Mini 1 graded by next lecture

Project 1 is out, sample writeups on website
Recall: HMMs

- **Input** \( x = (x_1, \ldots, x_n) \)  
  **Output** \( y = (y_1, \ldots, y_n) \)

\[
P(y, x) = P(y_1) \prod_{i=2}^{n} P(y_i | y_{i-1}) \prod_{i=1}^{n} P(x_i | y_i)
\]

- **Training**: maximum likelihood estimation (with smoothing)

\[
score_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) score_{i-1}(y_{i-1})
\]

- **Inference problem**: \( \arg\max_y P(y | x) = \arg\max_y \frac{P(y, x)}{P(x)} \)

- **Viterbi**:
This Lecture

- CRFs: model (+features for NER), inference, learning
- Named entity recognition (NER)
- (if time) Beam search
Named Entity Recognition

- BIO tagset: begin, inside, outside
- Sequence of tags — should we use an HMM?
- Why might an HMM not do so well here?
  - Lots of O’s, so tags aren’t as informative about context
  - Insufficient features/capacity with multinomials (especially for unks)

Barack Obama will travel to Hangzhou today for the G20 meeting.
CRFs
Conditional Random Fields

- HMMs are expressible as Bayes nets (factor graphs)

\[ y_1 \rightarrow y_2 \rightarrow \ldots \rightarrow y_n \]

\[ x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n \]

- This reflects the following decomposition:

\[ P(y, x) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \ldots \]

- Locally normalized model: each factor is a probability distribution that normalizes
Conditional Random Fields

- HMMs: \( P(y, x) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \ldots \)

- CRFs: discriminative models with the following globally-normalized form:
  \[
P(y|x) = \frac{1}{Z} \prod_k \exp(\phi_k(x, y))
\]

  normalizer

  any real-valued scoring function of its arguments

- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization \(\leftrightarrow\) generative vs. discriminative

- Locally normalized discriminative models do exist (MEMMs)

- How do we max over \(y\)? Intractable in general — can we fix this?
**Sequential CRFs**

- **HMMs:** \( P(y, x) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \ldots \)

- **CRFs:**

\[
P(y|x) \propto \prod_k \exp(\phi_k(x, y))
\]

\[
P(y|x) \propto \exp(\phi_o(y_1)) \prod_{i=2}^n \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^n \exp(\phi_e(x_i, y_i))
\]

- HMMs:

- CRFs:
Sequential CRFs

\[
P(y|x) \propto \exp(\phi_o(y_1)) \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(x_i, y_i))
\]

- We condition on \(x\), so every factor can depend on all of \(x\) (including transitions, but we won’t do this)
- \(y\) can’t depend arbitrarily on \(x\) in a generative model

Token index — lets us look at current word
- Notation: omit $x$ from the factor graph entirely (implicit)
- Don’t include initial distribution, can bake into other factors

Sequential CRFs:

$$P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x))$$
Feature Functions

\[
P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x))
\]

- This can be almost anything! Here we use linear functions of sparse features

\[
\begin{align*}
\phi_e(y_i, i, x) &= w^\top f_e(y_i, i, x) \\
\phi_t(y_{i-1}, y_i) &= w^\top f_t(y_{i-1}, y_i)
\end{align*}
\]

\[
P(y|x) \propto \exp w^\top \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right]
\]

- Looks like our single weight vector multiclass logistic regression model
Basic Features for NER

\[ P(y|x) \propto \exp w^\top \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right] \]

Barack Obama will travel to Hangzhou today for the G20 meeting.

Transitions: \( f_t(y_{i-1}, y_i) = \text{Ind}[y_{i-1} \& y_i] = \text{Ind}[O \rightarrow \text{B-LOC}] \)

Emissions: \( f_e(y_6, 6, x) = \text{Ind}[\text{B-LOC} \& \text{Current word = Hangzhou}] \)
\( \text{Ind}[\text{B-LOC} \& \text{Prev word = to}] \)
Features for NER

Leicestershire is a nice place to visit...

I took a vacation to Boston

Apple released a new version...

According to the New York Times...

Leonardo DiCaprio won an award...

Texas governor Greg Abbott said
Features for NER

- Word features (can use in HMM)
  - Capitalization
  - Word shape
  - Prefixes/suffixes
  - Lexical indicators
- Context features (can’t use in HMM!)
  - Words before/after
  - Tags before/after
- Word clusters
- Gazetteers

According to the *New York Times*...

Apple released a new version...
CRFs Outline

- Model: 
  \[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]
  \[ P(y|x) \propto \exp w^\top \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right] \]

- Inference

- Learning
Computing (arg)maxes

\[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]

- **argmax\_y P(y|x):** can use Viterbi exactly as in HMM case

\[
\max_{y_1, \ldots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, x)} \cdots e^{\phi_e(y_2, 2, x)} e^{\phi_t(y_1, y_2)} e^{\phi_e(y_1, 1, x)}
\]

\[
= \max_{y_2, \ldots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, x)} \cdots e^{\phi_e(y_2, 2, x)} \max_{y_1} e^{\phi_t(y_1, y_2)} e^{\phi_e(y_1, 1, x)}
\]

\[
= \max_{y_3, \ldots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_e(y_n, n, x)} \cdots \max_{y_2} e^{\phi_t(y_2, y_3)} e^{\phi_e(y_2, 2, x)} \max_{y_1} e^{\phi_t(y_1, y_2)} \text{score}_1(y_1)
\]

- \( \exp(\phi_t(y_{i-1}, y_i)) \) and \( \exp(\phi_e(y_i, i, x)) \) play the role of the Ps now, same dynamic program
Can do inference in any tree-structured CRF

Max-product algorithm: generalization of Viterbi to arbitrary tree-structured graphs (sum-product is generalization of forward-backward)
CRFs Outline

Model: $$P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x))$$

$$P(y|x) \propto \exp w^\top \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right]$$

Inference: argmax $$P(y|x)$$ from Viterbi

Learning
Training CRFs

\[ P(y|x) \propto \exp w^\top \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right] \]

- Logistic regression: \( P(y|x) \propto \exp w^\top f(x, y) \)
- Maximize \( \mathcal{L}(y^*, x) = \log P(y^*|x) \)
- Gradient is completely analogous to logistic regression:
  \[
  \frac{\partial}{\partial w} \mathcal{L}(y^*, x) = \sum_{i=2}^{n} f_t(y^*_{i-1}, y^*_i) + \sum_{i=1}^{n} f_e(y^*_i, i, x)
  \]
  \[
  -\mathbb{E}_y \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right]
  \]
  intractable!
Training CRFs

\[
\frac{\partial}{\partial w} \mathcal{L}(y^*, x) = \sum_{i=2}^{n} f_t(y^*_{i-1}, y^*_i) + \sum_{i=1}^{n} f_e(y^*_i, i, x) \\
- \mathbb{E}_y \left[ \sum_{i=2}^{n} f_t(y^*_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right]
\]

» Let’s focus on emission feature expectation

\[
\mathbb{E}_y \left[ \sum_{i=1}^{n} f_e(y_i, i, x) \right] = \sum_{y \in \mathcal{Y}} P(y|x) \left[ \sum_{i=1}^{n} f_e(y_i, i, x) \right] = \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} P(y|x) f_e(y_i, i, x)
\]

\[
= \sum_{i=1}^{n} \sum_{s} P(y_i = s|x) f_e(s, i, x)
\]
Computing Marginals

\[
P(y|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))
\]

- Normalizing constant \( Z = \sum_{\mathbf{y}} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x})) \)

- Analogous to \( P(\mathbf{x}) \) for HMMs

- For both HMMs and CRFs:

\[
P(y_i = s|\mathbf{x}) = \frac{\text{forward}_i(s) \text{backward}_i(s)}{\sum_{s'} \text{forward}_i(s') \text{backward}_i(s')}
\]
Posteriors vs. Probabilities

\[ P(y_i = s | x) = \frac{\text{forward}_i(s) \cdot \text{backward}_i(s)}{\sum_{s'} \text{forward}_i(s') \cdot \text{backward}_i(s')} \]

- Posterior is *derived* from the parameters and the data (conditioned on \( x \)!)  

\[
\begin{align*}
P(x_i | y_i), P(y_i | y_{i-1}) & \quad \text{HMM} \\
\text{Model parameter (usually multinomial distribution)} & \\
\text{Inferred quantity from forward-backward} & \end{align*}
\]

\[
\begin{align*}
P(y_i | \mathbf{x}), P(y_{i-1}, y_i | \mathbf{x}) & \quad \text{CRF} \\
\text{Undefined (model is by definition conditioned on } \mathbf{x}) & \\
\text{Inferred quantity from forward-backward} & \end{align*}
\]
Training CRFs

- For emission features:
  \[
  \frac{\partial}{\partial w} \mathcal{L}(y^*, x) = \sum_{i=1}^{n} f_e(y^*_i, i, x) - \sum_{i=1}^{n} \sum_{s} P(y_i = s | x) f_e(s, i, x)
  \]
  gold features — expected features under model

- Transition features: need to compute \( P(y_i = s_1, y_{i+1} = s_2 | x) \)
  using forward-backward as well

- ...but you can build a pretty good system without transition features
CRFs Outline

- **Model:**
  \[
P(y \mid x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x))
  \]

  \[
P(y \mid x) \propto \exp w^\top \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right]
  \]

- **Inference:** \( \text{argmax } P(y \mid x) \) from Viterbi

- **Learning:** run forward-backward to compute posterior probabilities; then

  \[
  \frac{\partial}{\partial w} \mathcal{L}(y^*, x) = \sum_{i=1}^{n} f_e(y^*_i, i, x) - \sum_{i=1}^{n} \sum_{s} P(y_i = s \mid x) f_e(s, i, x)
  \]
Pseudocode

for each epoch
  for each example
    extract features on each emission and transition (look up in cache)
    compute potentials phi based on features + weights
    compute marginal probabilities with forward-backward
    accumulate gradient over all emissions and transitions
Implementation Tips for CRFs

- Caching is your friend! Cache feature vectors especially

- Try to reduce redundant computation, e.g. if you compute both the gradient and the objective value, don’t rerun the dynamic program

- Exploit sparsity in feature vectors where possible, especially in feature vectors and gradients

- Do all dynamic program computation in log space to avoid underflow

- If things are too slow, run a profiler and see where time is being spent. Forward-backward should take most of the time
Debugging Tips for CRFs

- Hard to know whether inference, learning, or the model is broken!
- Compute the objective — is optimization working?
  - **Inference**: check gradient computation (most likely place for bugs)
    - Is $\sum_s \text{forward}_i(s) \cdot \text{backward}_i(s)$ the same for all $i$?
    - Do probabilities normalize correctly + look “reasonable”? (Nearly uniform when untrained, then slowly converging to the right thing)
  - **Learning**: is the objective going down? Can you fit a small training set? Are you applying the gradient correctly?
- If objective is going down but model performance is bad:
  - **Inference**: check performance if you decode the training set
CRF with lexical features can get around 85 F1 on this problem

Other pieces of information that many systems capture

World knowledge:

The delegation met the president at the airport, *Tanjug* said.

**Tanjug**

*From Wikipedia, the free encyclopedia*

*Tanjug* (/tǎnˈd͡ʒʊɡ/; *Serbian Cyrillic: Танђуљ*) is a Serbian state news agency based in Belgrade.[2]
Nonlocal Features

The news agency Tanjug reported on the outcome of the meeting. The delegation met the president at the airport, Tanjug said.

- More complex factor graph structures can let you capture this, or just decode sentences in order and use features on previous sentences.

Finkel and Manning (2008), Ratinov and Roth (2009)
Semi-Markov Models

Barack Obama will travel to Hangzhou today for the G20 meeting.

- Chunk-level prediction rather than token-level BIO
- $y$ is a set of touching spans of the sentence
- Pros: features can look at whole span at once
- Cons: there’s an extra factor of $n$ in the dynamic programs

Evaluating NER

Prediction of all Os still gets 66% accuracy on this example!

What we really want to know: how many named entity chunk predictions did we get right?

Precision: of the ones we predicted, how many are right?

Recall: of the gold named entities, how many did we find?

F-measure: harmonic mean of these two
### How well do NER systems do?

<table>
<thead>
<tr>
<th>System</th>
<th>Resources Used</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ LBJ-NER</td>
<td>Wikipedia, Nonlocal Features, Word-class Model</td>
<td>90.80</td>
</tr>
<tr>
<td>- (Suzuki and Isozaki, 2008)</td>
<td>Semi-supervised on 1G-word unlabeled data</td>
<td>89.92</td>
</tr>
<tr>
<td>- (Ando and Zhang, 2005)</td>
<td>Semi-supervised on 27M-word unlabeled data</td>
<td>89.31</td>
</tr>
<tr>
<td>- (Kazama and Torisawa, 2007a)</td>
<td>Wikipedia</td>
<td>88.02</td>
</tr>
<tr>
<td>- (Krishnan and Manning, 2006)</td>
<td>Non-local Features</td>
<td>87.24</td>
</tr>
<tr>
<td>- (Kazama and Torisawa, 2007b)</td>
<td>Non-local Features</td>
<td>87.17</td>
</tr>
<tr>
<td>+ (Finkel et al., 2005)</td>
<td>Non-local Features</td>
<td>86.86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lample et al. (2016)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-CRF (no char)</td>
<td>90.20</td>
</tr>
<tr>
<td>LSTM-CRF</td>
<td>90.94</td>
</tr>
<tr>
<td>S-LSTM (no char)</td>
<td>87.96</td>
</tr>
<tr>
<td>S-LSTM</td>
<td>90.33</td>
</tr>
<tr>
<td>BiLSTM-CRF + ELMo</td>
<td>92.2</td>
</tr>
</tbody>
</table>

Ratinov and Roth (2009)
Beam Search
Fed raises interest rates 0.5 percent

- n word sentence, s tags to consider — what is the time complexity?

- $O(ns^2)$ — s is ~40 for POS, n is ~20
Many tags are totally implausible

Can any of these be:

- Determiners?
- Prepositions?
- Adjectives?

Features quickly eliminate many outcomes from consideration — don’t need to consider these going forward

Fed raises interest rates 0.5 percent
**Beam Search**

- Maintain a beam of $k$ plausible states at the current timestep
- Expand all states, only keep $k$ top hypotheses at new timestep
- Beam size of $k$, time complexity $O(nks \log(ks))$

---

**Diagram and Table**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBD</td>
<td>+1.2</td>
</tr>
<tr>
<td>NNP</td>
<td>+0.9</td>
</tr>
<tr>
<td>VBN</td>
<td>+0.7</td>
</tr>
<tr>
<td>NN</td>
<td>+0.3</td>
</tr>
<tr>
<td>VBZ</td>
<td>+1.2</td>
</tr>
<tr>
<td>NNS</td>
<td>-1.0</td>
</tr>
<tr>
<td>DT</td>
<td>-5.3</td>
</tr>
<tr>
<td>PRP</td>
<td>-5.8</td>
</tr>
</tbody>
</table>

- Not expanded: Fed raises
- Maintained priority queue to efficiently add things
How good is beam search?

- $k=1$: greedy search

- Choosing beam size:
  - 2 is usually better than 1
  - Usually don’t use larger than 50
  - Depends on problem structure

- If beam search is much faster than computing full sums, can use structured SVM instead of CRFs, but we won’t discuss that here
Next Time

- Neural networks