CS388: Natural Language Processing

Lecture 5: Named Entity Recognition, CRFs

Greg Durrett
Mini 1 grading underway

Project 1 due next Thursday
Recall: HMMs

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

- **Training**: maximum likelihood estimation (count + normalize)

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

- **Viterbi**: $\text{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}) P(x_i|s) \text{score}_{i-1}(y_{i-1})$
Recall: Viterbi Algorithm

- Compute scores for next timestep (score of optimal tag sequence ending with tag $i$ at timestep $t$)
Viterbi/HMMs: Other Resources

- Lecture notes from my undergrad course (posted online)
  
  - We ignore the STOP token here. It’s not in the tag set and just don’t use these probabilities

- Eisenstein Chapter 7.3 **but** the notation covers a more general case than what’s discussed for HMMs

- Jurafsky+Martin 8.4.5
This Lecture

- Conditional random fields
- Features for NER
- Inference and Learning in CRFs
- Next time: finish up NER systems
Named Entity Recognition

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

- 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
  - Insufficient features/capacity with multinomials (especially for unks)
HMMs Pros and Cons

- Big advantage: transitions, scoring pairs of adjacent y’s

- Big downside: not able to incorporate useful word context information

- Solution: switch from generative to discriminative model (conditional random fields) so we can condition on the entire input.

- Conditional random fields: logistic regression + features on pairs of y’s
Conditional Random Fields
Conditional Random Fields

- Flexible discriminative model for tagging tasks that can use arbitrary features of the input. Similar to logistic regression, but *structured*

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

- `Curr_word=Barack & Label=B-PER`
- `Next_word=Obama & Label=B-PER`
- `Curr_word_starts_with_capital=True & Label=B-PER`
- `Posn_in_sentence=1st & Label=B-PER`
- `Label=B-PER & Next-Label = I-PER`

...
Tagging with Logistic Regression

- Logistic regression over each tag individually:

$$P(y_i = y | x, i) = \frac{\exp(w^T f(y, i, x))}{\sum_{y' \in Y} \exp(w^T f(y', i, x))}$$

Probability of the $i$th word getting assigned tag $y$ (B-PER, etc.)
Tagging with Logistic Regression

- Logistic regression over each tag individually: “different features” approach to features for a single tag

\[
P(y_i = y | x, i) = \frac{\exp(w^\top f(y, i, x))}{\sum_{y' \in Y} \exp(w^\top f(y', i, x))}
\]

- Over all tags:

\[
P(y = \tilde{y} | x) = \prod_{i=1}^{n} P(y_i = \tilde{y}_i | x, i) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^\top f(\tilde{y}_i, i, x) \right)
\]

- Score of a prediction: sum of weights dot features over each individual predicted tag (this is a simple CRF but not the general form)

- Set Z equal to the product of denominators; we’ll discuss this in a few slides

- Conditional model: \(x\) is observed, \(y\) isn’t
Example: “Emission Features” $f_e$

$$
\text{feats} = f_e(\text{B-PER, } i=1, \mathbf{x}) + f_e(\text{I-PER, } i=2, \mathbf{x}) + f_e(\text{O, } i=3, \mathbf{x}) + f_e(\text{O, } i=4, \mathbf{x})
$$
Adding Structure

\[
P(y = \tilde{y} | x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^\top f(\tilde{y}_i, i, x) \right)
\]

- We want to be able to learn that some tags don’t follow other tags — want to have features on tag pairs

\[
P(y = \tilde{y} | x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^\top f_e(\tilde{y}_i, i, x) + \sum_{i=2}^{n} w^\top f_t(\tilde{y}_{i-1}, \tilde{y}_i, i, x) \right)
\]

- Score: sum of weights dot \( f_e \) features over each predicted tag ("emissions") plus sum of weights dot \( f_t \) features over tag pairs ("transitions")

- This is a sequential CRF
Example

Barack Obama will travel

feats = \( f_e(B-PER, i=1, x) + f_e(I-PER, i=2, x) + f_e(O, i=3, x) + f_e(O, i=4, x) \)
+ \( f_t(B-PER, I-PER, i=1, x) + f_t(I-PER, O, i=2, x) + f_t(O, O, i=3, x) \)

Obama can start a new named entity (emission feats look okay), but we’re not likely to have two PER entities in a row (transition feats)
Sequential CRFs

\[
P(y = \tilde{y} | x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{n} w^\top f_e(\tilde{y}_i, i, x) + \sum_{i=2}^{n} w^\top f_t(\tilde{y}_{i-1}, \tilde{y}_i, i, x) \right)
\]

- Critical property: this structure is going to allow us to use dynamic programming (Viterbi) to sum or max over all sequences
- How does this compare to HMMs?
HMMs vs. CRFs

Both models are expressible in different factor graph notation

- Phis are “potentials”, used in the general CRF formulation
HMMs vs. CRFs

- HMMs: in the standard HMM, emissions consider one word at a time

- CRFs support features over many words simultaneously, non-independent features (e.g., suffixes and prefixes), not generative models

- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> generative vs. discriminative
  (locally normalized discriminative models do exist (MEMMs))
CRFs in General

- CRFs: discriminative model with the following form:

\[ P(y|x) = \frac{1}{Z} \prod_k \exp(\phi_k(x, y)) \]

\( Z \) is the normalizer, any real-valued scoring function of its arguments

- Our special case: linear feature-based potentials \( \phi_k(x, y) = w^\top f_k(x, y) \)

\[ P(y|x) = \frac{1}{Z} \exp \left( \sum_{k=1}^{n} w^\top f_k(x, y) \right) \]

- Problem: intractable inference in the general case! Computing \( Z \) requires an exponent sum
Features for NER
Basic Features for NER

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

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

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

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

\( \text{Ind}[\text{B-LOC} \& \text{Prev word} = \text{to}] \)
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

\[ \phi_e(y_i, i, x) \]
Emission 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...
Leicestershire
Boston
CRFs Outline

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

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

- **Inference**

- **Learning**
Inference and Learning in CRFs
Computing (arg)maxes

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

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

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

- \( \exp(\phi_t(y_{i-1}, y_i)) \) and \( \exp(\phi_e(y_i, i, x)) \) play the role of the Ps now, use the exact same Viterbi dynamic program
Inference in General CRFs

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

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

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: \( P(y|x) \propto \exp w^\top f(x, y) \)
- For CRFs: maximize \( \mathcal{L}(y^*, x) = \log P(y^*|x) \)
- Gradient is analogous to logistic regression: gold feats — expected feats

\[
\frac{\partial}{\partial w} \mathcal{L}(y^*, x) = \sum_{i=2}^{n} f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} f_e(y_i^*, i, x)
\]

\[
-\mathbb{E}_y \left[ \sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, x) \right]
\]

intractable!
Training CRFs

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

- Let’s focus on emission feature expectation

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

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

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

sum over timesteps

marginal probability

sum over tags

sum over tags of that tag at that step
Forward-Backward Algorithm

- How do we compute these marginals $P(y_i = s|x)$?

$$P(y_i = s|x) = \sum_{y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n} P(y|x)$$

- What did Viterbi compute?

$$P(y_{\text{max}}|x) = \max_{y_1, \ldots, y_n} P(y|x)$$

- Can compute marginals with dynamic programming as well using forward-backward
Forward-Backward Algorithm

\[ P(\mathbf{y}_3 = 2|\mathbf{x}) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}} \]
Forward-Backward Algorithm

\[ P(y_3 = 2|x) = \]
\[ \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}} \]

- Easiest and most flexible to do one pass to compute \( P(y_3 = 2|x) \) and one to compute

Slide credit: Dan Klein
Forward-Backward Algorithm

- Initial:
  \[ \alpha_1(s) = \exp(\phi_e(s, 1, x)) \]

- Recurrence:
  \[ \alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \exp(\phi_e(s_t, t, x)) \exp(\phi_t(s_{t-1}, s_t)) \]

- Same as Viterbi but summing instead of maxing!

- These quantities get very small!
  Store everything as log probabilities
Forward-Backward Algorithm

- **Initial:**
  \[ \beta_n(s) = 1 \]

- **Recurrence:**
  \[ \beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1}, t + 1, x)) \exp(\phi_t(s_t, s_{t+1})) \]

- Big differences: count emission for the next timestep (not current one)
Forward-Backward Algorithm

\[ \alpha_1(s) = \exp(\phi_e(s, 1, x)) \]
\[ \alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \exp(\phi_e(s_t, t, x)) \exp(\phi_t(s_{t-1}, s_t)) \]
\[ \beta_n(s) = 1 \]
\[ \beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1}, t + 1, x)) \exp(\phi_t(s_t, s_{t+1})) \]
\[ P(s_3 = 2 | x) = \frac{\alpha_3(2)\beta_3(2)}{\sum_i \alpha_3(i)\beta_3(i)} \]

- Does this explain why beta is what it is?
- What does the denominator here mean?
Computing Marginals

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

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

- Analogous to \( P(x) \) for HMMs

- For both HMMs and CRFs:

\[
P(y_i = s|x) = \frac{\text{forward}_i(s) \cdot \text{backward}_i(s)}{\sum_{s'} \text{forward}_i(s') \cdot \text{backward}_i(s')} \quad \text{Z for CRFs, } P(x) \text{ for HMMs}
\]
Posteriors vs. Probabilities

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

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

\[
P(x_i|y_i), P(y_i|y_{i-1}) \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad P(y_i|x), P(y_{i-1}, y_i|x)
\]

<table>
<thead>
<tr>
<th>Model</th>
<th>Inferred quantity from</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>\text{forward-backward}</td>
</tr>
<tr>
<td>CRF</td>
<td>\text{forward-backward}</td>
</tr>
</tbody>
</table>

- Inferred quantity from forward-backward
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 learned transition features (use heuristic weights, or just enforce constraints like B-PER -> I-ORG is illegal)
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: \( \arg\max P(y|x) \) from Viterbi

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

\[
\frac{\partial}{\partial w} \mathcal{L}(y^*, x) = \sum_{i=1}^{n} f_e(y^*_i, i, x) - \sum_{i=1}^{n} \sum_{s} P(y_i = s|x) f_e(s, i, x)
\]
Pseudocode and Tips
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 $\sum_i^{\text{forward}_i(s)\text{backward}_i(s)}$ 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? Try to fit 1 example / 10 examples. 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
Next Time

- Finish discussing NER
- Neural networks