CS395T: Structured Models for NLP
Lecture 4: Sequence Models I

Greg Durrett

Parts of this lecture adapted from Dan Klein, UC Berkeley and Vivek Srikumar, University of Utah

Administrivia

- Project 1 out today!
  - Viterbi algorithm, CRF NER system, extension
  - Extension should be substantial: don't just try one additional feature (try several features, do some error analysis, write some motivation)
  - This class will cover what you need to get started on it, the next 1-2 classes will cover everything you need to complete it
- Greg's Office Hours tomorrow: 9am — 11am (one-time change)

Recall: Multiclass Classification

- Logistic regression: $P(y|x) = \frac{\exp\left(w^\top f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w^\top f(x,y')\right)}$

  Gradient (unregularized): $\frac{\partial}{\partial w_i}\mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \mathbb{E}_y[f_i(x_j, y)]$

- SVM: defined by quadratic program (minimization, so gradients are flipped)

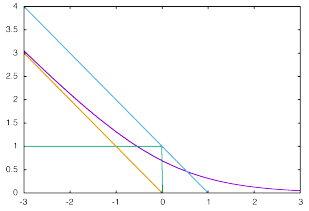
  Loss-augmented decode

  $$\xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*)$$

  Subgradient (unregularized) on *j*th example $= f_i(x_j, y_{\max}) - f_i(x_j, y_j^*)$

Structured Prediction

- Four elements of a structured machine learning method:
  - Model: probabilistic, max-margin, deep neural network
  - Objective



  - Inference: just maxes and simple expectations so far, but will get harder
  - Training: gradient descent

Optimization

- Stochastic gradient *ascent*
  - Very simple to code up
  - "First-order" technique: only relies on having gradient
  - Difficult to tune step size

$$w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w}\mathcal{L}$$

- Newton's method
  - Second-order technique
  - Optimizes quadratic instantly

$$w \leftarrow w + \left(\frac{\partial^2}{\partial w^2}\mathcal{L}\right)^{-1} g$$

Inverse Hessian: *n* x *n* mat, expensive!

- Quasi-Newton methods: L-BFGS, etc.
  - Approximate inverse Hessian with gradients over time

AdaGrad

- Optimized for problems with sparse features
- Sparse features are often heterogeneous: some fire on every example, some fire on one example in the corpus (but are still valuable!)

$$w_i \leftarrow w_i + \alpha \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}$$

per-parameter learning rate based on sum of previous gradients

- Avoids common features getting large values compared to rare features
- Usually works out-of-the-box with little tuning
- Other techniques for optimizing deep models — more later!

Duchi et al. (2011)

Implementation Details

- SGD/AdaGrad have a batch size parameter
  - Large batches (>50 examples): can parallelize within batch
  - …but bigger batches often means more epochs required because you make fewer parameter updates
- Shuffling: online methods are sensitive to dataset order
  - Fixed shuffle: breaks correlations between neighboring sentences
  - Per-epoch shuffle: lower final model variance
- Regularization: makes SGD slower to implement with sparse features
  - Either don't regularize (might work better than you think!), or do it lazily (see adagrad_trainer.py in Project 1)

This Lecture

- Sequence modeling
- HMMs for POS tagging
- HMM parameter estimation
- Viterbi algorithm

Linguistic Structures

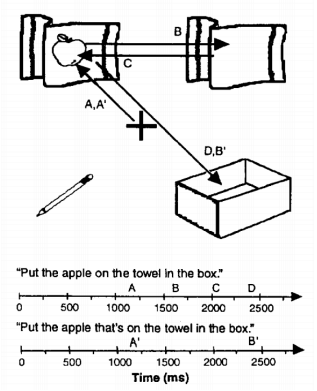
- Language is tree-structured

- Understanding syntax fundamentally requires trees — the sentences have the same shallow analysis

PRP	VBZ	DT	NN	IN	NNS	
I	ate	the	spaghetti	with	chopsticks	
PRP	VBZ	DT	NN	IN	NNS	
I	ate	the	spaghetti	with	meatballs	

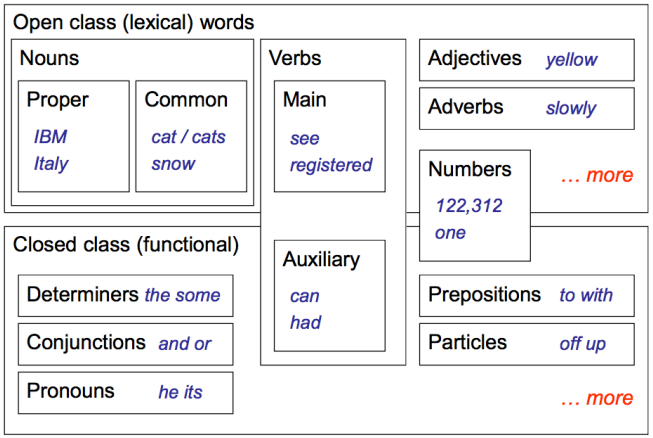
Linguistic Structures

- Language is sequentially structured: interpreted in an online way



Tanenhaus et al. (1995)

POS Tagging

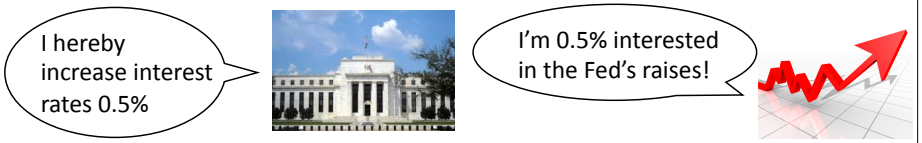


Slide credit: Dan Klein

POS Tagging

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

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



- Other paths are also plausible but even more semantically weird…
- What governs the correct choice? Word + context
  - Word identity: most words have <=2 tags, many have one (*percent*, *the*)
  - Context: nouns start sentences, nouns follow verbs, etc.

What is this good for?

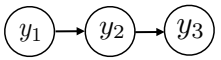
- Text-to-speech: record, lead
- Preprocessing step for syntactic parsers
- Domain-independent disambiguation for other tasks
- (Very) shallow information extraction

Sequence Models

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- POS tagging: ***x*** is a sequence of words, ***y*** is a sequence of tags (most of the time…)
- Today: generative models P(***x, y***); discriminative models next time

Hidden Markov Models

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- Model the sequence of *y* as a Markov process (dynamics model)
- Markov property: future is conditionally independent of the past given the present

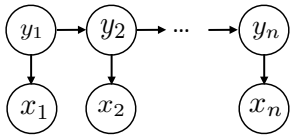


$$P(y_3|y_1, y_2) = P(y_3|y_2)$$

- Lots of mathematical theory about how Markov chains behave
- If *y* are tags, this roughly corresponds to assuming that the next tag only depends on the current tag, not anything before

Hidden Markov Models

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



$$P(\mathbf{y}, \mathbf{x}) = \underbrace{P(y_1)}_{\text{Initial distribution}} \underbrace{\prod_{i=2}^{n} P(y_i|y_{i-1})}_{\text{Transition probabilities}} \underbrace{\prod_{i=1}^{n} P(x_i|y_i)}_{\text{Emission probabilities}}$$

- Observation (*x*) depends only on current state (*y*)
- Multinomials: tag x tag transitions, tag x word emissions
- P(*x*|*y*) is a distribution over all words in the vocabulary — not a distribution over features

Transitions in POS Tagging

- Dynamics model $P(y_1)\prod_{i=2}^{n} P(y_i|y_{i-1})$

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

- $P(y_1 = \text{NNP})$ likely because start of sentence
- $P(y_2 = \text{VBZ}|y_1 = \text{NNP})$ likely because verb often follows noun
- $P(y_3 = \text{NN}|y_2 = \text{VBZ})$ direct object follows verb, other verb rarely follows past tense verb (main verbs can follow modals though!)

Transitions in POS Tagging

NNP VBZ NN NNS CD NN
Fed raises interest rates 0.5 percent

- Should *y* be a single tag?
- Trigram model: y_1 = (<S>, NNP), y_2 = (NNP, VBZ), …
- P((VBZ, NN) | (NNP, VBZ)) — more context! Noun-verb-noun S-V-O
- Tradeoff between model capacity and data size

Estimating Transitions

NNP VBZ NN NNS CD NN
Fed raises interest rates 0.5 percent

- Similar to Naive Bayes estimation: maximum likelihood solution = normalized counts (with smoothing) read off supervised data
- P(tag | NN) = (0.5 </S>, 0.5 NNS)
- How to smooth?
- One method: smooth with unigram distribution over tags

$$P(\text{tag}|\text{tag}_{-1}) = (1-\lambda)\hat{P}(\text{tag}|\text{tag}_{-1}) + \lambda\hat{P}(\text{tag})$$

$\hat{P}$ = empirical distribution (read off from data)

Emissions in POS Tagging

NNP VBZ NN NNS CD NN
Fed raises interest rates 0.5 percent

- Emissions P(*x* | *y*) capture the distribution of words occurring with a given tag
- P(word | NN) = (0.05 person, 0.04 official, 0.03 government, 0.03 market …)
- When you compute the posterior for a given word's tags, the distribution favors tags that are more likely to generate that word

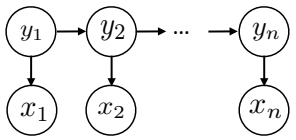
Estimating Emissions

NNP VBZ NN NNS CD NN
Fed raises interest rates 0.5 percent

- P(word | NN) = (0.5 interest, 0.5 percent) — hard to smooth!
- Can interpolate with distribution looking at word shape
  P(word shape | tag) (e.g., P(capitalized word of len >= 8 | tag))
- Alternative: use Bayes' rule $P(\text{word}|\text{tag}) = \frac{P(\text{tag}|\text{word})P(\text{word})}{P(\text{tag})}$
  - Fancy techniques from language modeling, e.g. look at type fertility — P(tag|word) is flatter for some kinds of words than for others)
- P(word|tag) can be a log-linear model — we'll see this in a few lectures

Inference in HMMs

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



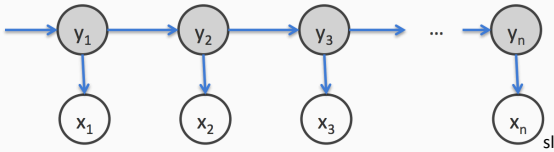
$$P(\mathbf{y}, \mathbf{x}) = P(y_1) \prod_{i=2}^{n} P(y_i|y_{i-1}) \prod_{i=1}^{n} P(x_i|y_i)$$

- Inference problem: $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \frac{P(\mathbf{y}, \mathbf{x})}{\cancel{P(\mathbf{x})}}$
- Exponentially many possible ***y*** here!
- Solution: dynamic programming (possible because of Markov structure!)
  - Many neural sequence models depend on entire previous tag sequence, need to use approximations like beam search

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

Transition probabilities Emission probabilities Initial probability



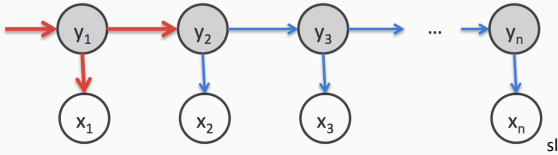
slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

The only terms that depend on y_1



slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

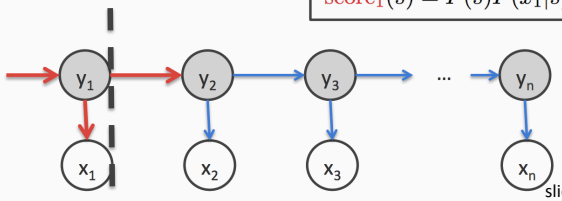
$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

Abstract away the score for all decisions till here into score

$$\text{score}_1(s) = P(s) P(x_1|s)$$



slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

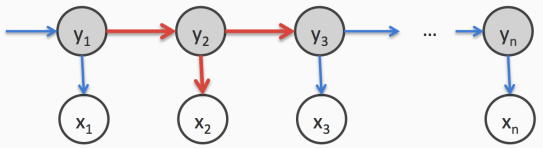
$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

Only terms that depend on y_2



slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

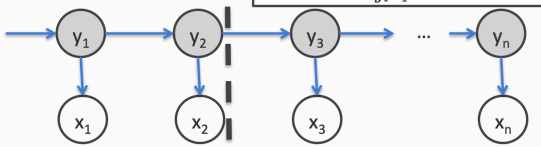
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \text{score}_2(y_2)$$

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



Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) P(y_1) P(x_1|y_1)$$

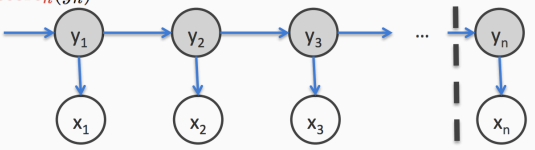
$$= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \max_{y_1} P(y_2|y_1) P(x_2|y_2) \text{score}_1(y_1)$$

$$= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1}) P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2) P(x_3|y_3) \text{score}_2(y_2)$$

$$\vdots$$

$$= \max_{y_n} \text{score}_n(y_n)$$



Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar

$$P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1,y_2,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

$$= \max_{y_3,\cdots,y_n} P(y_n|y_{n-1})P(x_n|y_n)\cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3)\text{score}_2(y_2)$$

$$\vdots$$

$$= \max_{y_n} \text{score}_n(y_n)$$

$$\text{score}_1(s) = P(s)P(x_1|s)$$

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

slide credit: Vivek Srikumar

1. Initial: For each state s, calculate

$$\text{score}_1(s) = P(s)P(x_1|s) = \pi_s B_{x_1,s}$$

2. Recurrence: For i = 2 to n, for every state s, calculate

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

$$= \max_{y_{i-1}} A_{y_{i-1},s} B_{s,x_i} \text{score}_{i-1}(y_{i-1})$$

3. Final state: calculate

$$\max_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}|\pi, A, B) = \max_{s} \text{score}_n(s)$$

π: Initial probabilities
A: Transitions
B: Emissions

This only calculates the max. To get final answer (*argmax*),

- keep track of which state corresponds to the max at each step
- build the answer using these back pointers

slide credit: Vivek Srikumar

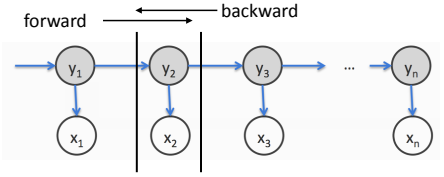
Forward-Backward Algorithm

- Compute marginal distributions $P(y_i = s|\mathbf{x})$
- Replace max with + everywhere, also run backward pass

$$\text{forward}_2(s)\text{backward}_2(s) = P(\mathbf{x}, y_2 = s)$$

$$P(y_2 = s|\mathbf{x}) \propto \text{forward}_2(s)\text{backward}_2(s)$$ ←i.e. normalize by P(**x**)

- Be careful not to double-count $P(x_2|y_2)$ when combining these!
- Store everything as log probabilities to avoid underflow



HMM POS Tagging

- Most frequent tag: ~90% accuracy
- Trigram HMM: ~95% accuracy / 55% on unknown words
- TnT tagger (tuned) HMM: 96.2% accuracy / 86.0% on unknown words
- Logistic regression P(t|w): 93.7% / 82.6% (*only* at current word)
- State-of-the-art (BiLSTM-CRFs): 97.5% / 89%+

Slide credit: Dan Klein

Errors

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	177	56	0	61	2	5	10	15	108	0	488
NN	244	0	103	0	12	1	1	29	5	6	19	525
NNP	107	106	0	132	5	0	7	5	1	2	0	427
NNPS	1	0	110	0	0	0	0	0	0	0	0	142
RB	72	21	7	0	0	16	138	1	0	0	0	295
RP	0	0	0	0	39	0	65	0	0	0	0	104
IN	11	0	1	0	169	103	0	1	0	0	0	323
VB	17	64	9	0	2	0	1	0	4	7	85	189
VBD	10	5	3	0	0	0	0	3	0	143	2	166
VBN	101	3	3	0	0	0	0	3	108	0	1	221
VBP	5	34	3	1	1	0	2	49	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

JJ/NN NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

Slide credit: Dan Klein / Toutanova + Manning (2000)

Remaining Errors

- Lexicon gap (word not seen with that tag in training) 4.5%
- Unknown word: 4.5%
- Could get right: 16% (many of these involve parsing!)
- Difficult linguistics: 20%

VBD / VBP? (past or present?)
They set up absurd situations, detached from reality

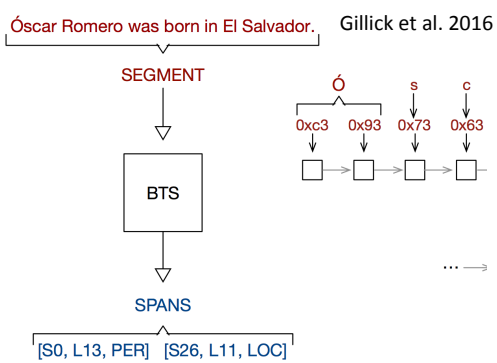
- Underspecified / unclear, gold standard inconsistent / wrong: **58%**

adjective or verbal participle? JJ / VBN?
a $ 10 million fourth-quarter charge against discontinued operations

Manning 2011 "Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?"

Other Languages

Language	CRF+	CRF	BTS	BTS*
Bulgarian	97.97	97.00	97.84	97.02
Czech	98.38	98.00	98.50	98.44
Danish	95.93	95.06	95.52	92.45
German	93.08	91.99	92.87	92.34
Greek	97.72	97.21	97.39	96.64
English	95.11	94.51	93.87	94.00
Spanish	96.08	95.03	95.80	95.26
Farsi	96.59	96.25	96.82	96.76
Finnish	94.34	92.82	95.48	96.05
French	96.00	95.93	95.75	95.17
Indonesian	92.84	92.71	92.85	91.03
Italian	97.70	97.61	97.56	97.40
Swedish	96.81	96.15	95.57	93.17
AVERAGE	96.04	95.41	95.85	95.06



Gillick et al. 2016

- Universal POS tagset (~12 tags), cross-lingual model works as well as tuned CRF using external resources

Next Time

- CRFs: feature-based discriminative models
- Structured SVM for sequences
- NER