CS388: Natural Language Processing
Lecture 5: Sequence Models II

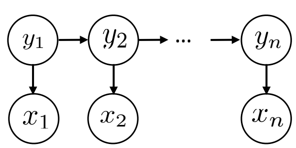
Greg Durrett

Administrivia

- Mini 1 graded by next lecture
- Project 1 is out, sample writeups on website

Recall: HMMs

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$



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

- Training: maximum likelihood estimation (with smoothing)
- Inference problem: $\mathrm{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Viterbi: $\mathrm{score}_i(s) = \max_{y_{i-1}} P(s|y_{i-1}) P(x_i|s) \mathrm{score}_{i-1}(y_{i-1})$

This Lecture

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

Named Entity Recognition

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 .

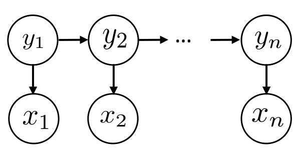
PERSON LOC ORG

- 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)

CRFs

Conditional Random Fields

- HMMs are expressible as Bayes nets (factor graphs)



- This reflects the following decomposition:

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

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

Conditional Random Fields

- HMMs: $P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2)\ldots$
- CRFs: discriminative models with the following globally-normalized form:

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

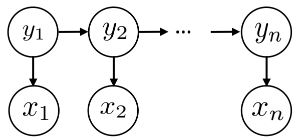
normalizer

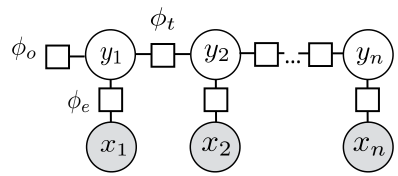
any real-valued scoring function of its arguments

- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> 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: $P(\mathbf{y}, \mathbf{x}) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2)\ldots$



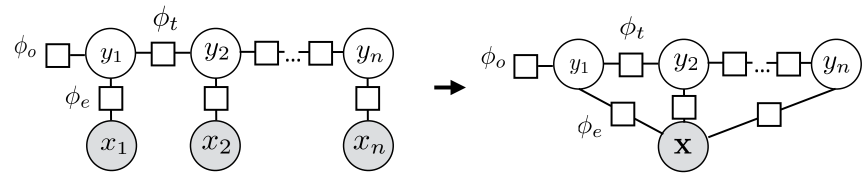


- CRFs:

$$P(\mathbf{y}|\mathbf{x}) \propto \prod_k \exp(\phi_k(\mathbf{x}, \mathbf{y}))$$

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

Sequential CRFs



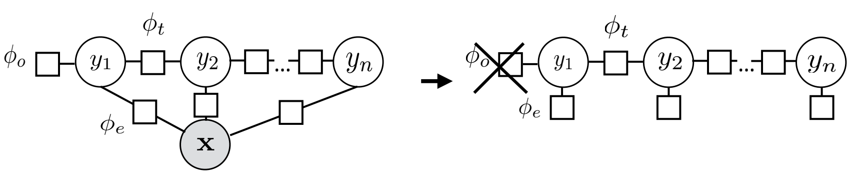
$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\phi_o(y_1)) \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \cancel{\exp(\phi_e(x_i, y_i))}$$

$$\prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$$

token index — lets us look at current word

- 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

Sequential CRFs



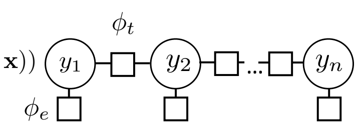
- Notation: omit **x** from the factor graph entirely (implicit)
- Don't include initial distribution, can bake into other factors

Sequential CRFs:

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

Feature Functions

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



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

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

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

- Looks like our single weight vector multiclass logistic regression model

Basic Features for NER

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

O B-LOC O

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

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

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

Features for NER

$\phi_e(y_i, i, \mathbf{x})$

LOC
Leicestershire is a nice place to visit…

PER
Leonardo DiCaprio won an award…

LOC
I took a vacation to Boston

ORG
Apple released a new version…

LOC PER
Texas governor Greg Abbott said

ORG
According to the New York Times…

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

Leicestershire

Boston

Apple released a new version…

According to the New York Times…

CRFs Outline

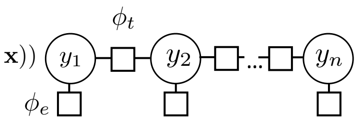
- Model: $P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, \mathbf{x}))$

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

- Inference
- Learning

Computing (arg)maxes

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



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

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

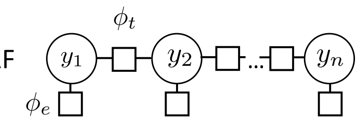
$$= \max_{y_2,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)}e^{\phi_e(y_n,n,\mathbf{x})}\ldots e^{\phi_e(y_2,2,\mathbf{x})}\boxed{\max_{y_1}}e^{\phi_t(y_1,y_2)}\underbrace{e^{\phi_e(y_1,1,\mathbf{x})}}$$

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

- $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the Ps now, same dynamic program

Inference in General CRFs

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

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

- Inference: argmax P(**y**|**x**) from Viterbi
- Learning

Training CRFs

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

- Logistic regression: $P(y|x) \propto \exp w^\top f(x, y)$
- Maximize $\mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \log P(\mathbf{y}^*|\mathbf{x})$
- Gradient is completely analogous to logistic regression:

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

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

intractable!

Training CRFs

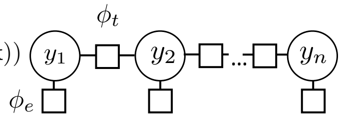
$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*,\mathbf{x}) = \sum_{i=2}^{n} f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} f_e(y_i^*, i, \mathbf{x})$$
$$-\mathbb{E}_{\mathbf{y}}\left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right]$$

- Let's focus on emission feature expectation

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

Computing Marginals

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



- 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(**x**) for HMMs
- For both HMMs and CRFs:

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

Z for CRFs, P(**x**) for HMMs

Posteriors vs. Probabilities

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

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

	$P(x_i\|y_i), P(y_i\|y_{i-1})$	$P(y_i\|\mathbf{x}), P(y_{i-1}, y_i\|\mathbf{x})$
HMM	Model parameter (usually multinomial distribution)	Inferred quantity from forward-backward
CRF	Undefined (model is by definition conditioned on **x**)	Inferred quantity from forward-backward

Training CRFs

- For emission features:

$$\frac{\partial}{\partial w}\mathcal{L}(\mathbf{y}^*,\mathbf{x}) = \sum_{i=1}^{n} f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^{n}\sum_{s} P(y_i = s|\mathbf{x}) f_e(s, i, \mathbf{x})$$

gold features — expected features under model

- 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

CRFs Outline

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

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

- Inference: argmax P(**y**|**x**) from Viterbi
- Learning: run forward-backward to compute posterior probabilities; then

$$\frac{\partial}{\partial w} \mathcal{L}(\mathbf{y}^*, \mathbf{x}) = \sum_{i=1}^{n} f_e(y_i^*, i, \mathbf{x}) - \sum_{i=1}^{n} \sum_{s} P(y_i = s|\mathbf{x}) f_e(s, i, \mathbf{x})$$

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 $\sum_s \mathrm{forward}_i(s)\mathrm{backward}_i(s)$ 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

NER

NER

- 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 (/ˈtʌnjʊɡ/) (Serbian Cyrillic: Тањуг) is a Serbian state news agency based in Belgrade.[2]

Nonlocal Features

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

ORG?
PER?

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 .

PER O LOC O ORG O

- 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

Sarawagi and Cohen (2004)

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 .

PERSON LOC ORG

- 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?

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

Ratinov and Roth (2009)

Lample et al. (2016)

LSTM-CRF (no char)	90.20
LSTM-CRF	**90.94**
S-LSTM (no char)	87.96
S-LSTM	90.33

BiLSTM-CRF + ELMo
Peters et al. (2018) **92.2**

Beam Search

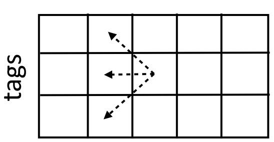
Viterbi Time Complexity

VBD		VB			
VBN VBZ		VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN
Fed	raises	interest	rates	0.5	percent

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

sentence

tags



- $O(ns^2)$ — s is ~40 for POS, n is ~20

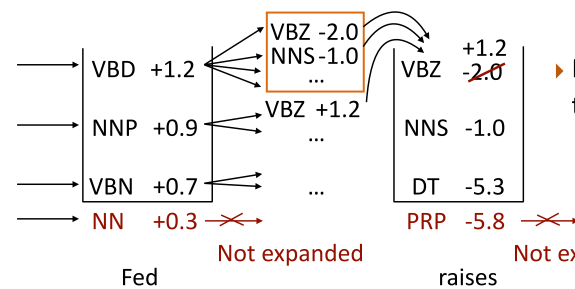
Viterbi Time Complexity

VBD VB
VBN VBZ VBP VBZ
NNP NNS NN NNS CD NN
Fed raises interest rates 0.5 percent

- 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

Beam Search

- Maintain a beam of *k* plausible states at the current timestep
- Expand all states, only keep *k* top hypotheses at new timestep



- Maintain priority queue to efficiently add things
- Beam size of k, time complexity O(nks log(ks))

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