Sequence Models II

Wei Xu

(many slides from Greg Durrett)

This Lecture

- Named entity recognition (NER)
- CRFs: model (+features for NER), inference, learning
- Reading: Eisenstein Chapter 7 & 8.3

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)

CRFs

Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

John Lafferty[†*] LAFFERTY@CS.CMU.EDU
Andrew McCallum[*†] MCCALLUM@WHIZBANG.COM
Fernando Pereira[*‡] FPEREIRA@WHIZBANG.COM

[*]WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

Abstract

We present *conditional random fields*, a framework for building probabilistic models to segment and label sequence data. Conditional random fields offer several advantages over hidden Markov models and stochastic grammars for such tasks, including the ability to relax strong independence assumptions made in those models. Conditional random fields also avoid a fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graphical models, which can be biased towards states

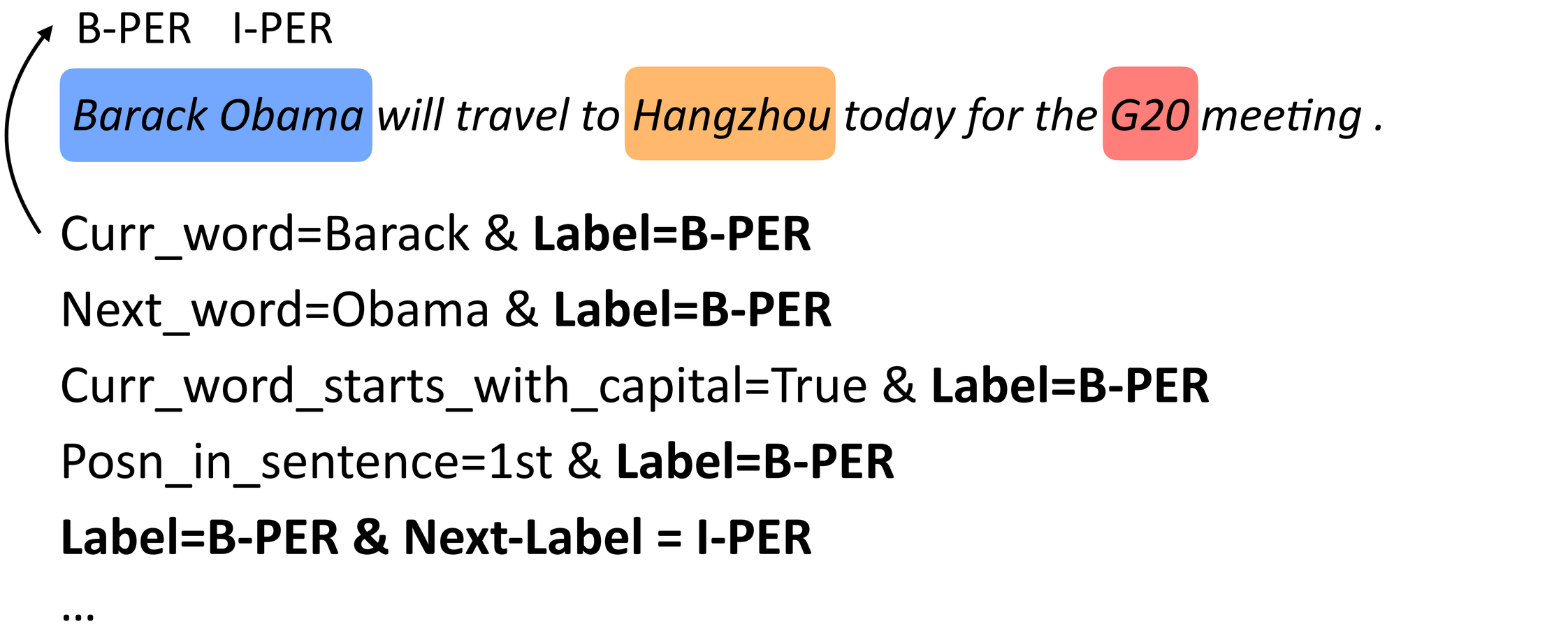
mize the joint likelihood of training examples. To define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

This difficulty is one of the main motivations for looking at conditional models as an alternative. A conditional model specifies the probabilities of possible label sequences given an observation sequence. Therefore, it does not expend

(ICML 2001)

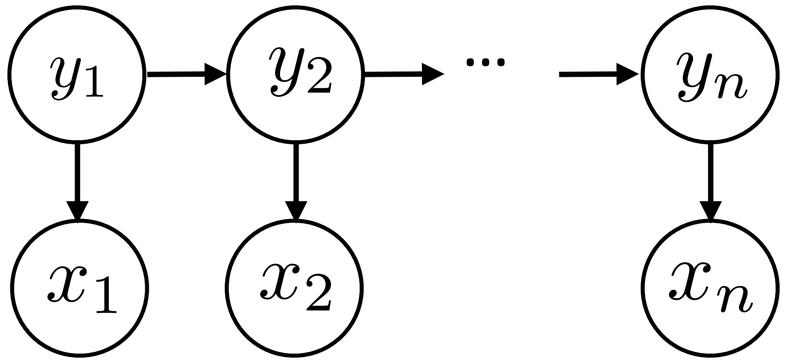
Where we're going

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



HMMs, Formally

- HMMs are expressible as Bayes nets (factor graphs)



- This reflects the following decomposition:

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

- 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

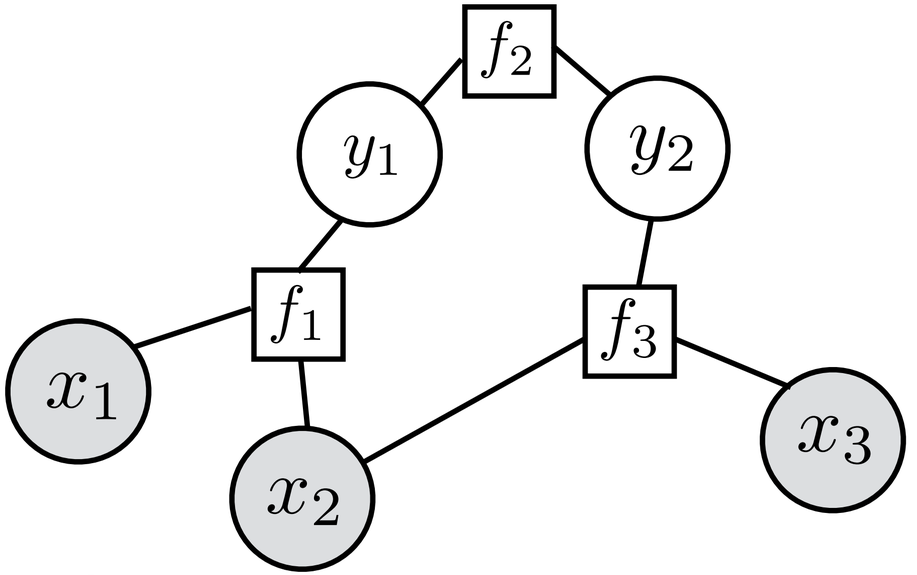
- Special case: linear feature-based potentials $\phi_k(\mathbf{x}, \mathbf{y}) = w^\top f_k(\mathbf{x}, \mathbf{y})$

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

- Looks like our single weight vector multiclass logistic regression model

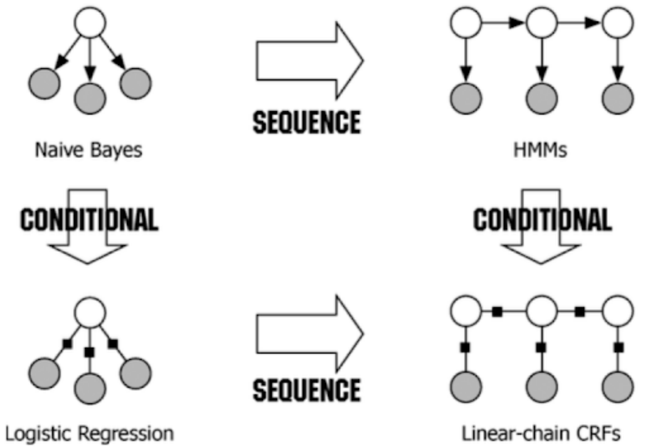
HMMs vs. CRFs

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



- Conditional model: x's are observed

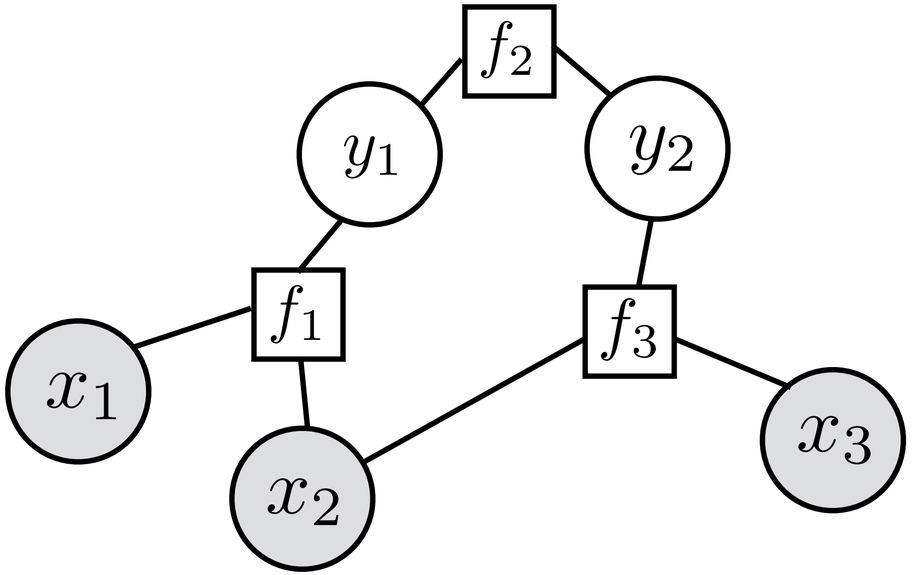
- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> generative vs. discriminative
  (locally normalized discriminative models do exist (MEMMs))



Sutton&McCallum: https://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf

HMMs vs. CRFs

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



- Conditional model: x's are observed
- Naive Bayes : logistic regression :: HMMs : CRFs
  local vs. global normalization <-> generative vs. discriminative
  (locally normalized discriminative models do exist (MEMMs))
- HMMs: in the standard setup, emissions consider one word at a time
- CRFs: features over many words simultaneously, non-independent features (e.g., suffixes and prefixes), doesn't have to be a generative model

Problem with CRFs

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

- Normalizing constant

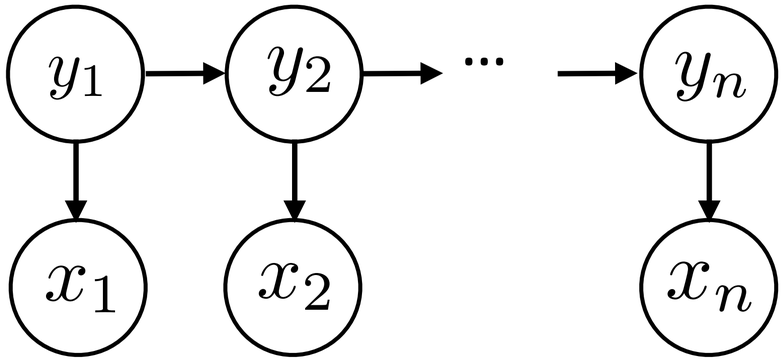
$$Z = \sum_{\mathbf{y}'} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y}')\right)$$

- Inference: $\mathbf{y}_{\text{best}} = \text{argmax}_{\mathbf{y}'} \exp\left(\sum_{k=1}^{n} w^\top f_k(\mathbf{x}, \mathbf{y}')\right)$

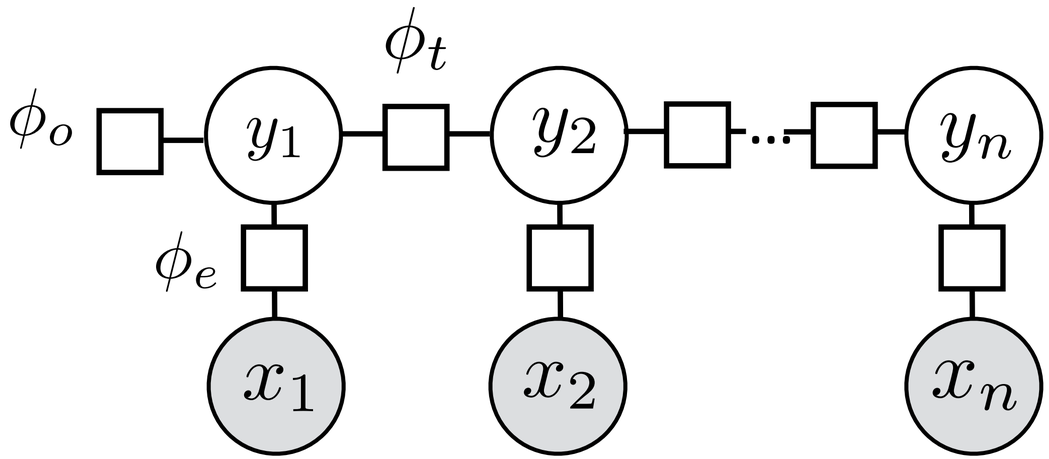
- If y consists of 5 variables with 30 values each, how expensive are these?
- Need to constrain the form of our CRFs to make it tractable

Sequential CRFs

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



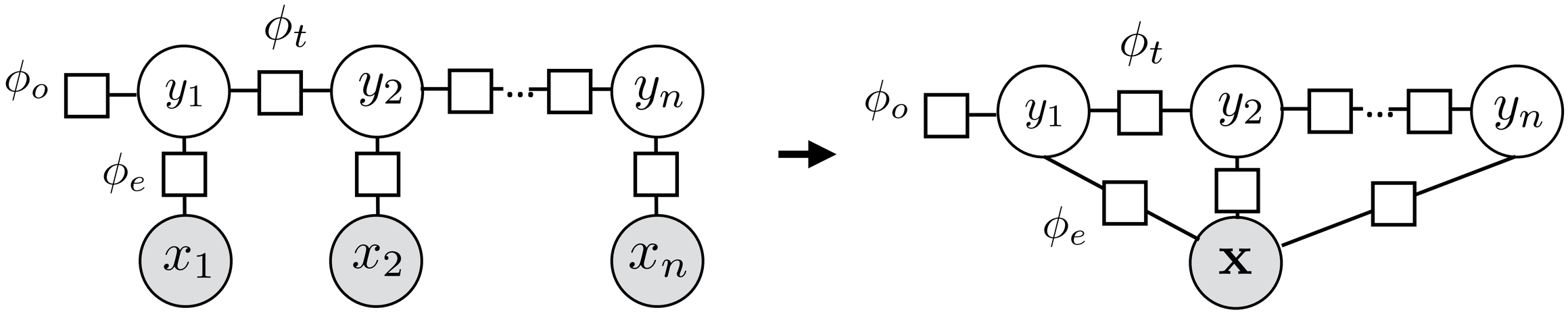
- CRFs:



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

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

Sequential CRFs



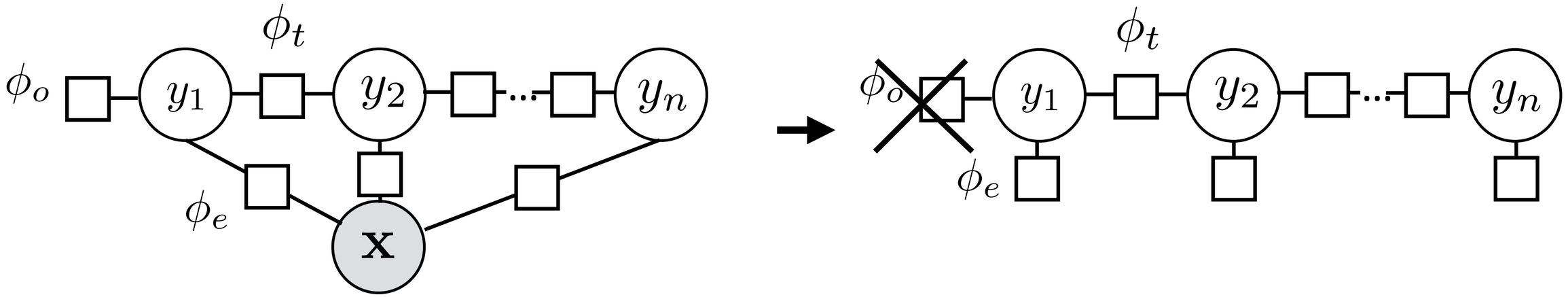
$$P(\mathbf{y}|\mathbf{x}) \propto \exp(\phi_o(y_1)) \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(x_i, y_i))$$

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

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

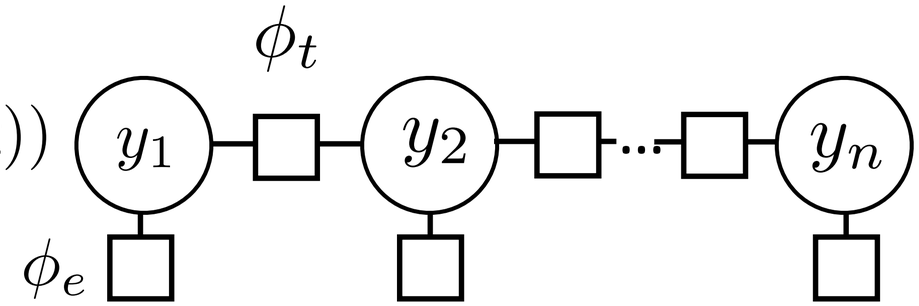
Sequential CRFs (aka linear-chain CRFs):

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

Features for NER

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

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

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

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

I[B-LOC & Prev word = *to*]

Emission 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
  - POS Tags before/after (if we run a POS tagger first)
- Word clusters
- Gazetteers

Leicestershire

Boston

Apple released a new version…

According to the New York Times…

Inference and Learning in CRFs

Linear-chain CRFs Outline

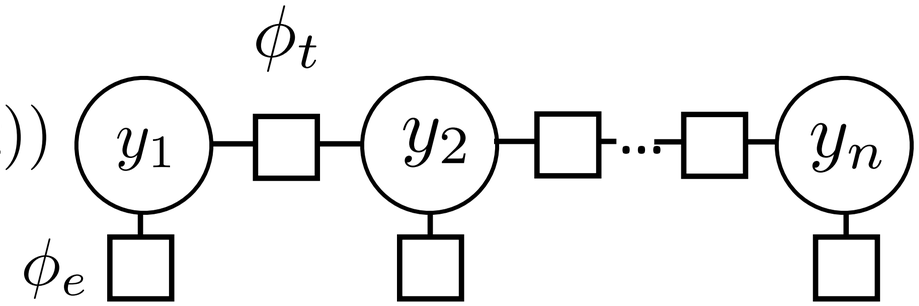
- Model: $$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$

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

- 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

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

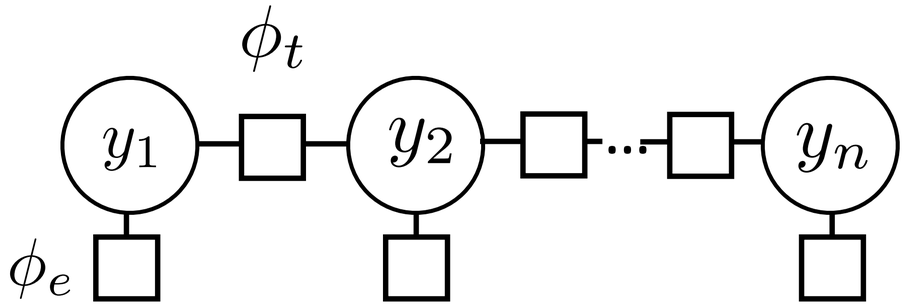
$$= \max_{y_2,\ldots,y_n} e^{\phi_t(y_{n-1},y_n)} e^{\phi_e(y_n,n,\mathbf{x})} \ldots e^{\phi_e(y_2,2,\mathbf{x})} \boxed{\max_{y_1} e^{\phi_t(y_1,y_2)} e^{\phi_e(y_1,1,\mathbf{x})}}$$

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

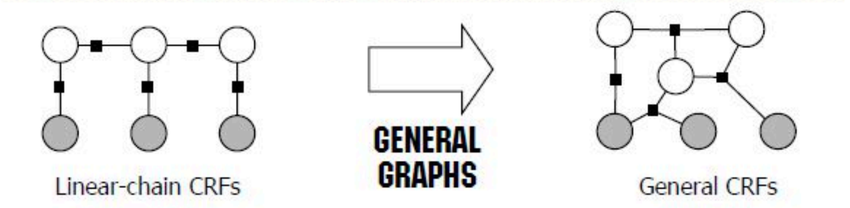
- $\exp(\phi_t(y_{i-1}, y_i))$ and $\exp(\phi_e(y_i, i, \mathbf{x}))$ play the role of the Ps now, same dynamic program

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:

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

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

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

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

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

Forward-Backward Algorithm

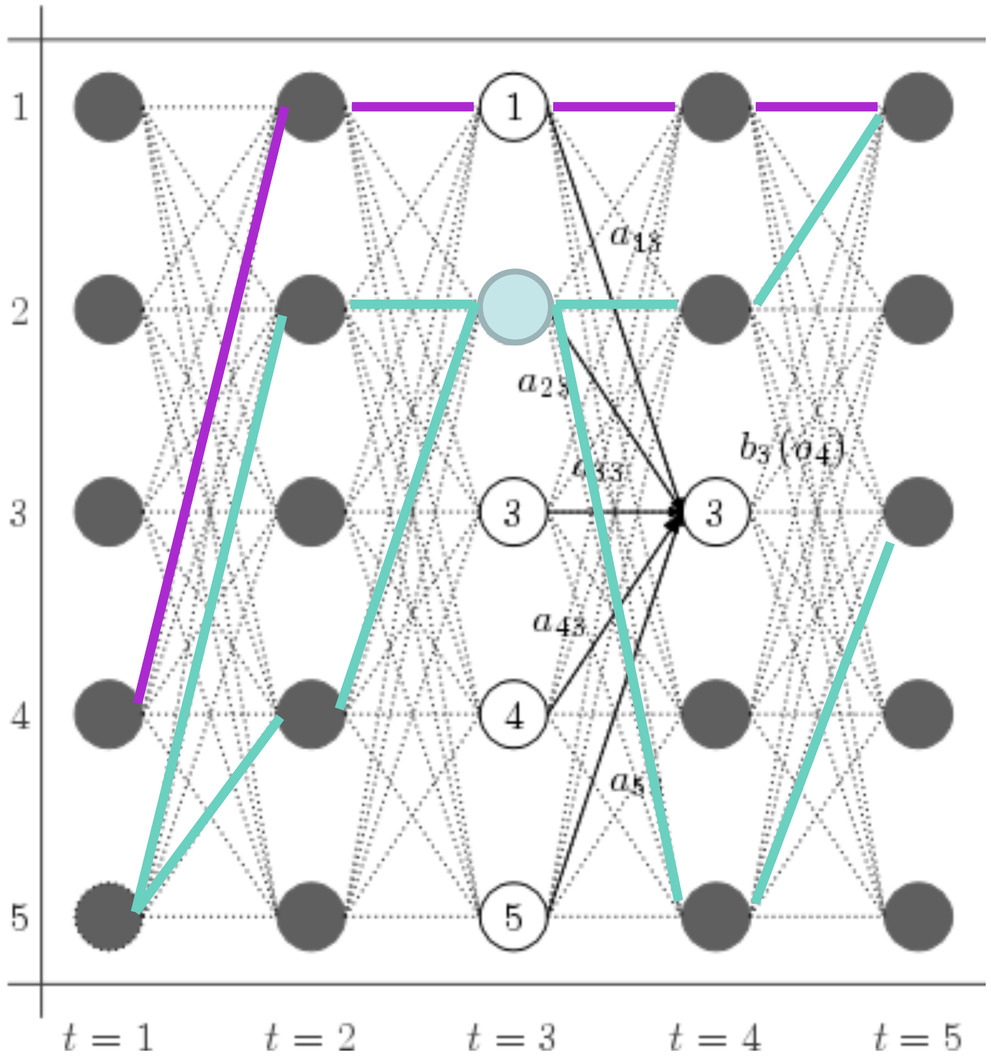
- How do we compute these marginals $P(y_i = s|\mathbf{x})$?

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

- What did Viterbi compute? $P(\mathbf{y}_{\max}|\mathbf{x}) = \max_{y_1,\ldots,y_n} P(\mathbf{y}|\mathbf{x})$

- Can compute marginals with dynamic programming as well using the forward-backward algorithm

Forward-Backward Algorithm

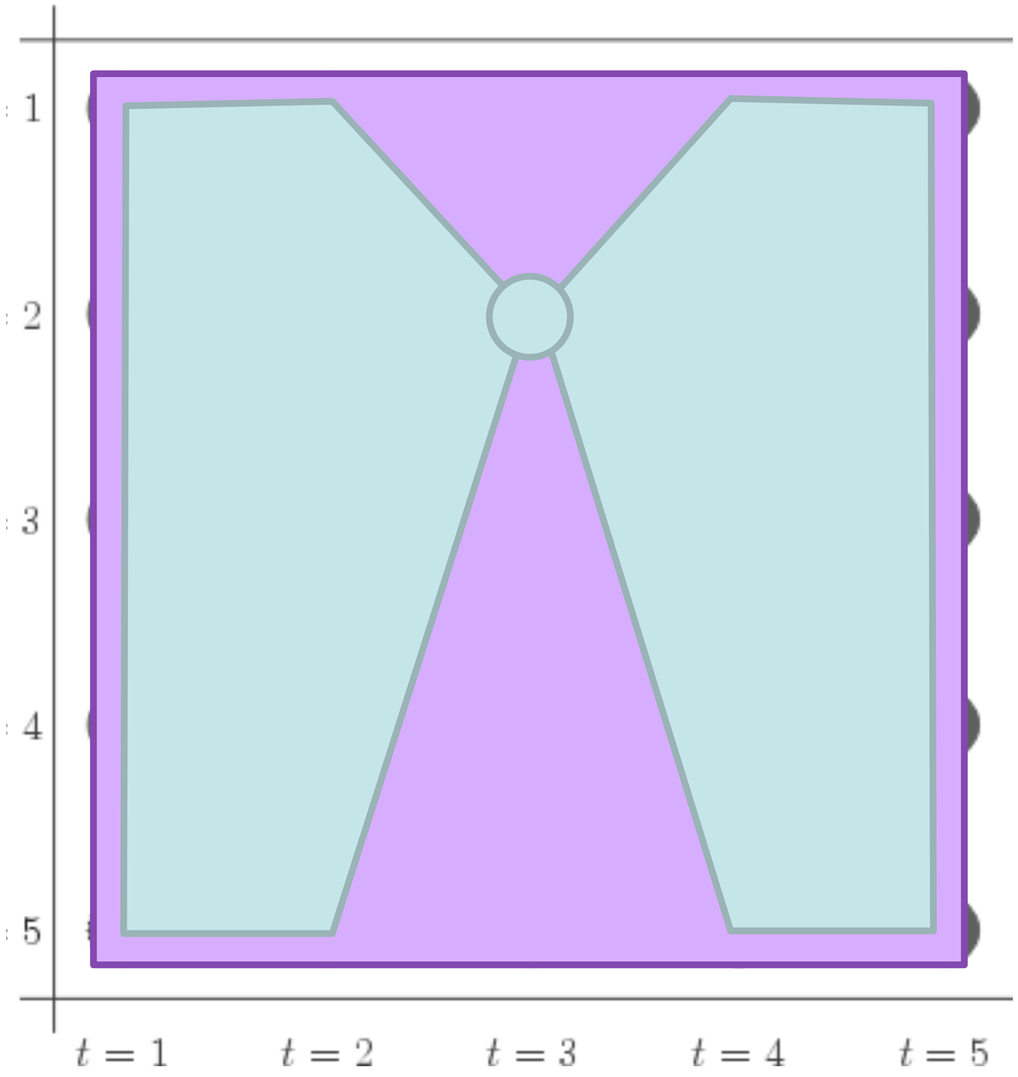


$$P(y_3 = 2|\mathbf{x}) =$$

$$\frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

slide credit: Dan Klein

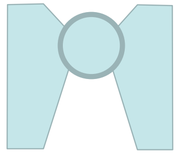
Forward-Backward Algorithm

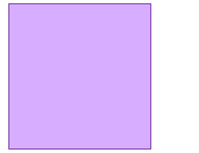


$$P(y_3 = 2|\mathbf{x}) =$$

$$\frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}}$$

$$= \frac{\quad}{\quad}$$

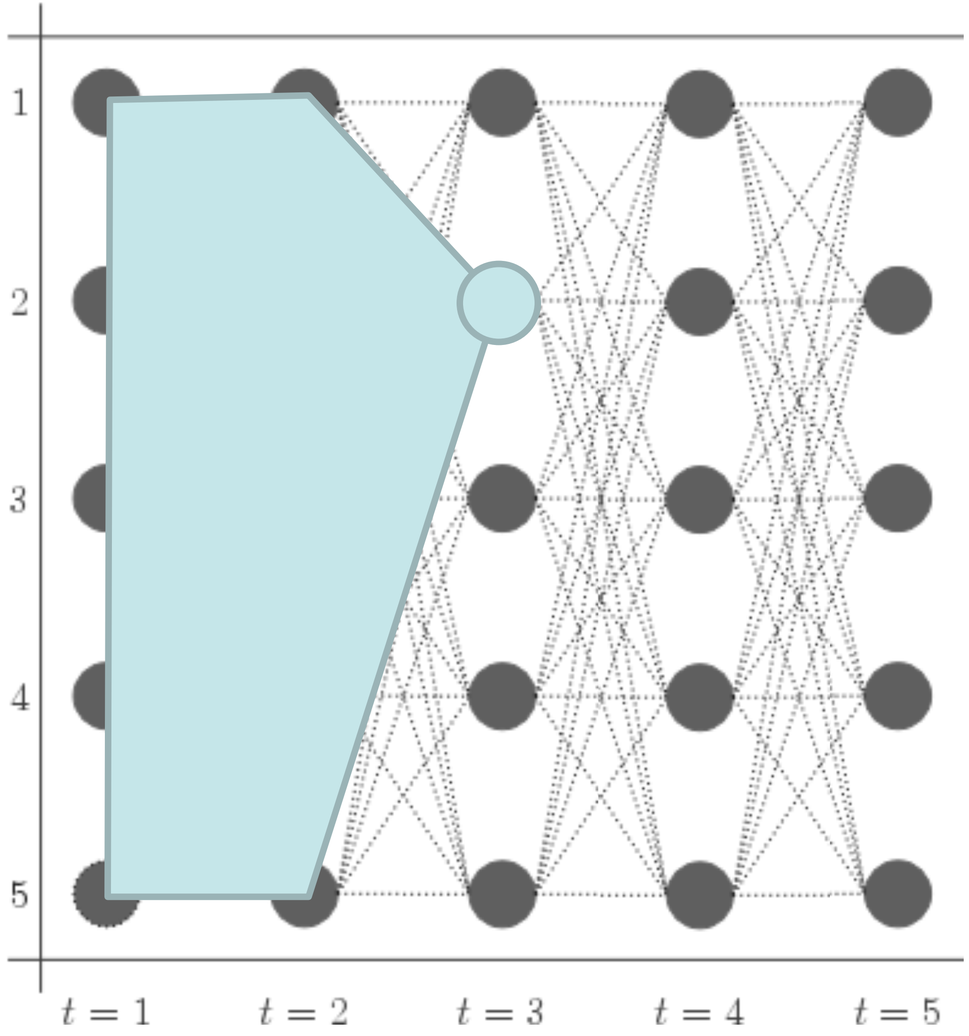




- Easiest and most flexible to do one pass to compute  and one to compute 

slide credit: Dan Klein

Forward-Backward Algorithm



- Initial:

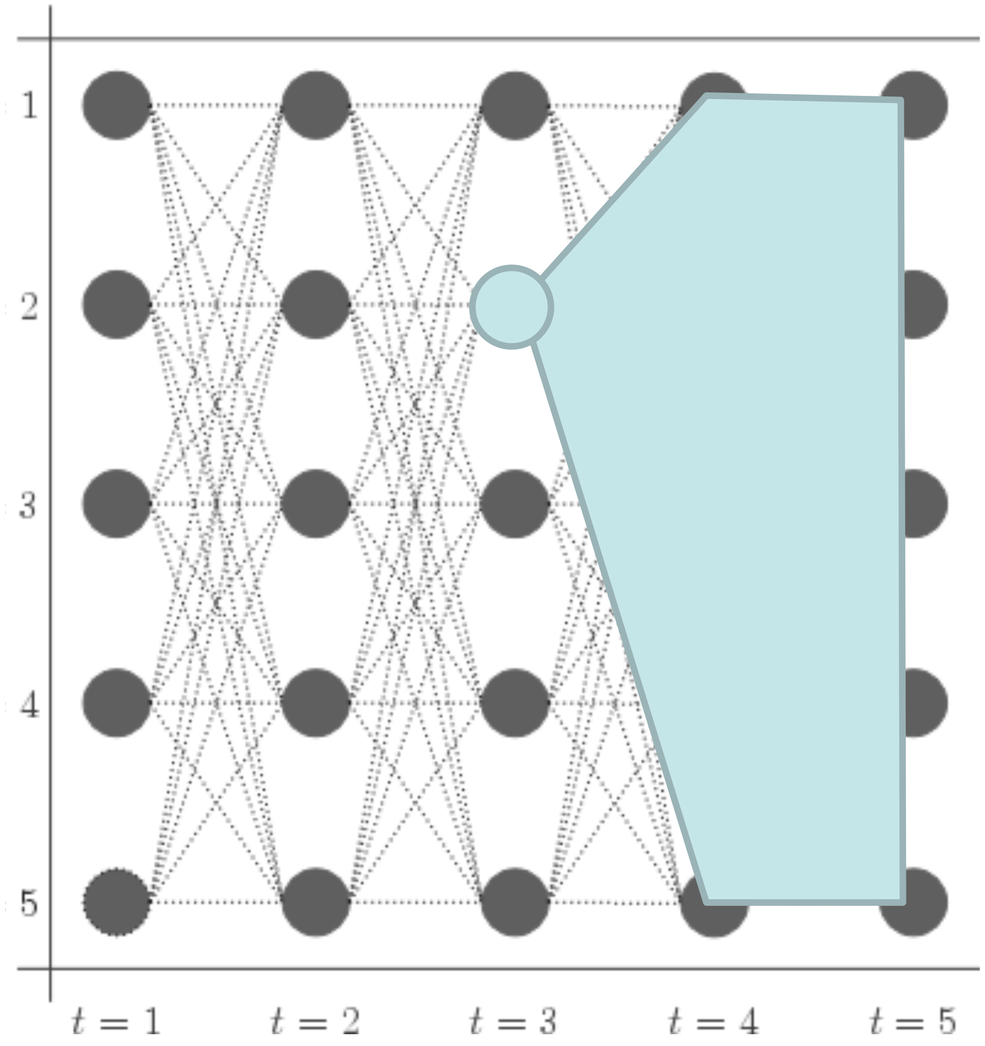
$$\alpha_1(s) = \exp(\phi_e(s, 1, \mathbf{x}))$$

- Recurrence:

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \exp(\phi_e(s_t, t, \mathbf{x})) \exp(\phi_t(s_{t-1}, s_t))$$

- Same as Viterbi but summing instead of maxing!
- These quantities get very small! Store everything as log probabilities

Forward-Backward Algorithm



- Initial:

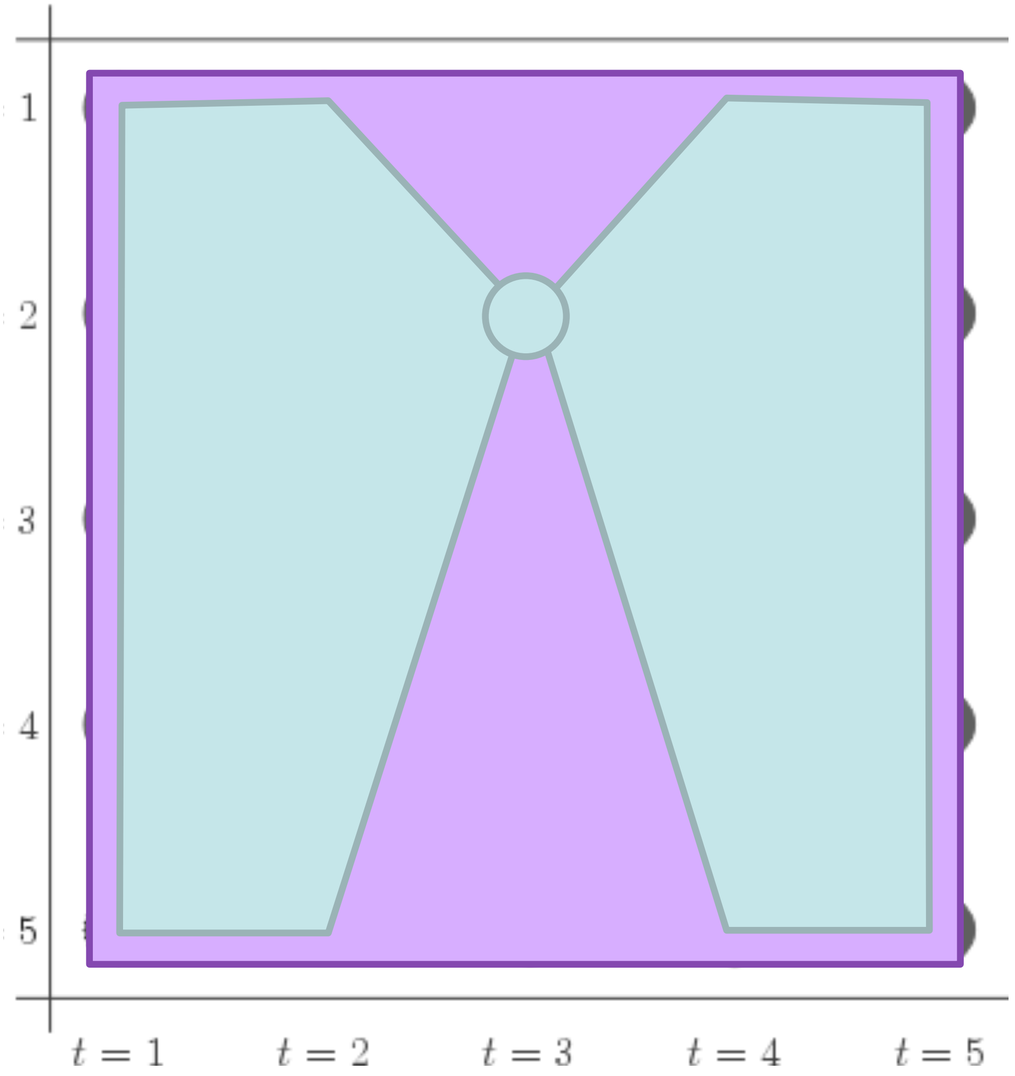
$$\beta_n(s) = 1$$

- Recurrence:

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1}, t+1, \mathbf{x})) \exp(\phi_t(s_t, s_{t+1}))$$

- Big differences: count emission for the *next* timestep (not current one)

Forward-Backward Algorithm



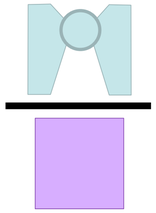
$$\alpha_1(s) = \exp(\phi_e(s, 1, \mathbf{x}))$$

$$\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \exp(\phi_e(s_t, t, \mathbf{x})) \exp(\phi_t(s_{t-1}, s_t))$$

$$\beta_n(s) = 1$$

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \exp(\phi_e(s_{t+1}, t+1, \mathbf{x})) \exp(\phi_t(s_t, s_{t+1}))$$

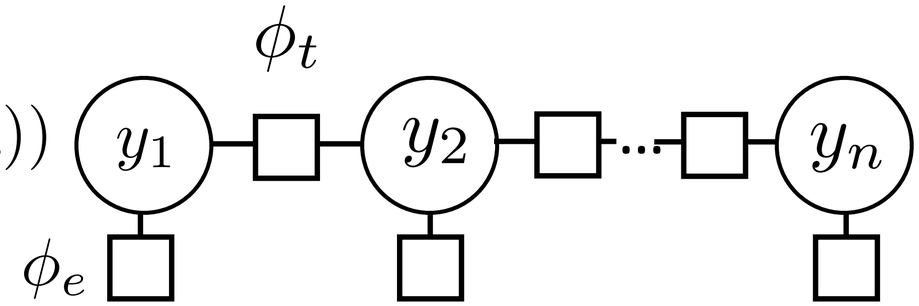
$$P(s_3 = 2|\mathbf{x}) = \frac{\alpha_3(2)\beta_3(2)}{\sum_i \alpha_3(i)\beta_3(i)} =$$



- What is the denominator here? Z

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

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 learned transition features (e.g., use heuristic weights, or just enforce constraints like B-PER -> I-ORG is illegal)

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

        - apply the gradient update

Implementation Tips for CRFs

- Caching is your friend! Cache feature vectors especially, and reduce redundant computation
- 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 $\sum_s \text{forward}_i(s)\text{backward}_i(s)$ 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 (of 1 or 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

Application in NER

Harmonic Mean (Math Review)

- Arithmetic Mean = (P + F) / 2
- Geometric Mean = $\sqrt{P \times F}$
- Harmonic Mean = 2 x P x F / (P + F)

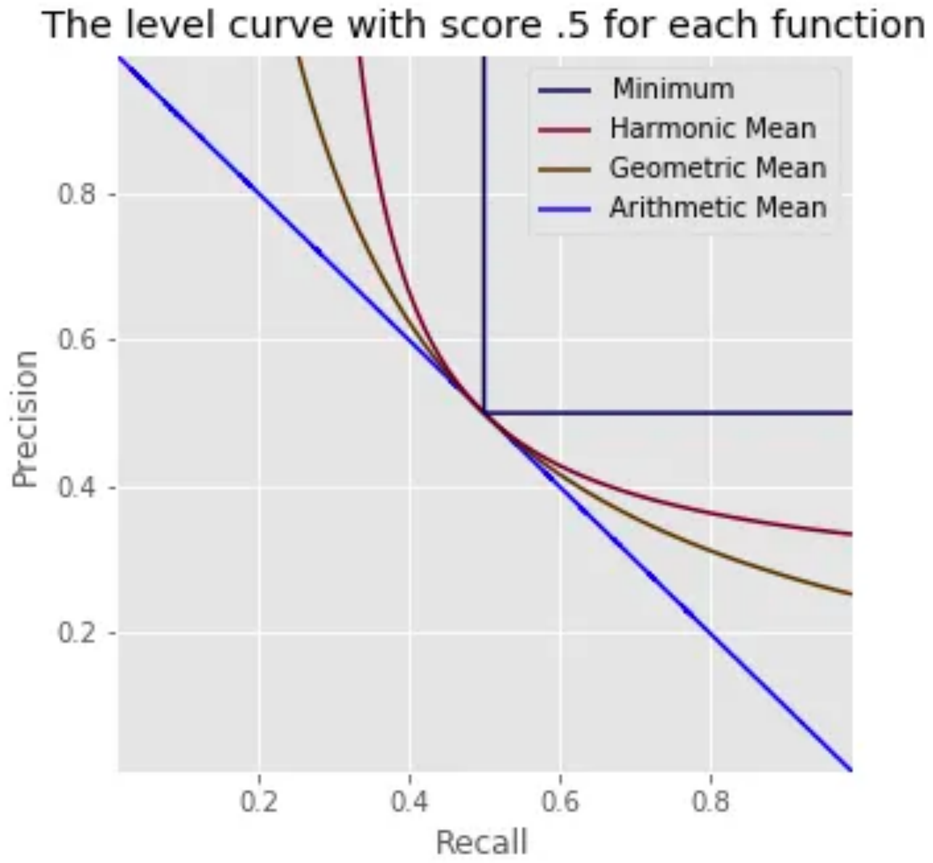
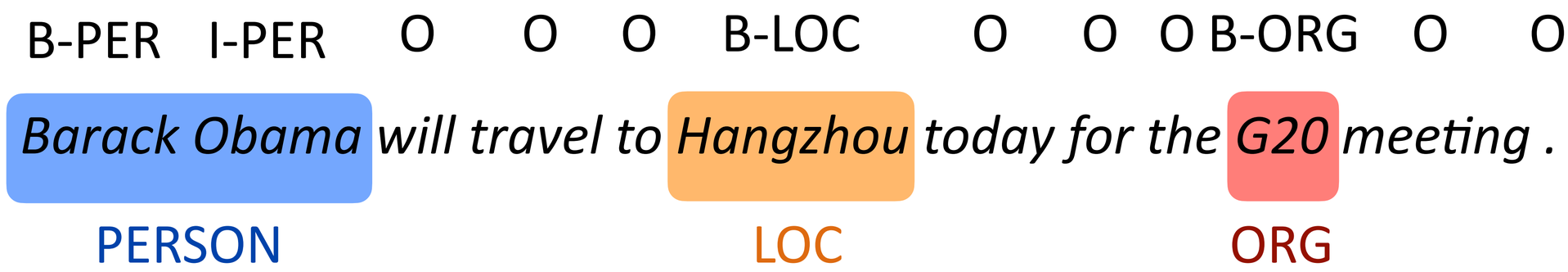


Image credit: Greg Gandenberger

Evaluating NER



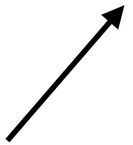
- Prediction of all Os still gets 66% accuracy on this example!
- What we really want to know: how many named entity *chunk* predictions did we get right?
  - Precision: of the ones we predicted, how many are right?
  - Recall: of the gold named entities, how many did we find?
  - F-measure: harmonic mean of these two

http://www.davidsbatista.net/blog/2018/05/09/Named_Entity_Evaluation/

NER Results

- CRF with lexical features can get around 85 F1 on CoNLL 2003:
  4 classes (PER, ORG, LOC, MISC) on newswire data
- What else do we need to capture?
- World knowledge:

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



Tanjug

From Wikipedia, the free encyclopedia

Tanjug (/ˈtʌnjʊg/) (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?

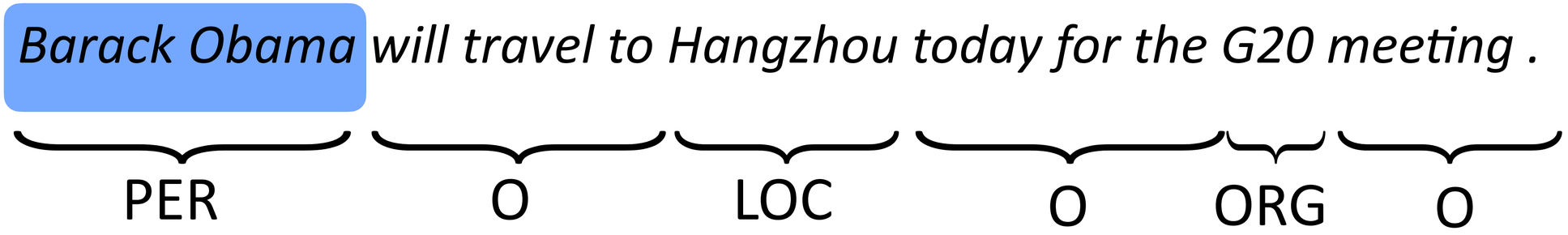
PER?

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 CRF Models



- Chunk-level (*n*-gram) 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)

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)

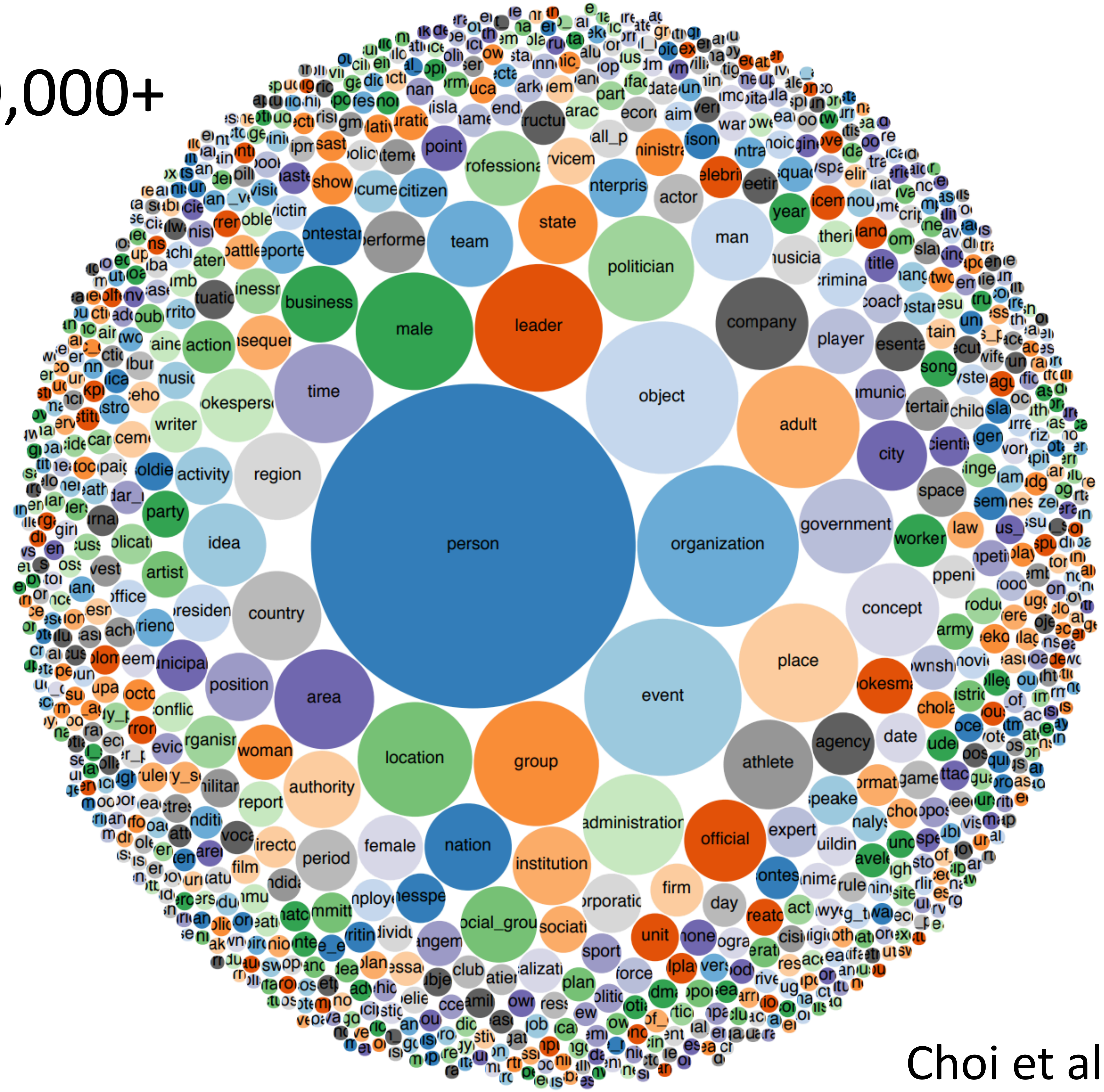
92.2

Fine-tuning approach	Dev F1	Test F1
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4

Devlin et al. (2019)

Ultra-fine Entity Typing

- Scaling up to larger typesets 10,000+
- Person/official/president …



Choi et al. (2018)

Next Up

- More sequential models
  - Recurrent Neural Network
  - Neural CRF model