_1) = 0.4 \times 0.45 \times 0.5 = 0.09 \]
Label Bias Problem

\[ P(s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_2) = 0.6 \times 0.3 \times 0.3 = 0.054 \]

\[ P(s_1 \rightarrow s_1 \rightarrow s_1 \rightarrow s_1) = 0.4 \times 0.45 \times 0.5 = 0.09 \]

States fewer outgoing transitions are favored
Label Bias Problem

- Because the model is locally normalized sum of outgoing transitions must sum to one.

\[
P(s|w) = \prod_i P_{ME}(s_i|w, s_{i-1}, s_{i-2})
\]

- Favors states that only lead to a few other states.

- What types of POS tags do you think would be favored?
Each local factor is now unnormalized.
Global normalizer $Z$ (called partition function)
Conditional Random Fields

\[ P(s|w) = \frac{1}{Z(w)} \prod_i \psi(s_i, s_{i-1}, w) \]
Label Bias Problem Solved

Edge weights no longer need to sum to one.
CRFs: Compute Normalizer

- Normalizer is the sum over all possible states.

\[ Z(w) = \sum_{s_1, \ldots, s_n} \prod_{i} \psi(s_i | s_{i-1}, w) \]

- Brute force is exponential

- But can be computed efficiently for linear chain CRFs
**CRFs: Compute Normalizer**

- Basically same strategy as Viterbi algorithm (except sum instead of max)

\[
Z(w) = \sum_{s_1, \ldots, s_n} \prod_{i} \psi(s_i|s_{i-1}, w)
\]

\[
= \sum_{s_n, s_{n-1}} \psi(s_n|s_{n-1}, w) \left( \sum_{s_1, \ldots, s_{n-2}} \prod_{i-1} \psi(s_i|s_{i-1}, w) \right)
\]

**recursion**
CRFs: Compute Normalizer

- Dynamic programming

\[ \delta(s_1) = \psi(s_1, w) \]

\[ \delta(s_2) = \sum_{s_1} \psi(s_2, s_1, w) \delta(s_1) \]

\[ \cdots \]

\[ Z(w) = \sum_{s_n} \delta(s_n) \]
CRFs: Compute Normalizer

- Note that same strategy can be used to compute other probabilities as well e.g.

\[ P(s_i, s_{i-1} \mid w) \quad \forall i \]
Generative to Discriminative
CRFs: Most Likely Tag Sequence

- Normalizer is a constant so can be ignored

$$\arg\max_s P(s|w) = \arg\max_s \left( \frac{1}{Z(w)} \prod_i \psi(s_i, s_{i-1}, w) \right)$$

$$= \arg\max_s \prod_i \psi(s_i, s_{i-1}, w)$$

- Use Viterbi algorithm (just like HMMs)
CRFs: Parameter Learning

\[ \ell(\alpha) = \sum_{m=1}^{M} \log P(s^{(m)}|w^{(m)}, \alpha) \]

\[ = \sum_{m=1}^{M} \log \frac{\prod_i \exp \left( \sum_k \alpha_{ik} f_k(s_i^{(m)}, s_{i-1}^{(m)}, w^{(m)}) \right)}{Z(w^{(m)})} \]

\[ = \sum \sum \sum \sum \alpha_{ik} f_k(s_i^{(m)}, s_{i-1}^{(m)}, w^{(m)}) - \log Z(w^{(m)}) \]

\[ \frac{d \ell(\alpha)}{d \alpha_{jl}} = \sum_m f_l(s_j^{(m)}, s_{j-1}^{(m)}, w^{(m)}) - \frac{d}{d \alpha_{jl}} \log Z(w^{(m)}) \]
\[
\frac{d}{d \alpha_{jl}} \log Z(\mathbf{w}^{(m)}) = \frac{\frac{d}{d \alpha_{jl}} \left( \sum_{s_1, \ldots, s_n} \prod_i \exp \left( \sum_k \alpha_{ik} f_k(s_i, s_{i-1}, \mathbf{w}) \right) \right)}{Z(\mathbf{w}^{(m)})} \\
= f_l(s_j, s_{j-1}, \mathbf{w}) \sum_{s_1, \ldots, s_n} \prod_i \exp \left( \sum_k \alpha_{ik} f_k(s_i, s_{i-1}, \mathbf{w}) \right) Z(\mathbf{w}^{(m)}) \\
= f_l(s_j, s_{j-1}, \mathbf{w}) P(s^{(m)}|\mathbf{w}^{(m)}, \alpha)
\]
CRFs: Parameter Learning

\[
\frac{d \ell(\alpha)}{d \alpha_{jl}} = \sum_m f_l(s_j^{(m)}, s_{j-1}^{(m)}, w^{(m)}) - \frac{d}{d \alpha_{jl}} \log Z(w^{(m)})
\]

\[
= \sum_m f_l(s_j^{(m)}, s_{j-1}^{(m)}, w^{(m)}) - \sum_m f_l(s_j, s_{j-1}, w) P(s^{(m)}|w^{(m)}, \alpha)
\]

- Just like MaxEnt!
- Except computing the probability now requires running inference
# CRF Results: POS Tagging

<table>
<thead>
<tr>
<th>Method</th>
<th>Known words</th>
<th>Unknown words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary Lookup - Most frequent tag</td>
<td>~90%</td>
<td>~50%</td>
</tr>
<tr>
<td>Features only based on word in question</td>
<td>~93.7%</td>
<td>~82.6%</td>
</tr>
<tr>
<td>Trigram HMM</td>
<td>~95%</td>
<td>~55%</td>
</tr>
<tr>
<td>TNT (Trigram HMM++)</td>
<td>~97%</td>
<td>~86%</td>
</tr>
<tr>
<td>CRF</td>
<td>~97.2%</td>
<td>~89%</td>
</tr>
</tbody>
</table>
More Experiments

Rockwell International Corp.’s Tulsa unit said it signed a tentative agreement extending its contract with Boeing Co. to provide structural parts for Boeing’s 747 jetliners.

- “Shallow Parsing,” identify all noun phrases
- Good for Information Extraction tasks
- Typically use second order sequence models

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMM</td>
<td>93.70</td>
</tr>
<tr>
<td>Voted Perceptron</td>
<td>94.09</td>
</tr>
<tr>
<td>CRF</td>
<td>94.38</td>
</tr>
</tbody>
</table>
More Experiments

- Find “Names”
- Organizations, Locations, People, etc.
- Capitalization strong cue in English, but what about other languages?

<table>
<thead>
<tr>
<th>Language</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>75%-90%</td>
</tr>
<tr>
<td>German</td>
<td>60%-80%</td>
</tr>
</tbody>
</table>
Simple Neural Network Tagging Model

- No structure, just independent classifiers
How to add Structure?

- Search and Global Training (later) [Andor et al. 2016]

- Recurrence? - not really structured, but works well in practice (later) [Ling et al. 2015]
Generative models tend to have higher asymptotic error, but they approach their asymptotic error faster than discriminative ones with number of training examples.

Discriminative models make no independence assumptions for observations, and are therefore more flexible for incorporating overlapping features.

Training time for generative models usually much lower. Testing time comparable.
Conclusions

- Accuracies degrade outside of domain
  - Up to triple error rate
  - Usually make the most errors on the things you care about in the domain (e.g. protein names)

- Open questions
  - How to effectively exploit unlabeled data from a new domain (what could we gain?)
  - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)
Unsupervised Tagging

- Also known as part-of-speech induction

- Task:
  - Raw sentences in
  - Tagged sentences out

- Is this even possible?
First consider a simpler task, clustering sentences into categories.

\[ x_1 \quad \text{Soccer team wins match} \]
\[ x_2 \quad \text{Stocks close up 3\%} \]
\[ x_3 \quad \text{Investing in the stock market has yielded …} \]
\[ x_4 \quad \text{The first game of the world series was on the morning of …} \]
\[ x_5 \quad \text{In overtime, we won.} \]
\[ x_6 \quad \text{Tomorrow is the day for all the companies to report earnings and …} \]

Assign a cluster \( c_i \) to sentence \( x_i \)
Simpler Task: Sentence Clustering

- Probabilistic model for sentence clustering.
- Because the label isn’t observed, we have to marginalize over it.

\[ \ell(\theta) = \log \prod_{m=1}^{M} P(x^{(m)}|\theta) \]

\[ = \log \prod_{m=1}^{M} \sum_{c=1}^{C} P(x^{(m)}, c|\theta) \]

\[ = \sum_{m=1}^{M} \log \sum_{c=1}^{C} P(x^{(m)}, c|\theta) \]

*non-concave, has local optima*
# Challenges: Sentence Clustering

- Could be many types of clustering

## By length
- Soccer team wins match
- Stocks close up 3%
- Investing in the stock market has ...
- In overtime, we won.
- The first game of the world series ...
- Tomorrow is the day for all the companies to report earnings and ...  

## By topic
- Soccer team wins match
- Stocks close up 3%
- Investing in the stock market has ...
- In overtime, we won.
- The first game of the world series ...
- Tomorrow is the day for all the companies to report earnings and ...  

## By start of first letter
- Soccer team wins match
- Stocks close up 3%
- Investing in the stock market has ...
- In overtime, we won.
- The first game of the world series ...
- Tomorrow is the day for all the companies to report earnings and ...
Not a problem in supervised learning

- Supervised learning: We can specify lots of features and given enough data/regularization, the ones deemed irrelevant to the target class will get low feature weights.

- Unsupervised learning: The set of features specified can more drastically affect what is learned.
Key Challenge in Unsupervised Learning

- Higher likelihoods may be uncorrelated with higher accuracies on the task we care about.

- **Reason 1:** There may be multiple assignments to the latent variables that achieve the highest likelihood (non-identifiability)

- **Reason 2:** Even if the solution is unique it may not correspond well with the actual metric we care about.
### Key Challenge in Unsupervised Learning

- **Unsupervised parsing (a very hard task)**
Key Challenge in Unsupervised Learning

- Trying to design the objective such that higher likelihoods align closely with better latent assignments (POS tags in this case).

- Challenging, often takes lot of tweaking/engineering.

- Devil’s advocate: You could just spend the time you spent tweaking your unsupervised algorithm instead on labelling examples.
The Promise/Challenge of Unsupervised Learning

- Obtaining detailed annotation for linguistic phenomena (e.g. syntax trees) is difficult and cannot easily be gathered at scale.

- On the other hand, weakly labelled data is often more readily available.

- Unsupervised data is abundant.
Unsupervised POS Tagging

\[ P(s, w) = \prod_{i} P(s_i | s_{i-1}) P(w_i | s_i) \]

transition matrix

emission matrix
Unsupervised POS Tagging

\[ \ell(\theta) = \log \prod_{m=1}^{M} P(w^{(m)}|\theta) \]

\[ = \log \prod_{m=1}^{M} \sum_{s_1, \ldots, s_n} \prod_{i} P(s_i|s_{i-1}) P(w_i|s_i, \theta) \]

\[ = \sum_{m=1}^{M} \log \sum_{s_1, \ldots, s_n} \prod_{i} P(s_i|s_{i-1}) P(w_i|s_i, \theta) \]

non-concave, has local optima

number of training examples
Unsupervised POS Tagging

- Restrictions to improve unsupervised POS tagging
  - Restrict set of possible tags for a word using a dictionary.
  - Enforce soft constraints via posterior regularization [Ganchev et al. 2012]
    - each sentence is expected to have at least one verb
    - each sentence is expected to end in punctuation
Merialdo: Setup

- Some (discouraging) experiments [Merialdo 94]

**Setup:**

- You know the set of allowable tags for each word
- Fix k training examples to their true labels
  - Learn P(w|t) on these examples
  - Learn P(t|t_{-1},t_{-2}) on these examples
- On n examples, re-estimate with EM

- Note: we know allowed tags but not frequencies
Merialdo: Results

<table>
<thead>
<tr>
<th>Number of tagged sentences used for the initial model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Iter</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>
Some More Recent Results

Ammar, Dyer, Smith, 2014
Learning problem is non-convex. One strategy might be to alternate between following two steps:

- For a given sequence, compute the most likely parameters.

\[
\theta^{(t+1)} = \arg\max_\theta \log \prod_{m=1}^{M} \log P(w^{(m)}_t, s^{(m)}_t | \theta)
\]

- For a given set of parameters compute most likely sequence.

\[
s^{(m),t+1} = \arg\max_s \log \prod_{m=1}^{M} \log P(w^{(m)}_t, s | \theta^{(t)})
\]

- Each step either does nothing or increases the objective
Same intuition as strategy on previous slide, but more principled, and done in a soft way.

Given a posterior distribution over the hidden states i.e. $P(s \mid w)$, compute the best set of parameters.

For a given set of parameters compute the posterior distribution over the hidden states i.e. $P(s \mid w)$.

Guaranteed to reach local optima.
**Expectation Maximization**

\[
\ell(\theta) = \sum_{m=1}^{M} \log \sum_{s} P(w^{(m)}, s|\theta)
\]

\[
= \sum_{m=1}^{M} \log \sum_{s} Q^{(m)}(s) \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)}
\]

\[
\geq \sum_{m=1}^{M} \sum_{s} Q^{(m)}(s) \log \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)}
\]  

(Jensen’s inequality)

some distribution over the states for a given example
Expectation Maximization

\[ \tilde{\ell}(\theta) = \sum_{m=1}^{M} \sum_{s} Q^{(m)}(s) \log \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)} \]

Alternate between maximizing:
1. \( Q \) given fixed \( \theta \) (E-step)
2. \( \theta \) given fixed \( Q \) (M-step)

Example of a coordinate ascent optimization algorithm.
EM (M-step)

\[
\text{argmax}_\theta \sum_{m=1}^{M} \sum_{s} Q^{(m),t}(s) \log \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)}
\]

\[
= \text{argmax}_\theta \sum_{m=1}^{M} \sum_{s} Q^{(m),t}(s) \log P(w^{(m)}, s|\theta) - \sum_{m=1}^{M} \sum_{s} Q^{(m),t} \log Q^{(m)}(s)
\]

weighted version of likelihood, concave

\[\text{a constant in terms of } \theta\]
EM (E-step)

\[
\text{argmax}_Q \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log \frac{P(w^{(m)}, s|\theta^{(t)})}{Q^{(m)}(s)}
\]

\[
= \text{argmax}_Q \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log \frac{P(w^{(m)}|\theta^{(t)}) P(s|w^{(m)}, \theta^{(t)})}{Q^{(m)}(s)}
\]

\[
= \text{argmax}_Q \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log P(w^{(m)}|\theta^{(t)}) + \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log \frac{P(s|w^{(m)}, \theta^{(t)})}{Q^{(m)}(s)}
\]

\[
= \text{argmax}_Q \sum_{m=1}^{M} P(w^{(m)}|\theta^{(t)}) - \sum_{m=1}^{M} \text{KL}(Q^{(m)}(s)||P(s|w^{(m)}))
\]

KL divergence is non-negative. We can minimize it by setting

\[
Q^{(m)}(s) = P(s|x^{(m)})
\]
EM

E-step: Set Q to posterior distribution

\[ Q^{(m)}(s) = P(s|w^{(m)}) \]

M-step: Maximize weighted likelihood

\[ \arg\max_{\theta} \sum_{m=1}^{M} \sum_{s} Q^{(m),t}(s) \log P(w^{(m)}, s|\theta) \]
**EM**

E-step: Set $Q$ to posterior distribution

$$Q^{(m)}(s) = P(s | w^{(m)})$$

M-step: Maximize weighted likelihood

$$\arg\max_{\theta} \sum_{m=1}^{M} \sum_s Q^{(m),t}(s) \log P(w^{(m)}, s | \theta)$$

we can decompose this efficiently using the structure of the HMM