CS395T: Structured Models for NLP
Lecture 5: Sequence Models II

Greg Durrett

Some slides adapted from Dan Klein, UC Berkeley
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)

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

- Exponentially many possible \( y \) here!

- Viterbi: \( \text{score}_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) \text{score}_{i-1}(y_{i-1}) \)
This Lecture

- Generative vs. discriminative models
- CRFs for sequence modeling
- Named entity recognition (NER)
- Structured SVM
- (if time) Beam search
Barack Obama will travel to Hangzhou today for the G20 meeting.

- BIO tagset: begin, inside, outside
- POS tagging is a plausible generative model of language — NER with this vanilla tag set is not
- What’s different about modeling $P(y|x)$ directly vs. $P(x,y)$ and computing the posterior later?
Generative vs. Discriminative Models

Reality

<table>
<thead>
<tr>
<th>Lights Working</th>
<th>Lights Broken</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(g,r,w) = 3/7</td>
<td>P(r,g,w) = 3/7</td>
</tr>
<tr>
<td>P(r,r,w) = 3/7</td>
<td>P(r,r,b) = 1/7</td>
</tr>
</tbody>
</table>

slide credit: Dan Klein
Generative vs. Discriminative Models

Reality

Lights Working

P(g,r,w) = 3/7
P(r,g,w) = 3/7

Lights Broken

P(r,r,b) = 1/7

NB Model

Working?

NS  EW

NB FACTORS:

- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2

- P(b) = 1/7
- P(r|b) = 1
- P(g|b) = 0

slide credit: Dan Klein
Generative vs. Discriminative Models

What does the model say when both lights are red?

- \( P(b,r,r) = (1/7)(1)(1) = 1/7 = 4/28 \)
- \( P(w,r,r) = (6/7)(1/2)(1/2) = 6/28 = 6/28 \)
- \( P(w|r,r) = 6/10! \)

Lights are working — wrong!
Generative vs. Discriminative Models

- What if P(b) were 1/2 instead of 1/7 (the NB estimate)?
  - P(b,r,r) = (1/2)(1)(1) = 1/2 = 4/8
  - P(w,r,r) = (1/2)(1/2)(1/2) = 1/8 = 1/8
  - P(w | r,r) = 1/5! ▶ Lights are broken — correct! Data likelihood is lower but
- Data likelihood P(x,y) is lower but posterior P(y|x) is more accurate

slide credit: Dan Klein
Conditional Random Fields

- HMMs are expressible as Bayes nets (factor graphs)

\[ y_1 \rightarrow y_2 \rightarrow \ldots \rightarrow y_n \]
\[ x_1 \rightarrow x_2 \rightarrow x_n \]

- 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 <-> generative vs. discriminative

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

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

- CRFs:
  \[
P(\mathbf{y}|\mathbf{x}) \propto \prod_k \exp(\phi_k(\mathbf{x}, \mathbf{y}))
  \]
  \[
P(\mathbf{y}|\mathbf{x}) \propto \exp(\phi_o(y_1)) \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(x_i, y_i))
  \]
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 variable can depend on all of \( x \)
- \( x \) can’t depend arbitrarily on \( y \) in a generative model — would make inference hard
...in fact, we typically don’t show $x$ at all

Don’t include initial distribution, can bake into other factors

Sequential CRFs:

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

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

- \( \exp(\phi_t(y_{i-1}, y_i)) \) and \( \exp(\phi_e(y_i, i, x)) \) play the role of the Ps now, same dynamic program
\[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]

- 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:

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

\( P(y_i = s, x) \) for HMMs; sums out other \( y \)s

\( Z \) for CRFs, \( P(x) \) for HMMs
Inference in General CRFs

- Can do inference in any tree-structured CRF
- Sum-product algorithm: generalization of forward-backward to arbitrary tree-structured graphs
- We’ll come back to this in a few lectures when we deal with other kinds of graphs
Feature Functions

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

- Phi can have sophisticated features! Generally look like linear models

\[
\phi_e(y_i, i, x) = w^\top f_e(y_i, i, x) \quad \phi_t(y_{i-1}, y_i) = w^\top f_t(y_{i-1}, y_i)
\]

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

- Log-linear model — structurally like logistic regression!
Training CRFs

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

- Assume \( \phi_t \) and \( \phi_e \) are both linear feature functions \( w^Tf(\text{args}) \)

\[ \mathcal{L}(y^*, \mathbf{x}) = \log P(y^*|\mathbf{x}) = \sum_{i=2}^{n} w^T f_t(y^*_{i-1}, y^*_i) + \sum_{i=1}^{n} w^T f_e(x_i, y^*_i) - \log Z \]

- Gradient is gold features minus expected features under model, like in LR

\[
\frac{\partial}{\partial w_j} \mathcal{L}(y^*, \mathbf{x}) = \sum_{i=2}^{n} f_{t,j}(y^*_{i-1}, y^*_i) + \sum_{i=1}^{n} f_{e,j}(x_i, y^*_i) \\
- \mathbb{E}_y \left[ \sum_{i=2}^{n} f_{t,j}(y_{i-1}, y_i) + \sum_{i=1}^{n} f_{e,j}(x_i, y_i) \right]
\]
Training CRFs

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

- How to compute expectations?
- Forward-backward helps you compute \( P(y_i = s|\mathbf{x}) \)
- Take weighted sum over all features at all tags and positions
- Transition features: need to compute \( P(y_i = s_1, y_{i+1} = s_2|\mathbf{x}) \) using forward-backward as well
- ...but you can build a pretty good system without transition features
Implementation Tips

- Often many features but only a few are active on a single sentence even across many different labels

- Maintain the gradient as a sparse vector for efficiency

  - Counter in `utils.py` is a way to do this
Basic Features for NER

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

Barack Obama will travel to \textcolor{orange}{Hangzhou} today for the G20 meeting.

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

Emissions: \( f_e(y_6, 6, x) = \text{Ind}[\text{B-LOC} \& \text{Current word = Hangzhou}] \)
\( \text{Ind}[\text{B-LOC} \& \text{Prev word = to}] \)
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
  - Capitalization
  - Word shape
  - Prefixes/suffixes
  - Lexical indicators
- Context features
  - Words before/after
  - Tags before/after
- Word clusters
- Gazetteers

According to the New York Times...

Apple released a new version...

Leicestershire

Boston
Nonlocal Features

The news agency Tanjug reported on the outcome of the meeting.

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

- Various ways to capture this information — we’ll talk about this in a few lectures

Finkel and Manning (2008), Ratinov and Roth (2009)
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
- Viterbi looks like looping over all spans that could lead to a given point
- Pros: features can look at whole span at once
- Cons: there’s an extra factor of \( n \) during inference

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
  - Partial credit? Typically no but more complex metrics exist

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

B-PER  I-PER  O  O  O  B-LOC  O  O  O  B-ORG  O  O

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

- More correct: ROC curve
- Measure the area under the curve as a way of evaluating the system holistically
### 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>Model</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collobert et al. (2011)*</td>
<td>89.59</td>
</tr>
<tr>
<td>Lin and Wu (2009)</td>
<td>83.78</td>
</tr>
<tr>
<td>Lin and Wu (2009)*</td>
<td>90.90</td>
</tr>
<tr>
<td>Huang et al. (2015)*</td>
<td>90.10</td>
</tr>
<tr>
<td>Passos et al. (2014)</td>
<td>90.05</td>
</tr>
<tr>
<td>Passos et al. (2014)*</td>
<td>90.90</td>
</tr>
<tr>
<td>Luo et al. (2015)* + gaz</td>
<td>89.9</td>
</tr>
<tr>
<td>Luo et al. (2015)* + gaz + linking</td>
<td>91.2</td>
</tr>
<tr>
<td>Chiu and Nichols (2015)</td>
<td>90.69</td>
</tr>
<tr>
<td>Chiu and Nichols (2015)*</td>
<td>90.77</td>
</tr>
<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>
</tbody>
</table>

Ratinov and Roth (2009)  
Lample et al. (2016)
Structured SVM

- **CRF:** \( \log P(y|x) \propto \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i) \)

- We can formulate an SVM using the same features

\[
 w^\top f(x, y) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i) 
\]

Minimize \( \lambda \|w\|^2_2 + \sum_{j=1}^{m} \xi_j \)

s.t. \( \forall j \quad \xi_j \geq 0 \)

\( \forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y_j^*) \geq w^\top f(x_j, y) + \ell(y, y_j^*) - \xi_j \)
Structured SVM

\[ w^\top f(x, y) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i) \]

Minimize \( \lambda \|w\|^2_2 + \sum_{j=1}^{m} \xi_j \)

s.t. \( \forall j \quad \xi_j \geq 0 \)

\( \forall j \forall y \in \mathcal{Y} \quad w^\top f(x_j, y^*_j) \geq w^\top f(x_j, y) + \ell(y, y^*_j) - \xi_j \)

- Exponentially large state space! Use Viterbi for loss-augmented decode
- Same as normal Viterbi but boost wrong labels’ scores by 1 (if using Hamming loss)
- Only need Viterbi, not forward-backward...hmm...
Viterbi Time Complexity

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
Viterbi Time Complexity

- 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 state

- $O(nks)$ time complexity with beam size of $k$
How good is beam search?

- Big enough beam size: always exact! Usually works well even with smaller beams
- What’s the case when $k=1$?
- How about when there’s no transition model?
  - Depends on the strength of nonlocal interactions — we’ll come back to this later!
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. The weight vector needs to be stored explicitly, but all features and gradients are typically faster to handle sparsely

- Think about your data structures: if things are too slow
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 bug)
    - Are expectations being computed correctly? Do probabilities normalize / expectations look reasonable?
  - **Learning**: 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
  - **Model**: if dev set performance is bad: work on features more!
Next Time

- Unsupervised sequence modeling
- Writing tips as you prepare your report