Organizing Web Information

CS 728

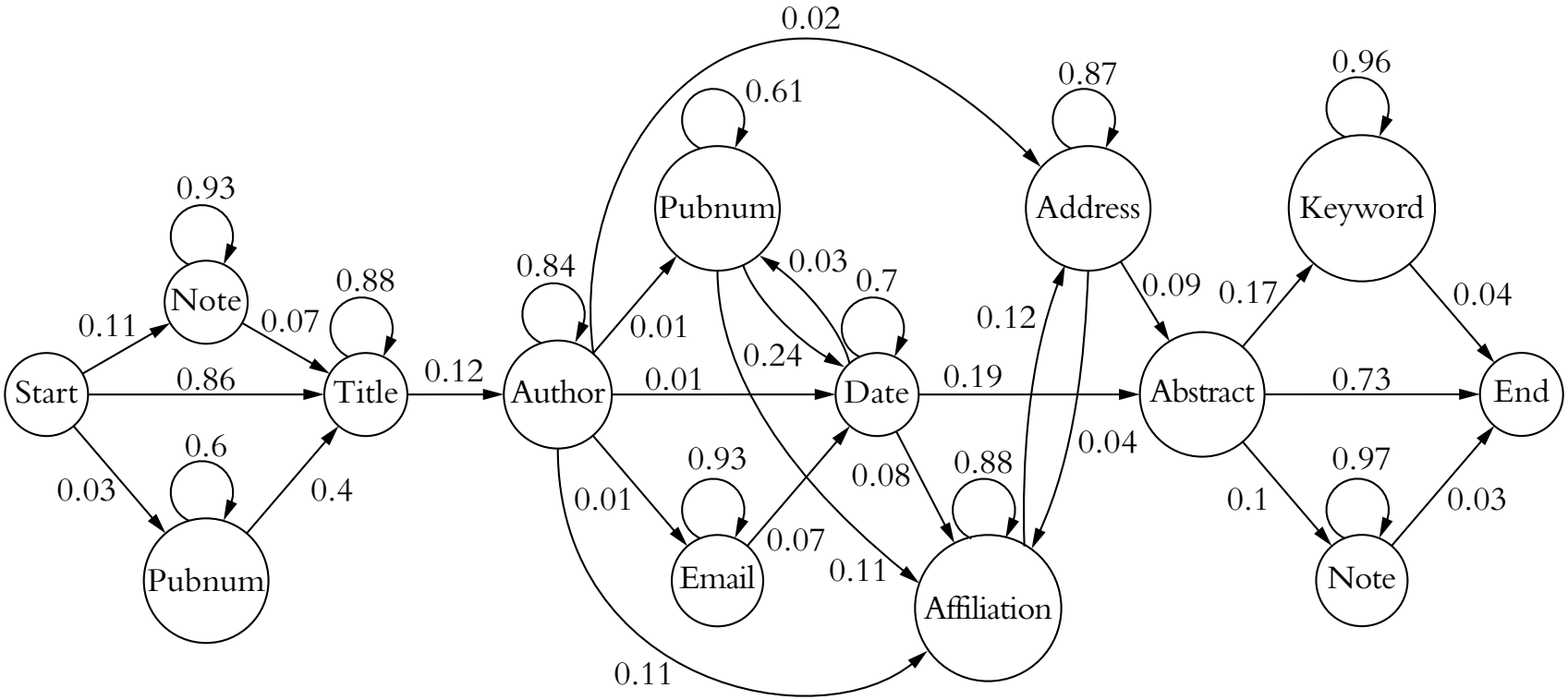
Soumen Chakrabarti
IIT Bombay
http://www.cse.iitb.ac.in/~soumen/

Sequence labeling

Tagging token spans

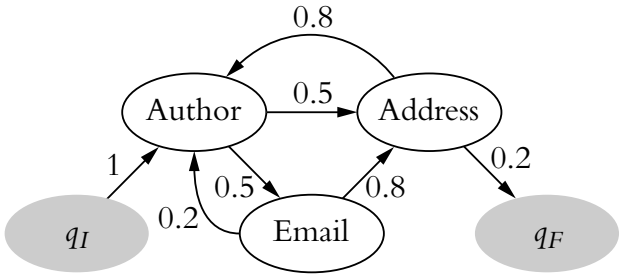
- Text modeled as a sequence of tokens
- Tokens may include punctuations, different kinds of spaces, etc.
- Restricted set of types or domains
  - Classic application in NLP: part-of-speech (POS) tagging, with token labels like NN, NNS, VB, VBD, VBP, PP$, JJ, JJR, RB, etc.
  - Named entity recognition (NER): person, place, organization, date, paper title, conference venue, journal name, page range, number and street name
- *Closed domain* means examples available for each label
- A test instance is an unlabeled token sequence
- The job is to mark each token with a label seen in training instances
- In NER, most tokens may have a default "none" label

Markov models

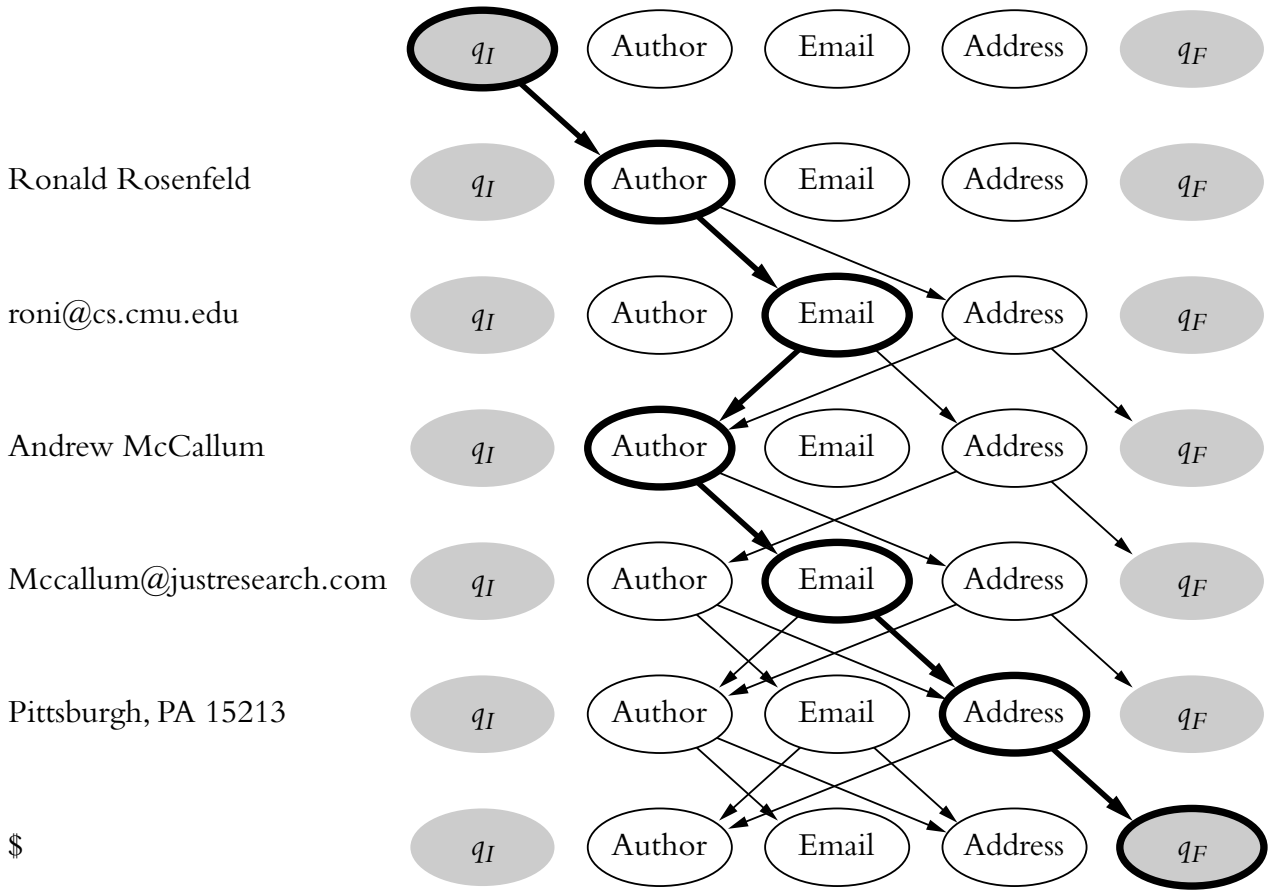


- System is in one of M states y_t
- In each state, emits x_t one of N symbols
- Then moves to one of M states y_{t+1}
- Many state transitions disallowed

Viterbi decoding



x1 Ronald Rosenfeld
x2 roni@cs.cmu.edu
x3 Andrew McCallum
x4 mccallum@justresearch.com
x5 Pittsburgh, PA 15213
x6 $



Dynamic programming

- Sequence of length T, pad with special positions 0 and $T+1$, let $t \in [0, T+1]$ be a token position
- Let $m \in [M]$ be a state; add special states $y_0 = q_I, y_{T+1} = q_F$ at positions 0 and $T+1$
- Define transitions to/from special states and emissions from special states suitably
- $\psi(m, m') = \Pr(m \to m')$ is the probability of a state transition from m to m'
- $\lambda(m, x) = \Pr(m \uparrow x)$ is the probability of emitting token x while in state m
- Let $V(t, m)$ be the largest probability of a state transition path ending in state m at position t
- Base case $V(0, y_0) = 1$

Dynamic programming (2)

- For time steps $t > 0$ and $m' \in [M]$:

$$V(t, m') = \max_{m \in [M]} V(t-1, m)\, \psi(m, m')\, \lambda(m', x_t)$$

- Highest probability path can be read off as $V(T+1, q_F)$
- In actual code, multiplying small probabilities will underflow rapidly, so use $\log \psi(m, m')$ and $\log \lambda(m, x)$

$$\begin{aligned} V(0, q_I) &= 0 \\ V(0, m) &= -\infty \quad \text{for } m \neq q_I \\ V(t, m') &= \max_{m \in [M]} V(t-1, m) + \log \psi(m, m') + \log \lambda(m', x_t) \end{aligned}$$

- Keep track of which m maximized $V(t, m')$, so we can recover the path

Max probability $\Leftrightarrow$ shortest path

- Each edge (u, v) has a probability $p(u, v)$
- Probability of a path is the product of edge probabilities
- The goal is to find the max probability path
- Instead, let the edge have a weight $w(u, v) = -\log p(u, v) \geq 0$
- Goal changes to conventional shortest path
- If there are k transitions from state m to m', $-\log p(m, m')$ is accumulated k times
- Similarly for symbol emission probabilities

▸ HW What if you wanted to list the top 10 shortest paths?

Estimating $\Pr(m \to m')$ and $\Pr(m \uparrow x)$

- If we had completely labeled data sets $(\vec{x}, \vec{y})$, then this amounts to just counting
  - The number of times symbol x was emitted while in state m
  - The number of times state m transitioned to state m'

  and then normalizing suitably to probabilities
- Care is required with smoothing counts for rare transition and emission events — we covered this in Web Search and Mining (A)
- Given incompletely or not labeled sequences, EM is a common technique:
  - Start with a "good enough" initial estimate of these probabilities
  - Use these to (re)label given sequences, getting distributions over states
  - Use these distributions to soft-count and update probability estimates

B-I-O and B-C-E-O state models

- States none, person, place, epoch
- Roy went to Rio de Janeiro in 1994
- Change state/label definitions to placeB, placeI, epochB, epochI, . . . , other=none
- "Begin", "In", "Other/out"
- Also used: begin-continue-end, other states
- Emission distributions can be fine tuned to (three kinds of) positions inside long mentions
- $B \to I$, $I \to I$, $I \to O$ transition probabilities can be tuned separately
- No $I \to B$ or $O \to I$ transitions allowed

Surface features

- Would like to think of x_t not as a symbol from a set of N symbols
- But as a feature vector with F features
- I.e. we are interested in various properties of token x_t rather than the identity of x_t itself
- $x_t[f]$ is the fth feature at position t
- Examples of features: `hasCap`, `isAllCap`, `isXxx`, `hasDigit`, `isAllDigits`, `isDDDD`, part of speech
- Does x_t start with an uppercase letter and end with "sky", "ski", or "jee"?
- Note that the identity of x_t itself can still be passed on through *lexicalized* features: "is x_t equal to *coffee*?"
- I.e., if there are 20,000 words in the vocabulary and 50 derived features, $F = 20050$

Limitations of generative models: feature dependence

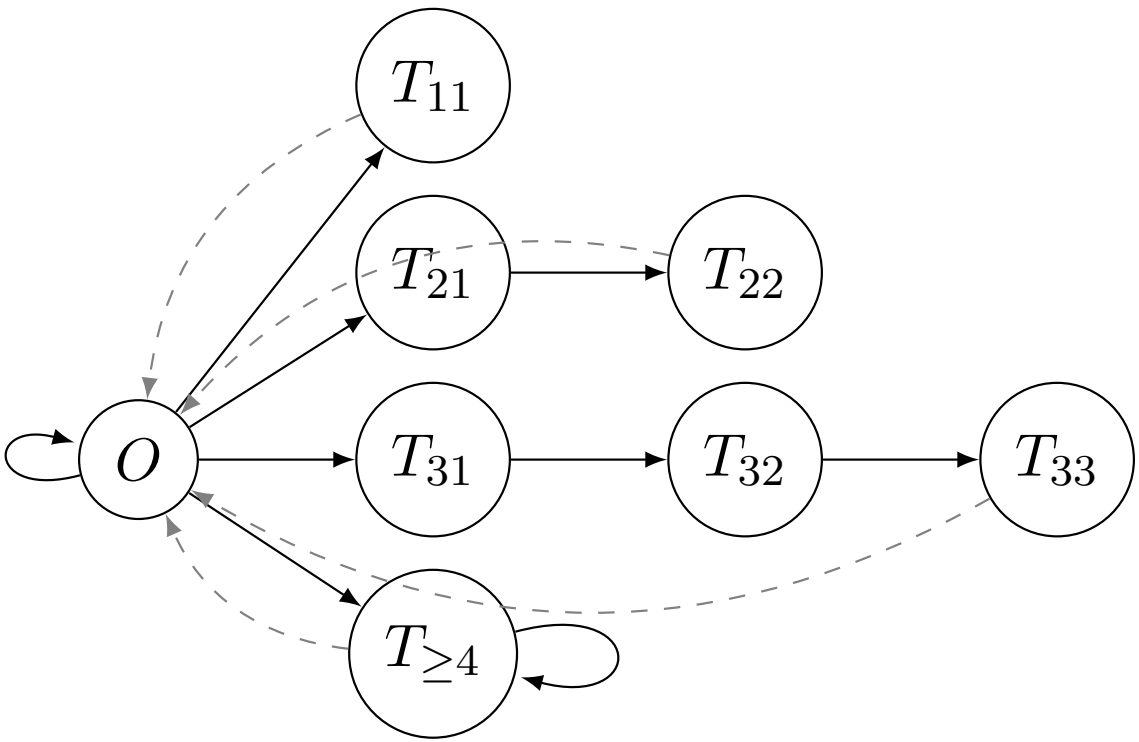
- Some of the F features are heavily correlated
  - `isAllCap` $\Rightarrow$ `hasCap`
  - `isDDDD` $\Rightarrow$ `isAllDigits`
  - `isProperNoun` almost always implies `isXxx`
- Standard generative HMM says

$$\Pr(\vec{x}, \vec{y}) = \Pr(y_0) \prod_{t=1}^{T} \Pr(y_t|y_{t-1}) \Pr(x_t|y_t)$$

- Now $\Pr(x_t|y_t)$ changes from univariate multinomial to a vector of (binary, highly correlated) features
- $\prod_{t=1}^{T} \Pr(y_t|y_{t-1}) \boxed{\prod_{f=1}^{F} \Pr(x_t[f]|y_t)}$ too crude
- Prefer to model $\Pr(\vec{y}|\vec{x})$ rather than model $\Pr(\vec{x}|\vec{y})$ and use Bayes rule (naive Bayes vs. logistic regression)

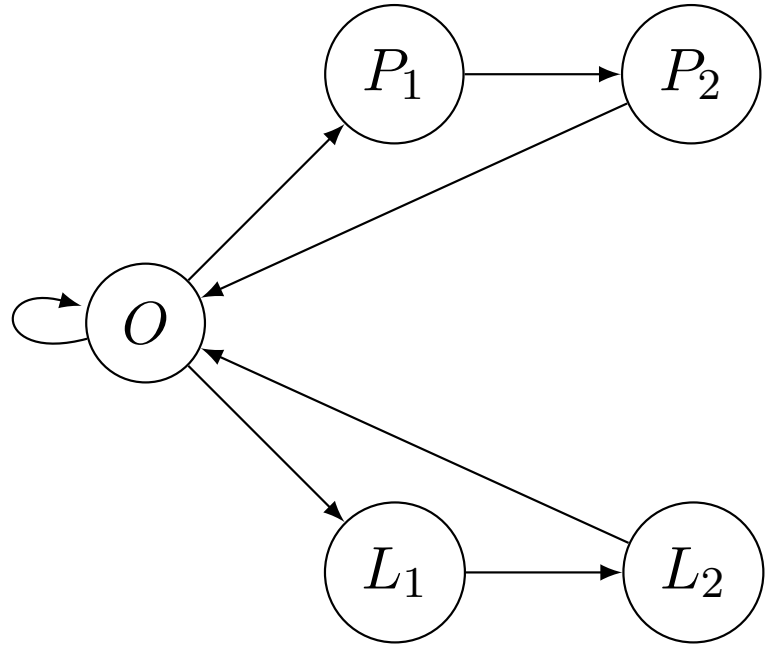
Limitations of generative models: segment lengths

- Long named entity spans like "paper title", "movie title", or "organization name" modeled as self-loop on corresponding states
- Implies that the number of tokens in such spans is geometrically distributed
- Assumption does not match data
- Can patch with a collection of chains; training may suffer



Limitations of generative models: Label bias[1]

- Labels other (O), person (P), location (L)
- Every person and location name has two tokens
- States specialized to $P_1, P_2; L_1, L_2$
- Harvey Ford: person 9 times, location 1 time
- Harvey Park: person 1 time, location 9 times



$\Longrightarrow$ No clue to choose correct edge out of O after seeing Harvey

$$\Pr(P_1|O, \text{Harvey}) = \Pr(L_1|O, \text{Harvey}) = 1/2$$
$$\Pr(P_2|P_1, *) = \Pr(L_2|L_1, *) = 1$$

- Normalized transition probability $\prod_t \Pr(y_t|y_{t-1})$ is culprit

[1] Example by Ralph Grishman

From local to global normalization

- Recall multinomial naive Bayes text classification

$$\Pr(y|x) \propto \Pr(y) \prod_f \Pr(f|y)^{x[f]} \propto \Pr(y) \exp\left[\sum_f x[f] \log \Pr(f|y)\right]$$

- Here $\Pr(f|y)$ is locally normalized: for each y, $\sum_f \Pr(f|y) = 1$
- Logistic regression replaced this with general weights

$$\Pr(y|x) \propto \exp(x \cdot w_y) = \exp\left[\sum_f x[f] w_y[f]\right] = \prod_f \exp\left[x[f] w_y[f]\right]$$

- No local normalization requirement on w_y; only $\sum_y \Pr(y|x)$ normalized if needed
- Will repeat this recipe for transitions and features

Transition and features for one step

- Design a function $\varphi : \mathcal{X} \times [M] \times [M] \to \{0,1\}^{MF+M^2}$

  initialize $A \leftarrow 0^{M \times F}, B \leftarrow 0^{M \times M}$

  $B[m, m'] \leftarrow 1$

  for each feature f **do**

  $A[m, f] \leftarrow x[f]$

  return $[A, B]$
- Typically called as $A_t, B_t \leftarrow \varphi(x_t, y_{t-1}, y_t)$
- Let model weights be $w = (\alpha, \beta)$ where $\alpha \in \mathbb{R}^{M \times F}, \beta \in \mathbb{R}^{M \times M}$
- The score for the t-th step is written as $\exp\Big[[A_t, B_t] \otimes [\alpha, \beta]\Big]$ where $\otimes$ is elementwise inner-product between matrices, i.e.,

$$\exp\left[\sum_{m,f} A_t[m,f]\alpha[m,f] + \sum_{m,m'} B_t[m,m']\beta[m,m']\right]$$

$$= \exp\left[\sum_f x_t[f]\,\alpha[m,f] + \beta[y_t, y_{t-1}]\right] = e^{\beta[y_t, y_{t-1}]} \prod_f \exp(\cdots)$$

Summing over steps t

- Because of the log-linear form, we can write

$$A = \sum_t A_t, \quad B = \sum_t B_t, \quad \text{and then}$$

$$\Pr(\vec{y}|\vec{x}) \propto \exp\Big[[A, B] \otimes [\alpha, \beta]\Big] = \prod_t e^{\beta[y_t, y_{t-1}]} \prod_{t,f} \exp(\cdots)$$

- $A[m, f]$ is the number of times feature f was "fired" while in state m
- $B[m, m']$ is the number of transitions from m' to m
- To ease notation, let

$$[A, B] = \phi(\vec{x}, \vec{y}) = \sum_t \varphi(x_t, y_{t-1}, y_t)$$

$$w = [\alpha, \beta]$$

Summing over steps t (2)

- Summarizing ...
- $\vec{x} = (x_t : t = 1, \ldots, T)$ is the sequence of visible symbols
- Each position is associated with F binary (say) features
- $\vec{y} = (y_t : t = 1, \ldots, T)$ is the label/state sequence
- Suppose there are M states; $m \in [1, M]$
- Define a feature map $\phi(x, y) \in \mathbb{R}^d$
- From labeled training data $x^\ell, y^\ell : \ell = 1, \ldots, L$, learn a model $w \in \mathbb{R}^d$
- Given test sequence $\vec{x}$, predict label sequence $\arg\max_{\vec{y}} w^\top \phi(\vec{x}, \vec{y})$
- Enumerating all $\vec{y}$ not practical, but for chain (and some other) dependency graphs, dynamic programming suffices

Inspecting more/all of $\vec{x}$

- Can generalize to $\varphi(t, \vec{x}, m', m)$ to look at more or all of $\vec{x}$, provided we retain $m' = y_{t-1}$ and $m = y_t$
- Can also change the behavior of φ with t, e.g., make some states, transitions or emissions more or less likely near specific times/offsets t
- Another application in the next slide
- This sort of enhancement usually needs massive state-space blowup in conventional HMMs

Features between $x_{t\pm1}$ and y_t

- So far we have limited interaction between $\vec{x}$ and $\vec{y}$ to individual positions, e.g., x_t and y_t
- States none, person, place, epoch
  - Roy went to Rio de Janeiro in 1994
- "at" or "to" often precede place
- "on" or "in" often precede epoch
- But the state of *at, to, on, in* are all 'none'
- Define features between $x_{t\pm1}$ and y_t:
  - $x_{t-1} =$ "at" $\wedge\, y_t =$ placeB
  - $x_{t+1} =$ "said" $\wedge\, y_t =$ personI
- Effectively multiplies F by some (small) factor

Inference

- Given a partial label sequence $(y_1, \ldots, y_{t-1}, y_t = m)$ of length t
- Let $v(t, m)$ be the maximum value of $w^\top \phi(\vec{x}, (y_1, \ldots, y_{t-1}, y_t = m))$ over all possible states $y_1, \ldots, y_{t-1}$ and y_t pinned to m
- Base case $v(0, m) = 0$ for all m
- Recurrence step for $t > 0$:

$$v(t, m) = \max_{m'} v(t-1, m') + w \cdot \varphi(t, x_t, m', m)$$

- Final solution is (the path traced to get from y_0 at time 0 to) $\max_m v(T, m)$
- Viterbi's dynamic programming algorithm

Discriminative training of w

- For each sequence $\vec{x}^\ell$, there is (say) one correct labeling $\vec{y}^\ell$; all other (exponentially many) labelings $\vec{y} \neq \vec{y}^\ell$ are incorrect
- Want to fit w such that

$$\forall \ell, \forall \vec{y} \neq \vec{y}^\ell : \; w^\top \phi(\vec{x}^\ell, \vec{y}^\ell) > w^\top \phi(x^\ell, \vec{y})$$

- The worse $\vec{y}$ is, compared to $\vec{y}^\ell$, the bigger we want the gap to be
- Loss function $\Delta(\vec{y}^\ell, \vec{y}) \geq 0$; $\Delta(\vec{y}, \vec{y}) = 0$
- E.g., Hamming loss

$$\forall \ell, \forall \vec{y} \neq \vec{y}^\ell : w^\top \phi(\vec{x}^\ell, \vec{y}^\ell) \geq w^\top \phi(\vec{x}^\ell, \vec{y}) + \Delta(\vec{y}^\ell, \vec{y})$$

- Allow a slack $\xi_\ell \geq 0$:

$$\forall \ell, \forall \vec{y} \neq \vec{y}^\ell : w^\top \phi(\vec{x}^\ell, \vec{y}^\ell) + \xi_\ell \geq w^\top \phi(\vec{x}^\ell, \vec{y}) + \Delta(\vec{y}^\ell, \vec{y})$$

Discriminative training of w (2)

- Objective has two parts:
  - $\sum_\ell \xi_\ell$ representing upper bound on training loss
  - $\frac{1}{2}\|w\|_2^2$, the model complexity

  usually balanced with a tuned magic constant C

$$\min_{\xi \geq \vec{0}; w} \frac{1}{2}\|w\|_2^2 + C \sum_\ell \xi_\ell \quad \text{s.t.}$$

$$\forall \ell, \forall \vec{y} \neq \vec{y}^\ell : w^\top \boxed{\phi(\vec{x}^\ell, \vec{y}^\ell) - \phi(\vec{x}^\ell, \vec{y})} \geq \Delta(\vec{y}^\ell, \vec{y}) - \xi_\ell$$

- $M^T - 1$ constraints per training sequence!
- StructSVM to the rescue

(Loss augmented) inference

- After w is trained, the inference problem is to find $\arg\max_{\vec{y} \in \mathcal{Y}} w^\top \phi(\vec{x}, \vec{y})$ for test x
- M states, sequence length $T \implies |\mathcal{Y}| = M^T$
- Viterbi dynamic programming takes $O(M^2 T)$ time
- Loss augmented inference is to find $\tilde{y} = \operatorname{argmax}_{\vec{y}} w^\top \phi(\vec{x}^\ell, \vec{y}) + \Delta(\vec{y}^\ell, \vec{y})$ for training instance $(\vec{x}^\ell, \vec{y}^\ell)$
- (I.e., find bad $\tilde{y}$ whose loss and score are both large)
- If $w \cdot \phi(\vec{x}^\ell, \tilde{y}) + \Delta(\vec{y}^\ell, \tilde{y}) < w \cdot \phi(\vec{x}^\ell, \vec{y}^\ell)$, no bad $\tilde{y}$ really exists, can terminate
- Otherwise, make a new constraint using $\tilde{y}$ and optimize for w again

Decomposable loss examples

- Can also be solved via dynamic programming for decomposable (and some non-decomposable) Δ
- All-or-nothing $\Delta_{0/1}(\vec{y}, \vec{y}') = [\![\vec{y} \neq \vec{y}']\!]$
- Hamming $\Delta_h(\vec{y}, \vec{y}') = \sum_t [\![y_t \neq y'_t]\!]$
- At most three errors $\Delta_{\leq 3}(y, y') = [\![\Delta_h(y, y') > 3]\!]$
- Dimishing marginal annoyance:
  - In a sequence of length T there can be $k \in [0, T]$ mistakes
  - Let the loss be $\Gamma\left(\sum_t [\![y_t \neq y'_t]\!]\right)$
  - E.g. $\Gamma(\bullet) = \sqrt{\bullet}$ or $\log(1 + \bullet)$
- $V(t, m, k)$ is the largest (loss augmented) score of ending in state m at time t with k mistakes
- $V(0, y_0, 0) = 0$, $V(0, \neq y_0, *) = -\infty$
- $V(t, *, > t) = -\infty$ for all t (cannot make $> t$ mistake up to time t)

Decomposable loss examples (2)

- Let $\vec{y}^*$ be the gold sequence
- Suppose $y_t^* = m$, i.e., no mistake at time t

$$V(t, m, k) = \max_{m'} \left\{ V(t-1, m', k) + w \cdot \varphi(x_t, m', m) \right\}$$

- Suppose $y_t^* \neq m$, i.e., mistake at time t

$$V(t, m, k) = \max_{m'} \Big\{ V(t-1, m', k-1) + w \cdot \varphi(x_t, m', m) + \Gamma(k) - \Gamma(k-1) \Big\}$$

- Because loss is nonlinear, must remember and take off the previous loss $\Gamma(k-1)$ and add current accumulated loss $\Gamma(k)$

▸ HW Check, complete and implement

- Useful for robust learning with some fraction of 'hopeless' training sequences that just add a noise floor to training loss

Conditional probability model

- Another option is to model a conditional probability: $\Pr(\vec{y}|\vec{x}) \propto \exp(w^\top \phi(\vec{x}, \vec{y}))$
- Sometimes written as $\Pr(\vec{y}|\vec{x}; w)$ to make model w explicit
- Inference is $\arg\max_{\vec{y}} \Pr(\vec{y}|\vec{x}) = \arg\max_{\vec{y}} w^\top \phi(\vec{x}, \vec{y})$ which is exactly the same as in max-margin training
- Using a normalizing factor $Z_w(\vec{x}) = \sum_{\vec{y}} \exp(w^\top \phi(\vec{x}, \vec{y}))$, we can write $\Pr(\vec{y}|\vec{x}) = \exp(w^\top \phi(\vec{x}, \vec{y}))/Z_w(x)$
- Training w amounts to optimizing

$$\arg\max_w \prod_\ell \Pr(\vec{y}^\ell|\vec{x}^\ell) = \arg\max_w \sum_\ell \log \Pr(\vec{y}^\ell|\vec{x}^\ell)$$
$$= \arg\max_w \sum_\ell w^\top \phi(\vec{x}^\ell, \vec{y}^\ell) - \log Z_w(\vec{x}^\ell)$$

- Concave, global maximum, use Newton method

Computing $\nabla_w \log Z_w(x)$

- First let us find $Z_w(\vec{x})$ in polynomial time
- Recall $\phi(\vec{x}, \vec{y}) = \sum_{1 \le t \le T} \varphi(x_t, y_{t-1}, y_t)$
- Define partial sums

$$\phi_{[1:t]}(\vec{x}[1:t], \vec{y}[1:t]) = \sum_{1 \le \tau \le t} \varphi(x_\tau, y_{\tau-1}, y_\tau) \quad \text{and}$$

$$\alpha(t, m) = \sum_{\gamma_1, \ldots, \gamma_{t-1}} \exp\Big(w \cdot \phi_{[1:t]}\big(\vec{x}[1:t], (\gamma_1, \ldots, \gamma_{t-1}, m)\big)\Big)$$

- By definition, $Z_w(\vec{x}) = \sum_m \alpha(T, m)$
- Base case $\alpha(0, m) = 1$ for all m
- Recurrence (last transition $m' \to m$):

$$\alpha(t, m) = \sum_{m'} \alpha(t-1, m') \exp\big(w \cdot \varphi(x_t, m', m)\big)$$

Computing $\nabla_w \log Z_w(x)$ (2)

- $\nabla_w \log Z_w(x) = \frac{1}{Z_w(x)} \nabla_w Z_w(x) =$
  $\frac{1}{Z_w(x)} \sum_{\vec{y}} \nabla_w \exp(w \cdot \phi(\vec{x}, \vec{y})) = \frac{1}{Z_w(x)} \sum_{\vec{y}} \phi(\vec{x}, \vec{y}) \exp(w \cdot \phi(\vec{x}, \vec{y}))$
- Btw,
  $\frac{1}{Z_w(x)} \sum_{\vec{y}} \phi(\vec{x}, \vec{y}) \exp(w \cdot \phi(\vec{x}, \vec{y})) = \sum_{\vec{y}} \phi(\vec{x}, \vec{y}) \frac{\exp(w \cdot \phi(\vec{x}, \vec{y}))}{Z_w(x)} =$
  $\sum_{\vec{y}} \phi(\vec{x}, \vec{y}) \Pr(\vec{y}|x; w) = \mathbb{E}_{\Pr(\vec{y}|\vec{x};w)} \phi(\vec{x}, \vec{y})$
- Note that this is an expected feature vector
- We have already computed $Z_w(\vec{x})$, now we need to compute $\sum_{\vec{y}} \phi(\vec{x}, \vec{y}) \exp(w \cdot \phi(\vec{x}, \vec{y}))$ efficiently
- Again define a partial sum over positions

$$\eta(t, m) = \sum_{\gamma_1, \ldots, \gamma_{t-1}} \phi_{[1:t]}(\vec{x}[1:t], (\gamma_1, \ldots, \gamma_{t-1}, m)) \exp\big(w \cdot \phi_{[1:t]}(\vec{x}[1:t], (\gamma_1, \ldots, \gamma_{t-1}, m))\big)$$

- By definition, our goal is $\sum_m \eta(T, m)$
- Base case $\eta(0, m) = \vec{0}$ for all m

Computing $\nabla_w \log Z_w(x)$ (3)

- By unrolling out once, $\eta(t, m) =$

$$\sum_{m'} \sum_{\gamma[1:t-2]} \left(\phi_{[1:t-1]}(\vec{x}[1:t-1], \underline{\gamma_1, \ldots, \gamma_{t-2}, m'}) + \varphi(x_t, m', m) \right)$$

$$\times \exp\left[w \cdot (\text{same})\right]$$

HW Simplifying, we get the recurrence

$$\eta(t, m) = \sum_{m'} \Big[\eta(t-1, m') + \alpha(t-1, m') \varphi(t, x_t, m', m) \Big] e^{w \cdot \varphi(x_t, m', m)}$$

- Now we can do gradient descent, e.g., AdaGrad
- Given global convexity, can take advantage of second order methods like L-BFGS

Modeling long-range influence

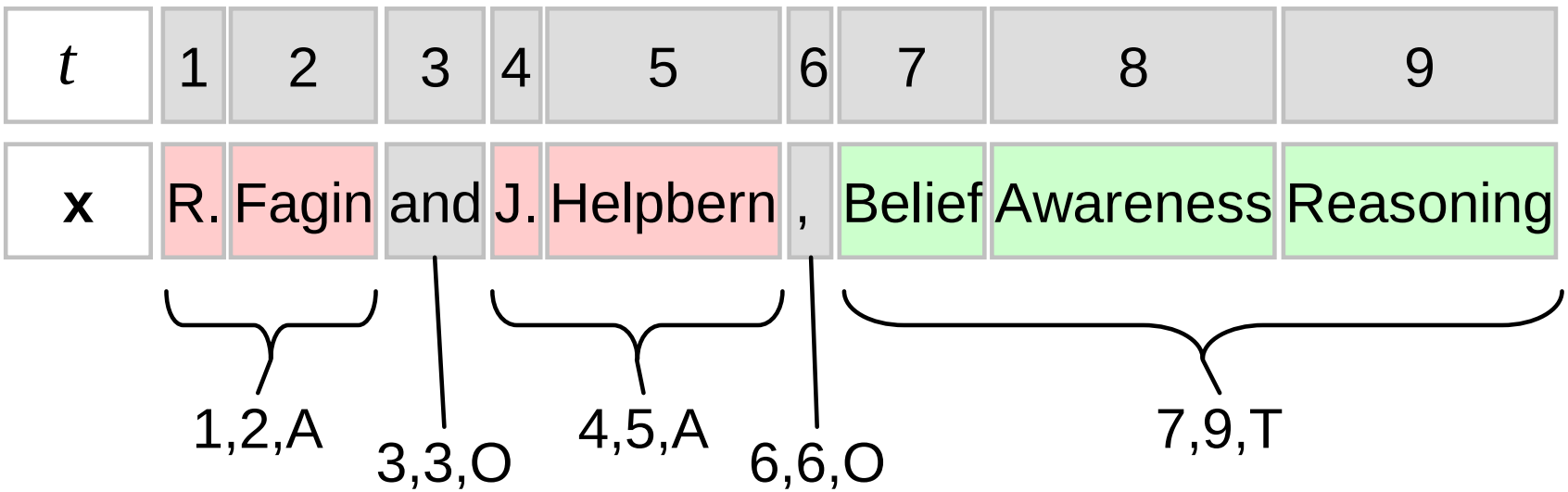
- Long segments — can still do dynamic programming, at increased cost
- Can extend from linear chains to trees without pain — (probabilistic) context free grammars
- Same token distributed through a document should get correlated labels — general graph, intractable, can only approximate

Segment CRF motivation

- Suppose the "paper title" state has a self-loop with probability 0.9
- Then the number of tokens in a paper title follows a geometric distribution with mean 10
- Real data may not support a geometric distribution
- Many long text fields like book or movie title comprise ordinary words (e.g., Gone with the Wind)
- Often easier to recognize because of close phrase match with a dictionary
- Or anomalous words just before and after:
  . . . the script for [GWTW] , he had no idea . . .
  for Gone and wind, he
- Can't model this at a single token level

Segment CRF model

- A segment s_j is defined by start token offset ℓ_j and end token offset u_j
- y_t replaced with $s_j = \langle \ell_j, u_j, y_j \rangle$
- Feature function extended to $\varphi(\ell, u, \vec{x}, m', m)$
- Usually instantiated to $\varphi(\ell_j, u_j, x_{\ell_j} \ldots x_{u_j}, y_{j-1}, y_j)$
- Vector of segments called $\vec{s}$ similar to $\vec{y}$



Inference for segment CRF

- Suppose J is the length of the longest allowed segment
- Let $v(t, m)$ be the best score of segmentations ending at t with label m
- $v(0, m) = 0$ for all m
- If $t < 0$ then $v(t, m) = -\infty$
- In general for $t > 0$, $v(t, m) =$

$$\max_{m'} \max_{t'=t-J}^{t-1} v(t', m') + w \cdot \varphi(t' + 1, t, \vec{x}, m', m)$$

Limitations of linear segmentation

- BizContact $\rightarrow$ BizName Address BizPhone
- PersonalContact $\rightarrow$ PersonName Address HomePhone
- If we see a 1-800 number, the name is more likely to be a business name
- Conversely, "Associates" in name makes the phone number more likely to be a business number
- Business numbers are more likely to have "1-800" in them

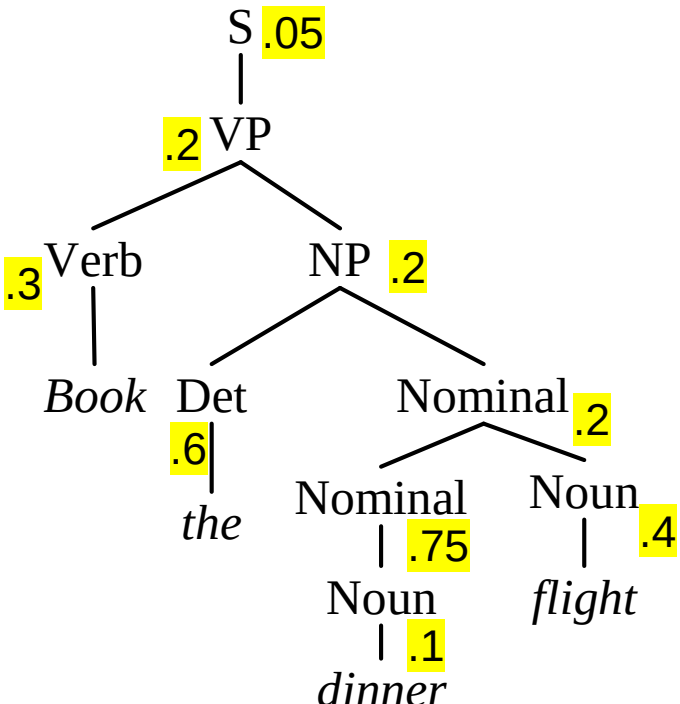
Fred Jones	Fred Jones and Associates
10 Main St.	10 Main St.
Cambridge, MA 02146	Cambridge, MA 02146
(425) 994-8021	1-800-555-1212

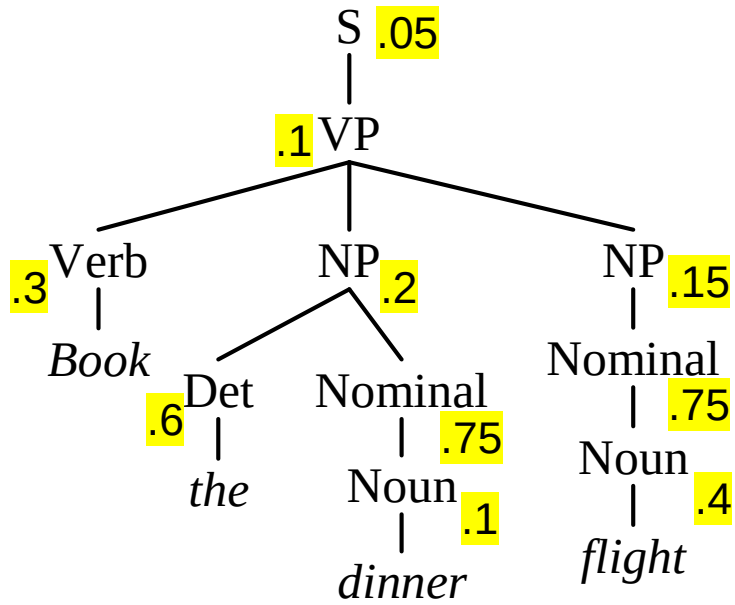
Probabilistic context free grammar (PCFG)

- A context-free grammar (CFG) is a 4-tuple $G = (N, \Sigma, R, S)$ where:
  - N is a finite set of non-terminal symbols.
  - Σ is a finite set of terminal symbols.
  - R is a finite set of rules of the form $X \to Y_1 Y_2 \dots Y_n$, where $X \in N$, $n \geq 0$, and $Y_i \in N \cup \Sigma$ for $i = 1, \dots, n$. For simplicity we will assume that $n = 1$ and $n = 2$ for productions to terminals and non-terminals respectively.
  - $S \in N$ is a distinguished start symbol.
- A PCFG, in addition, has a parameter $q(\alpha \to \beta) \geq 0$ for each rule $\alpha \to \beta \in R$
- Interpreted as the conditional probability of choosing this rule in a left-most derivation, given that the non-terminal being expanded is α
- For any $X \in N$, $\sum_\beta q(X \to \beta) = 1$

PCFG example

		Rules	q
S	$\rightarrow$	VP	.05
VP	$\rightarrow$	Verb NP	.2
VP	$\rightarrow$	Verb NP NP	.1
NP	$\rightarrow$	Nominal	.15
NP	$\rightarrow$	Det Nominal	.2
Nominal	$\rightarrow$	Nominal Noun	.2
Nominal	$\rightarrow$	Noun	.75
Verb	$\rightarrow$	*book*	.3
Det	$\rightarrow$	*the*	.6
Noun	$\rightarrow$	*dinner*	.1
Noun	$\rightarrow$	*flight*	.4





PCFG inference

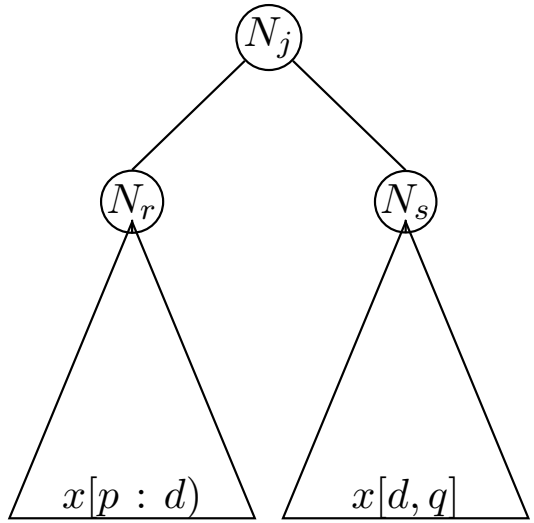
- What is the probability of a sentence given a PCFG, over all possible parse trees?
- What is the most likely parse tree for a sentence?
- For simplicity consider the retricted grammar

$$N_i \to w_j, \qquad N_i \to w_j N_k$$

- Let input sequence be $x_1, \ldots, \underbrace{x_p, \ldots, x_q}_{\text{inside}}, \ldots, x_T$
- Suppose inside span was produced from nonterminal N_j
- Inside probability is $\beta_j(p, q) = \Pr(x_{[p:q]} | N_j)$

CYK[2] dynamic programming algorithm

- Base case $\beta_j(k,k) = \Pr(x_k|N_j) = \Pr(N_j \to x_k)$
- General case $\beta_j(p,q)$ can be decomposed as
  - Pick two nonterminals to produce $N_j \to N_r N_s$
  - Pick a split point $[p,d)$ and $[d,q]$
  - N_r (eventually) produces $x_{[p,d)}$
  - N_s (eventually) produces $x_{[d,q]}$

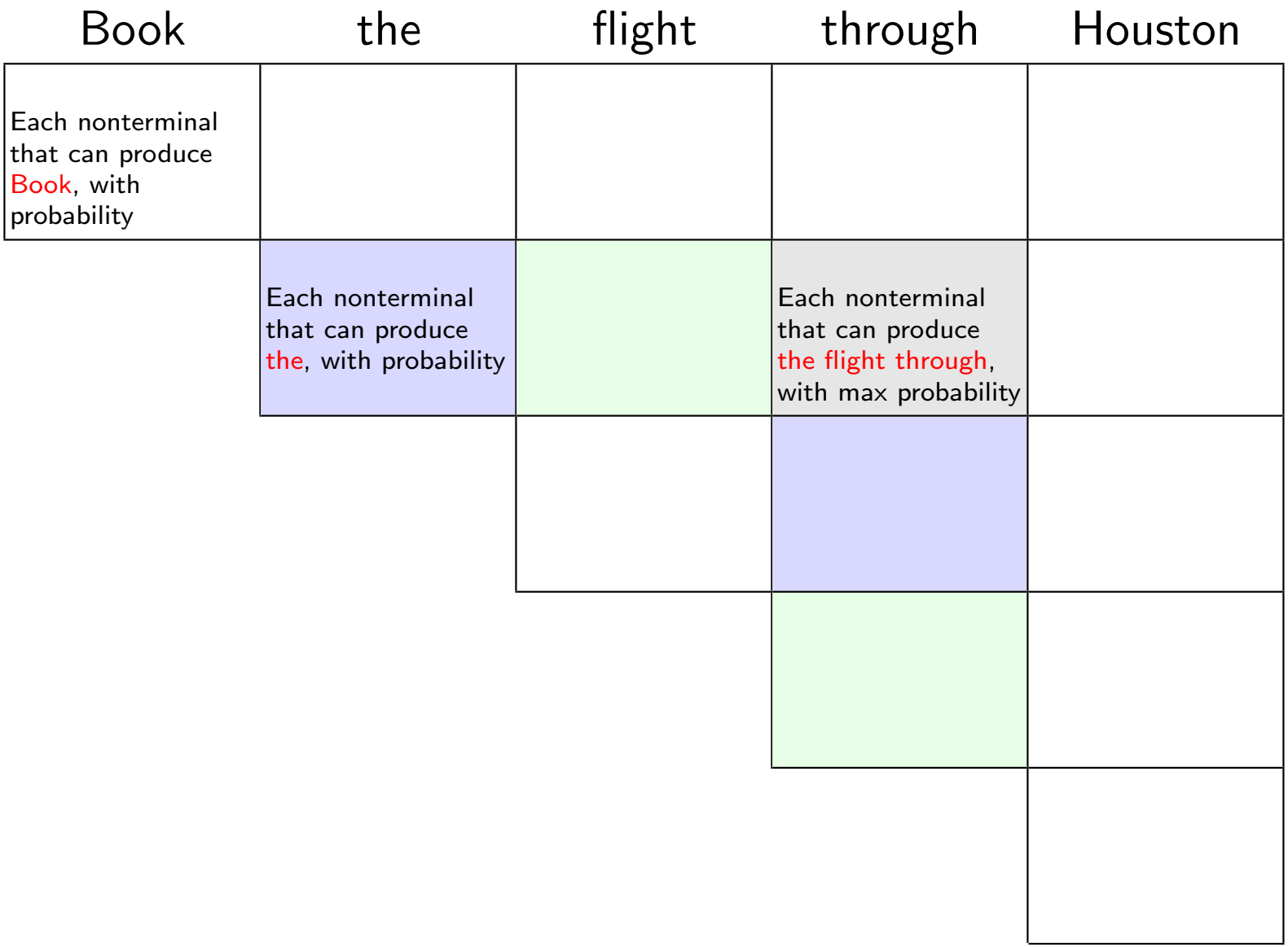


$$\beta_j(p,q) = \sum_{r,s} \sum_{p \le d \le q} \Pr(N_j \to N_r N_s) \beta_r(p,d) \beta_s(d+1,q)$$

- Probability of the entire sentence is $\beta_{\text{root}}(1,T)$
- Time $T^3 R$ where R is the number of rules in grammar
- For single best parse, replace sums above with max
- Rule probabilities $\Pr(N \to \zeta)$ estimated from fully labeled data as in HMMs

[2] Cocke-Kasami-Younger

The chart


Book
the
flight
through
Houston
Each nonterminal that can produce Book, with probability
Each nonterminal that can produce the, with probability
Each nonterminal that can produce the flight through, with max probability

Discriminative CFGs

- Replace the probabilistic form of $\beta_j(p, q)$ with a score computed from a feature vector and a model weight vector
- Let R_k be the rule $N_i \rightarrow \zeta_j$, where ζ_j is a sequence of terminals and nonterminals
- Each rule R_k has associated model weight vector $\boldsymbol{\lambda}_k \in \mathbb{R}^F$, say
- From $\vec{x}$, span $[p, q]$, and rule R_k, we extract feature vector $\boldsymbol{\phi}_k(\vec{x}, p, q) \in \mathbb{R}^F$ as well

$$S(N_i \rightarrow \zeta_j, p, q) = \boldsymbol{\lambda}_k \cdot \boldsymbol{\phi}_k(\zeta_j, \vec{x}, p, q)$$

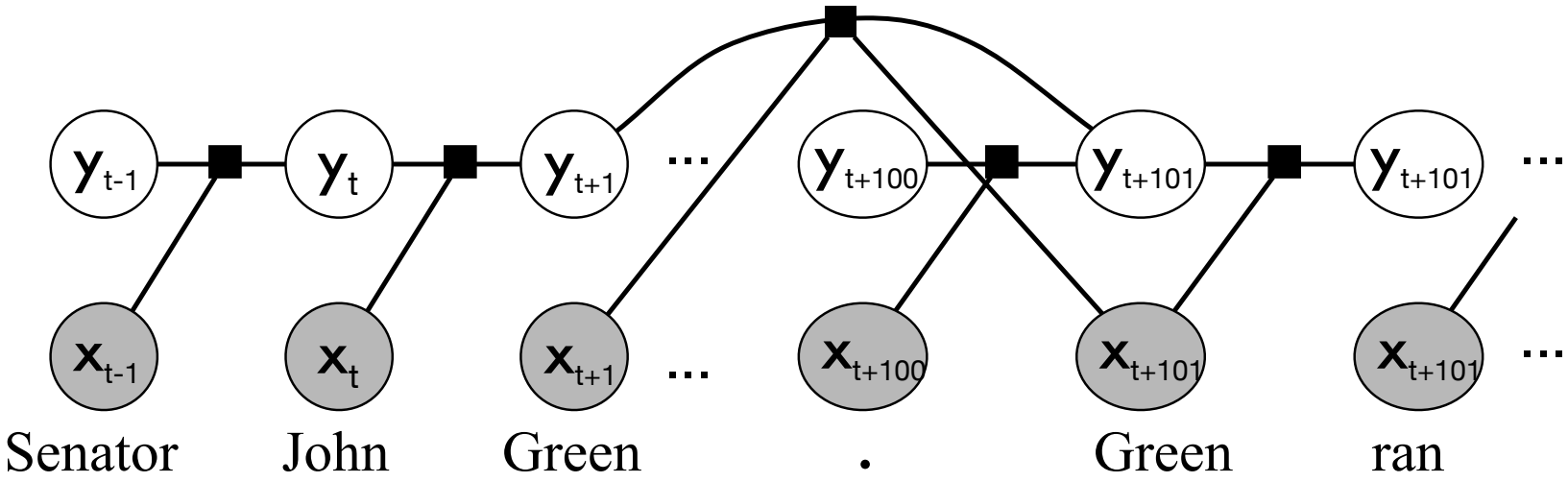
- E.g., the terminals of ζ_j will be compared to parts of $\vec{x}$ to fire some features
- There may be features for each (i, r) pair where nonterminal N_r occurs in ζ_j
- As in conditional Markov sequence models, it is ok for $\boldsymbol{\phi}$ to look at $\vec{x}$ outside the $[p, q]$ span

Inference and training

- MAP inference is as in generative PCFGs
- Here "$\vec{y}$" is replaced by a parse tree expressed in a suitable structured label format
- For discriminative training, must assert constraints corresponding to "all wrong parses" — difficult
- Collin's averaged perceptron:

for each instance $(\vec{x}, y^*)$ **do**
  let $\hat{y}$ be the best parse of $\vec{x}$ using $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_k\}$
  if $\hat{y} \neq y^*$ **then**
    for each rule R_k used in $\hat{y}$ but not y^* **do**
      if feature f is active/fired in $\vec{x}$ **then**
        $\boldsymbol{\lambda}_k[f] \leftarrow \boldsymbol{\lambda}_k[f] - 1$
    for each rule R_k used in y^* but not $\hat{y}$ **do**
      if feature f is active/fired in $\vec{x}$ **then**
        $\boldsymbol{\lambda}_k[f] \leftarrow \boldsymbol{\lambda}_k[f] + 1$

Skip-chain CRFs



- The same word within a document is likely to have the same label
- Not universally true, can only encourage but not enforce
- "Green party nominees like John Green did not agree."
- Express potential dependencies between distant occurrences x_t, y_t and $x_{t'}, y_{t'}$ using a factor node ■
- Can also use factor nodes to couple y_t, x_t, y_{t+1}
- Will explain shortly how to use factor nodes in inference

Factor graph

- Bipartite graph, factor layer F, variable layer V
- Assume all variables unobserved (will fix later)
- Factors denoted $a, b, \ldots$, variables $u, v, \ldots$
- Variable neighbors of factor a denoted $N(a)$
- Factor neighbors of variable u denoted $N(u)$
- $\vec{y}$ is an assignment of values to all variables
- u takes values from $\mathcal{Y}_u$
- If $S \subseteq V$ is a subset of variables, $\vec{y}_S$ is a vector of values for variables in S
- S except variable v is denoted $S \setminus v$
- E.g., $\vec{y}_{N(a)} \in \mathcal{Y}_{N(a)} = \times_{u \in N(a)} \mathcal{Y}_u$ is a cartesian product of domains
- Associated with each factor a is a function $\Psi_a : \mathcal{Y}_{N(a)} \to \mathbb{R}_+$

Factor graph (2)

- Joint distribution over $\vec{y}$ is written in a factored form as $\Pr(\vec{y}) \propto \prod_a \Psi_a(\vec{y}_{N(a)})$
- (Will see later how Ψ can be parameterized)

Marginals

- Given a specific value $\gamma \in \mathcal{Y}_u$ for variable u, what is the marginal probability $\Pr(Y_u = \gamma)$?
- If the number of variables is small, this can be enumerated $\sum_{\vec{y} \in \mathcal{Y}_{V \setminus u}} \Pr(Y_{V \setminus u} = \vec{y}, Y_u = \gamma)$
- Prohibitive for large V, factor graph helps us evaluate
- Exact if at most one cycle, effective heuristic otherwise

Messages

- $m_{u \to a}$ is a message from variable u to factor a
- $n_{a \to u}$ is a message from factor a to variable u
- Both messages are functions mapping $\mathcal{Y}_u$ to $\mathbb{R}_+$
- I.e., a message is a table
- For each $\gamma \in \mathcal{Y}_u$, nodes a and u will send a positive scalar value to each other
- Usually written as $m_{u \to a}(y_u)$ and $n_{a \to u}(y_u)$
- Where y_u is a specific value like γ that Y_u, the random variable, can take
- In step k,

$$m^{(k)}_{u \to a}(y_u) = \prod_{b \in N(u) \setminus a} n^{(k-1)}_{b \to u}(y_u)$$

$$n^{(k)}_{a \to u}(y_u) = \sum_{\vec{y} \in \mathcal{Y}_{N(a) \setminus u}} \Psi(\vec{y}, y_u) \prod_{v \in N(a) \setminus u} m^{(k)}_{v \to a}(y_v)$$

Messages (2)

- Note y_v is bound by the choice of $\vec{y} \in \mathcal{Y}_{N(a)\setminus u}$
- m expressed in terms of n and vice versa
- Converges to correct solution if the graph has at most one cycle
- Called the "sum product" algorithm
- Resemblance to Kleinberg's hubs and authorities iterations

Max product algorithm

- Recover single variable marginal as $\Pr(y_u) \propto \prod_{a \in N(u)} n_{a \to u}(y_u)$
- Clearly, different to ask for $\arg\max_{\vec{y}} \Pr(\vec{y})$ where $\Pr(\vec{y})$ given as product of factors
- The latter can be solved by max-product belief propagation:

$$n_{a \to u}^{(k)}(y_u) = \max_{\vec{y} \in \mathcal{Y}_{N(a) \setminus u}} \Psi(\vec{y}, y_u) \prod_{v \in N(a) \setminus u} m_{v \to a}^{(k)}(y_v)$$

- For proofs see books by Koller and Friedman, or Bishop

Parameterizing Ψ in factors

- Commonly used model

$$\Psi_a(\vec{y}_{N(a)}) = \exp\left(w_a \cdot \phi(\vec{y}_{N(a)})\right)$$

  where ϕ is a feature function
- Usually this will result in way too many parameters to train
- Therefore we tie the models at different factors by sharing parameters
- Represented easily by one level of indirection
- Let $\tau(a)$ map from factor a to a template
- Index w not with a but with $\tau(a)$

$$\Psi_a(\vec{y}_{N(a)}) = \exp\left(w_{\tau(a)} \cdot \phi(\vec{y}_{N(a)})\right)$$

- E.g., in the skip-chain graph,
  - all factors connecting y_{t-1}, x_{t-1}, y_t share one set of parameters

Parameterizing Ψ in factors (2)

- all factors at the centers of stars like "Green" share a second set of parameters
- A more common way to write this is using clique templates
- Partition the factor nodes into clusters (also called cliques) $\{C\}$, and write

$$\Pr(\vec{y}) = \frac{1}{Z} \prod_{C} \prod_{a \in C} \Psi_a(\vec{y}_{N(a)}) = \frac{1}{Z} \prod_{C} \prod_{a \in C} e^{w_C \cdot \phi(\vec{y}_{N(a)})}$$

- Therefore,

$$\log \Pr(\vec{y}) = \sum_{C} \sum_{a \in C} w_C \cdot \phi(\vec{y}_{N(a)}) - \log Z$$

- Similar to logistic regression and chain CRF, except that evaluating Z and expected feature vectors will be via belief propagation

Training

- Note that there is no need to represent labeled examples $\vec{y}^{\ell}$ separately, because they can be disconnected components in a single graph
- The training problem is to find $\{w_C\}$ for all C so as to maximize

$$\sum_C \sum_{a \in C} w_C \cdot \phi(\vec{y}_{N(a)}) - \log Z$$

HW Given a belief propagation subroutine, write down precise pseudocode for computing the objective and gradient for every iteration of a Newton method

Partially observed data

- Thus far we have represented all random variables as Y_u that are unobserved, for simplicity
- Recall we started with both $\vec{x}$ and $\vec{y}$
- In our information extraction applications some variables are observed as x_ts
- $\Pr(\vec{y})$ changes to $\Pr(\vec{y}|\vec{x})$
- Z changes to $Z(\vec{x})$
- $\phi(\vec{y}_{N(a)})$ changes to $\phi(\vec{x}, \vec{y}_{N(a)})$
- Note we can still use any part of $\vec{x}$

Dual decomposition

- Indicator variables $y(t, m', m) = 1$ if positions $t-1, t$ are assigned labels m', m, 0 otherwise
- Sequence label space $\mathcal{Y} \in \{0,1\}^{TM^2}$, size is typically exponential in input size
- (Not all combinations may be used)
- Inference amounts to finding $\arg\max_{y \in \mathcal{Y}} h(y)$

HW For the linear HMM or CRF objective, $h(y)$ has the form $\theta \cdot y$, where θ depends on observations $\vec{x}$

HW Why didn't we simply design $y(t, m) = 1$ if position t is labeled m and 0 otherwise? Why pull in m'?

- Suppose there exists $\mathcal{Y}' \supset \mathcal{Y}$ such that finding $\arg\max_{y \in \mathcal{Y}'} \theta \cdot y$ is 'easy'
- However, the application requires restricting $\mathcal{Y}'$ in other ways, e.g., $\mathcal{Y} = \{y : y \in \mathcal{Y}' \text{ and } Ay = b\}$

Dual decomposition (2)

- E.g., restricting labels of some distant tokens to be equal or related can be handled with such constraints
- Say there are c constraints
- Introduce Lagrangian multipliers $u \in \mathbb{R}^c$ and define

$$L(u, y) = \theta \cdot y + u \cdot (Ay - b)$$

- The dual problem is

$$\min_{u \in \mathbb{R}^c} L(u) = \min_{u \in \mathbb{R}^c} \left\{ \max_{y \in \mathcal{Y}'} L(u, y) \right\}$$

- A 'game' between u and y: y tries to satisfy $Ay = b$, if not, u tries to drag down $L(u, y)$

Dual decomposition (3)

- Common approach: set $u^{(0)} = \vec{0}$, then:

$$\text{set} \qquad y^{(k)} = \operatorname*{argmax}_{y \in \mathcal{Y}'} L(u^{(k-1)}, y)$$

$$\text{followed by} \qquad u^{(k)} = u^{(k-1)} - \delta_k (A y^{(k-1)} - b)$$

  where δ_k is a stepsize

- Note that

$$\begin{aligned} \operatorname*{argmax}_{y \in \mathcal{Y}'} L(u^{(k-1)}, y) &= \operatorname*{argmax}_{y \in \mathcal{Y}'} \left(\theta \cdot y + u^{(k-1)} (A y - b) \right) \\ &= \operatorname*{argmax}_{y \in \mathcal{Y}'} \theta' \cdot y \end{aligned}$$

  which is also 'easy'

- Theoretical guarantees in the paper

Dual decomposition (4)

- An extension of the above is where there are two overlapping label spaces $\mathcal{Y}, \mathcal{Z}$, with

$$f(y) = y \cdot \theta^{(1)} \quad \text{and} \quad g(z) = z \cdot \theta^{(2)}$$

- E.g., joint sequence labeling consistent with parsing
- Or joint coref resolution with entity disambiguation
- The decoding objective is

$$\operatorname*{argmax}_{y \in \mathcal{Y}, z \in \mathcal{Z}} y \cdot \theta^{(1)} + z \cdot \theta^{(2)} \quad \text{s.t.} \quad Ay + Cz = b$$

- Same approach as before: relax $\mathcal{Y}, \mathcal{Z}$ to $\mathcal{Y}', \mathcal{Z}'$ and define

$$\min_u \max_{y,z} L(u, y, z) = \min_u \left\{ \max_{y,z} y \cdot \theta^{(1)} + z \cdot \theta^{(2)} + u(Ay + Cz - b) \right\}$$

Dual decomposition (5)

- Set $u^{(0)} = \vec{0}$, and alternate

$$y^{(k)}, z^{(k)} = \operatorname*{argmax}_{y,z} L(u^{(k-1)}, y, z)$$

with $$u^{(k)} = u^{(k-1)} - \delta_k(Ay^{(k)} + Cz^{(k)} - b)$$

- Again, note that updating $y^{(k)}, z^{(k)}$ amounts to separately optimizing

$$\operatorname*{argmax}_{y \in \mathcal{Y}'} y \cdot \tilde{\theta}^{(1)} \quad \text{and} \quad \operatorname*{argmax}_{z \in \mathcal{Z}'} z \cdot \tilde{\theta}^{(2)}$$

- Unlike in max-product message passing, at the end of above optimization we still need to round the fractional solutions to feasible 0/1 solutions

Dependency parsing

- PCFGs are used for constituency parsing, which reveal the hierarchical phrase/clause structure of the sentence; intermediate levels of the hierarchy are represented by nonterminal nodes
- In contrast, a dependency parse connects pairs of words in a tree structure without introducing new nodes; edges connecting words are labeled with the nature of relationship between them
- Important techniques:
  - Maximum rooted directed arborescence
  - "Deterministic" shift-reduce style
- Apart from standard NLP tasks, used for:
  - Composing continuous representation of sentences
  - Translating questions into structured query plans
  - Features in fine type tagging, relation extraction

http://nlp.stanford.edu:8080/corenlp/
https://stp.lingfil.uu.se/~nivre/docs/ACLslides.pdf

Embeddings and neural techniques

Distributional vectors and word clusters

- Finite labeled data, feature sparsity, and out-of-vocabulary (OOV) words/features have always troubled POS and NER tagging and estimating n-gram statistics
- E.g. we saw car in the training sequences but never sedan, or Shanghai in test data but only other cities in training data
- But an unlabeled corpus has enough clues that these are related words
- A well-established partial fix is word cluster features
- First proposed by IBM researchers in 1992 http://aclweb.org/anthology/J/J92/J92-4003.pdf
- Given a word like sedan, collect all context windows (say) at most 11 words wide centered on it
- From these contexts collect a bag of other words, count them
- Possibly transform from raw counts to TFIDF

Distributional vectors and word clusters (2)

- Represent as a sparse vector in a space as large as the corpus vocabulary; perhaps scale to unit length
- This is the distributional vector for sedan
- Turns out the d.v.'s of similar/related words are similar
- Can cluster these d.v.'s using standard clustering tools; see https://en.wikipedia.org/wiki/Brown_clustering
- In standard CRF implementations, one of the features for each token is its cluster ID
- The best number of clusters may be application-dependent

Word embeddings

- In a token window, the focus token f is at the center and others are context tokens c
- Each word in the vocabulary is associated with two embeddings, $u_w \in \mathbb{R}^D$ as focus and $v_w \in \mathbb{R}^D$ as context
- Typically D ranges from 100 to 1000
- Two dominant paradigms to train $\boldsymbol{U}, \boldsymbol{V}$

GloVe: $$\log X_{fc} \approx u_f \cdot v_c + b_f + b_c,$$

where $b_w \in \mathbb{R}$ is a per-word offset and X_{fc} is the cooccurrence count of words f and c, and

Word2vec: $$\Pr(f, c \text{ cooccur}) = \sigma(u_f \cdot v_c),$$

where $\sigma(\bullet) = 1/(1 + e^{-\bullet})$ is the sigmoid function

Word embeddings (2)

- Variations of low-rank factorization of a transformed cooccurrence matrix
- Usually only $\boldsymbol{U}$ used for downstream tasks, one vector per word, usually scaled to unit L2 norm
- Although not explicitly trained to those ends, the focus embeddings are useful for many tasks
  - $u_{\text{auto}} \approx u_{\text{sedan}}$
  - $u_{\text{king}} - u_{\text{man}} + u_{\text{woman}} \approx u_{\text{queen}}$, etc.
- A common use of word embeddings is to inject them into sequential networks as "x_t"

Using word embeddings in sequence labeling

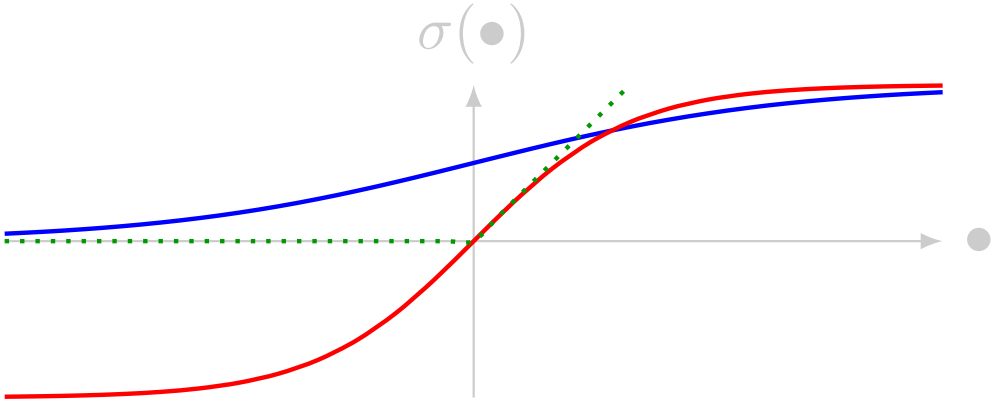
- Concatenate u_{sedan} with all other lexical and derived features of sedan
- If $D = 300$ and we had 20000 lexical features and 50 derived features, now we have $F = 20350$ features
- (May need some scaling of feature groups so embeddings don't crowd out the high-info derived features)
- Note, the embeddings themselves do not adapt to the task (POS/NER/etc.) at hand
- Best current approach:
  - Directly build neural network to translate from sequence of continuous vectors to sequence of continuous vectors/states
  - Learn a translation from continuous to discrete states demanded by application
  - Train through backprop

Standard neural network terminology

- A **hidden layer** inputs a vector $x \in \mathbb{R}^I$, multiplies by a matrix $W \in \mathbb{R}^{I \times O}$, and outputs an output vector $y = xW \in \mathbb{R}^O$
- Often a vector $b \in \mathbb{R}^O$ is added; i.e., $y = xW + b$
- Usually a hidden layer then applies a *nonlinearity*
- **Nonlinearity**, generically denoted $\sigma(\bullet)$, applied elementwise to input scalar, vector, matrix or tensor
  - Sigmoid: $\sigma(\bullet) = 1/(1 + e^{-\bullet})$
  - Rectified linear unit (ReLU): $\sigma(\bullet) = \max\{0, \bullet\}$
  - Tanh: $\sigma(\bullet) = \tanh(\bullet) = \dfrac{e^{\bullet} - e^{-\bullet}}{e^{\bullet} + e^{-\bullet}}$

Standard neural network terminology (2)

When to use each?



- Sometimes a nonlinearity has its own associated model weights to be learnt, e.g.,

$$\sigma_{a,b}(\bullet) = \frac{1}{1 + \exp(-b(\bullet - a))}$$

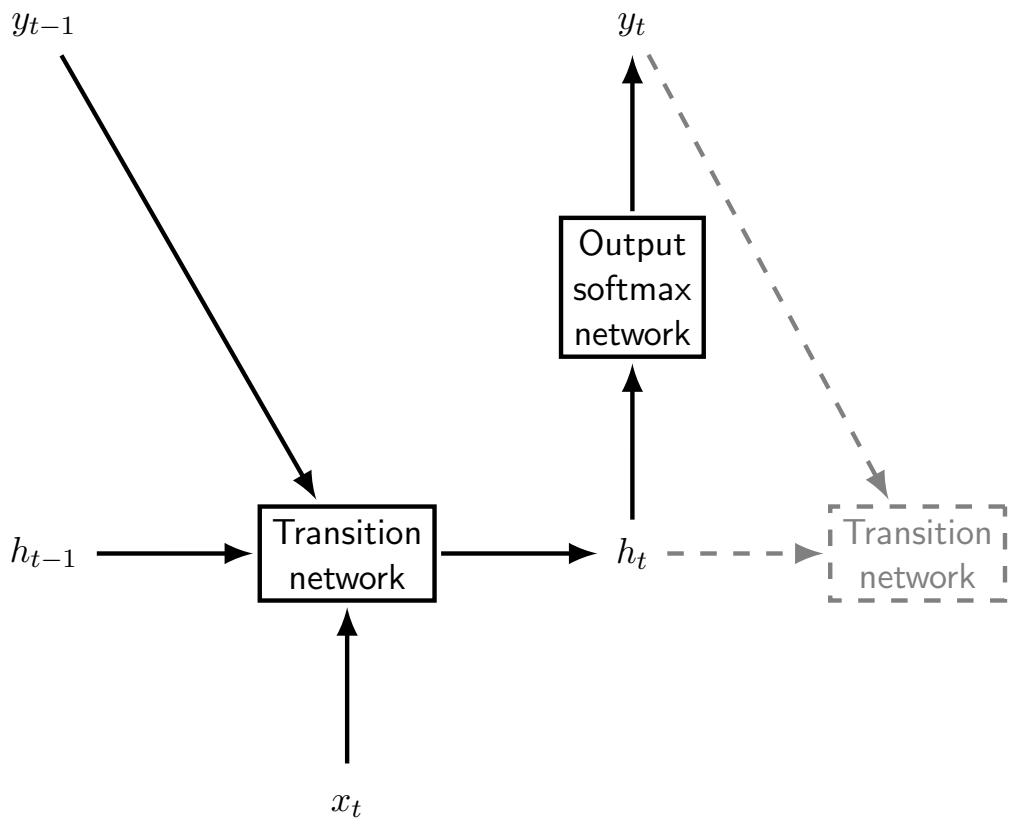
  which changes the offset (a) and slope (b) of saturation
- Or it may be folded into the preceding hidden layer by appending a constant dimension to the input vector
- I.e., the typical hidden layer amounts to $y = \sigma(xW + b)$

Standard neural network terminology (3)

- A **softmax** layer turns K real (positive or negative) numbers $a_1, \ldots, a_K$ into a K-way multinomial distribution as $b_k = \exp(a_k) / \sum_{k'} \exp(a_{k'})$
- Typically used for transforming the output of a hidden layer into a multinomial distribution over labels
- Elementwise product of tensors $x \circ y$
- Tensorflow hello world: `regression.py`, `svm.py`

Basic network template

- Discrete state y_t replaced with continuous vector states h_t; usually $h_0 = \vec{0}$
- The transition network inputs previous state h_{t-1} and current input x_t (also continuous form) and outputs current state h_t
- y_t for end application derived from h_t using an output network ending in a softmax



Basic recurrent network (RNN)

- The transition network and the output network are standard multi-layer neural networks with linear combinations and nonlinearities at every hidden layer

$$\begin{aligned}
&\text{Elman network:} && h_t = \sigma(x_t W_h + h_{t-1} U_h + b_h) \\
& && y_t = \sigma(h_t W_y + b_y) \\
&\text{Jordan network:} && h_t = \sigma(x_t W_h + y_{t-1} U_h + b_h) \\
& && y_t = \sigma(h_t W_y + b_y)
\end{aligned}$$

- Weights W_h, U_h, b_h of transition networks at all positions are tied; same for weights W_y, b_y in the output network
- Potential problems: vanishing or exploding gradient

Gated recurrent unit (GRU)

$$
\begin{aligned}
\text{Reset} \quad & r_t = \sigma(x_t W_r + h_{t-1} U_r + b_r) \\
\text{Combine} \quad & c_t = \tanh(x_t W_h + (r_t \circ h_{t-1}) U_h + b_h) \\
\text{Update} \quad & z_t = \sigma(x_t W_z + h_{t-1} U_z + b_z) \\
\text{Output} \quad & h_t = (1 - z_t) \circ h_{t-1} + z_t \circ c_t
\end{aligned}
$$

- σ is sigmoid; other nonlinearities specified explicitly
- r_t decides if c_t is computed as in a basic RNN, or h_{t-1} is discarded and only x_t is used
- z_t decides how to blend c_t and h_{t-1} as-is
- If we set by force $r_t \approx 1$ and $z_t \approx 1$, equivalent to basic RNN
- https://arxiv.org/abs/1412.3555

Long short-term memory (LSTM)

In	$i_t = \sigma(x_t U_i + h_{t-1} W_i + b_i)$
Forget	$f_t = \sigma(x_t U_f + h_{t-1} W_f + b_f)$
Out	$o_t = \sigma(x_t U_o + h_{t-1} W_o + b_o)$
RNN	$g_t = \tanh(x_t U_g + h_{t-1} W_g + b_g)$
Memory	$c_t = c_{t-1} \circ f_t + g_t \circ i_t$
State	$h_t = \tanh(c_t) \circ o_t$

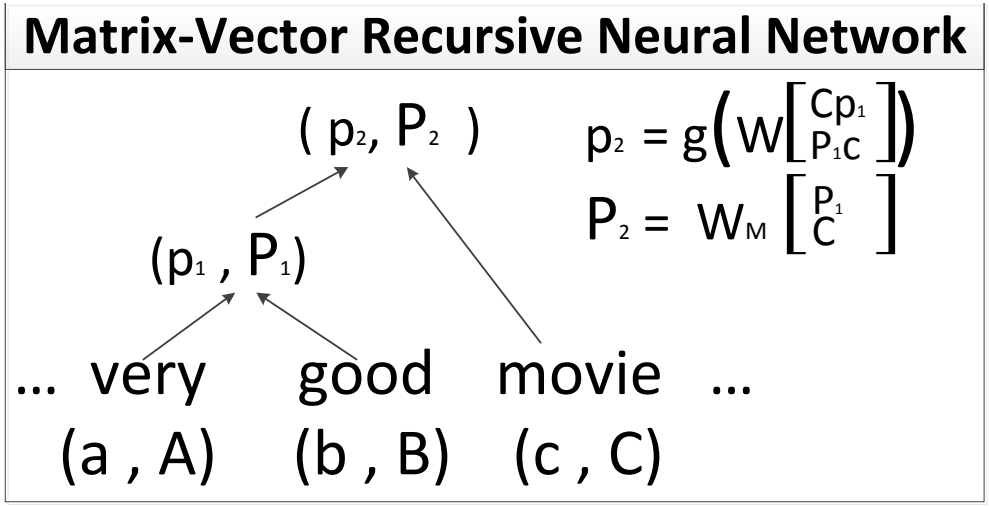
- Understanding LSTMs
- Accuracy comparable to GRUs
- Widely used for sequence labeling, sequence-to-sequence comparison, sequence-to-sequence translation, etc.
- Usually left-to-right and right-to-left LSTMs in tandem as a bi-LSTM; captures context from both sides
- Unreasonable effectiveness

Composition along parse trees

- If we have a reliable dependency or constituency parse, makes sense to use it for composing word vectors
- Words a, b, with vectors $w_a, w_b \in \mathbb{R}^D$, are children of internal node c
- First-cut: Fit $w_c = \sigma\left(M \begin{bmatrix} w_a \\ w_b \end{bmatrix}\right)$ where $M \in \mathbb{R}^{D \times 2D}$
- Single M unlikely to be able to capture all natural language 'operators'
- Associate each word a with vector $w_a \in \mathbb{R}^D$ and matrix $M_a \in \mathbb{R}^{D \times 2D}$
- Let $w_c = \sigma\left(M_0 \begin{bmatrix} M_b w_a \\ M_a w_b \end{bmatrix}\right)$ where $M_0 \in \mathbb{R}^{D \times 2D}$ is a global matrix

Composition along parse trees (2)

- Each word operates on the other, results combined



- For compositionality, must define M_c as well: $M_c = M_1 \begin{bmatrix} M_a \\ M_b \end{bmatrix}$

  where $M_1 \in \mathbb{R}^{D \times 2D}$ is another global matrix
- Overall loss depends on task at hand
  - w_{root} can be mapped to $\text{softmax}(M_{\text{root}} w_{\text{root}})$ for classification tasks
  - IMDB movie reviews: unbelievably sad, really awesome, pretty bad, not terrible

Composition along parse trees (3)

Method	Avg KL
$\frac{1}{2}(w_a + w_b)$	0.103
$w_a \circ w_b$	0.103
$M_a w_b$	0.103
MV-RNN	**0.091**

- Classifying semantic relationships
  - "My [apartment]$_{e_1}$ has a pretty large [kitchen]$_{e_2}$" $\longrightarrow$ component-whole

Relationship	Sentence with labeled nouns for which to predict relationships
Cause-Effect(e2,e1)	Avian [influenza]$_{e1}$ is an infectious disease caused by type a strains of the influenza [virus]$_{e2}$.
Entity-Origin(e1,e2)	The [mother]$_{e1}$ left her native [land]$_{e2}$ about the same time and they were married in that city.
Message-Topic(e2,e1)	Roadside [attractions]$_{e1}$ are frequently advertised with [billboards]$_{e2}$ to attract tourists.
Product-Producer(e1,e2)	A child is told a [lie]$_{e1}$ for several years by their [parents]$_{e2}$ before he/she realizes that ...
Entity-Destination(e1,e2)	The accident has spread [oil]$_{e1}$ into the [ocean]$_{e2}$.
Member-Collection(e2,e1)	The siege started, with a [regiment]$_{e1}$ of lightly armored [swordsmen]$_{e2}$ ramming down the gate.
Instrument-Agency(e2,e1)	The core of the [analyzer]$_{e1}$ identifies the paths using the constraint propagation [method]$_{e2}$.
Component-Whole(e2,e1)	The size of a [tree]$_{e1}$ [crown]$_{e2}$ is strongly correlated with the growth of the tree.
Content-Container(e1,e2)	The hidden [camera]$_{e1}$, found by a security guard, was hidden in a business card-sized [leaflet box]$_{e2}$ placed at an unmanned ATM in Tokyo's Minato ward in early September.

Composition along parse trees (4)

- Performance on SemEval-2010, task 8

Classifier	Feature Sets	F1
SVM	POS, stemming, syntactic patterns	60.1
SVM	word pair, words in between	72.5
SVM	POS, WordNet, stemming, syntactic patterns	74.8
SVM	POS, WordNet, morphological features, thesauri, Google n-grams	77.6
MaxEnt	POS, WordNet, morphological features, noun compound system, thesauri, Google n-grams	77.6
SVM	POS, WordNet, prefixes and other morphological features, POS, dependency parse features, Levin classes, PropBank, FrameNet, NomLex-Plus, Google n-grams, paraphrases, TextRunner	82.2
RNN	-	74.8
Lin.MVR	-	73.0
MV-RNN	-	79.1
RNN	POS,WordNet,NER	77.6
Lin.MVR	POS,WordNet,NER	78.7
MV-RNN	POS,WordNet,NER	**82.4**

Table 4: Learning methods, their feature sets and F1 results for predicting semantic relations between nouns. The MV-RNN outperforms all but one method without any additional feature sets. By adding three such features, it obtains state of the art performance.

Convolutional networks over text

- Composing continuous representation without parses
- Understanding convnets for NLP

References

[1] D. Freitag and A. McCallum, "Information extraction using HMMs and shrinkage," in *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999, pp. 31–36.

[2] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *JMLR*, vol. 6, no. Sep., pp. 1453–1484, 2005. [Online]. Available: http://ttic.uchicago.edu/~altun/pubs/TsoJoaHofAlt-JMLR.pdf

[3] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[4] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *HLT-NAACL*, 2003, pp. 134–141. [Online]. Available: http://acl.ldc.upenn.edu/N/N03/N03-1028.pdf

[5] X. Ling and D. S. Weld, "Fine-grained entity recognition." in *AAAI*, 2012. [Online]. Available: http://xiaoling.github.io/pubs/ling-aaai12.pdf

References (2)

[6] D. Gillick, N. Lazic, K. Ganchev, J. Kirchner, and D. Huynh, "Context-dependent fine-grained entity type tagging," *arXiv preprint arXiv:1412.1820*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.1820.pdf

[7] D. Yogatama, D. Gillick, and N. Lazic, "Embedding methods for fine grained entity type classification," in *ACL Conference*, 2015, pp. 26–31. [Online]. Available: http://anthology.aclweb.org/P/P15/P15-2048.pdf

[8] S. Shimaoka, P. Stenetorp, K. Inui, and S. Riedel, "An attentive neural architecture for fine-grained entity type classification," *arXiv preprint arXiv:1604.05525*, 2016. [Online]. Available: https://arxiv.org/pdf/1604.05525.pdf

[9] Y. Yaghoobzadeh, H. Adel, and H. Schütze, "Noise mitigation for neural entity typing and relation extraction," *arXiv preprint arXiv:1612.07495*, 2016. [Online]. Available: https://arxiv.org/pdf/1612.07495.pdf

References (3)

[10] S. Dill *et al.*, "SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation," in *WWW Conference*, 2003, pp. 178–186.

[11] R. Mihalcea and A. Csomai, "Wikify!: linking documents to encyclopedic knowledge," in *CIKM*, 2007, pp. 233–242. [Online]. Available: http://portal.acm.org/citation.cfm?id=1321440.1321475

[12] R. Bunescu and M. Pasca, "Using encyclopedic knowledge for named entity disambiguation," in *EACL*, 2006, pp. 9–16. [Online]. Available: http://www.cs.utexas.edu/~ml/papers/encyc-eacl-06.pdf

[13] S. Cucerzan, "Large-scale named entity disambiguation based on Wikipedia data," in *EMNLP Conference*, 2007, pp. 708–716. [Online]. Available: http://www.aclweb.org/anthology/D/D07/D07-1074

[14] J. Hoffart *et al.*, "Robust disambiguation of named entities in text," in *EMNLP Conference*. Edinburgh, Scotland, UK: SIGDAT, Jul. 2011, pp. 782–792. [Online]. Available: http://aclweb.org/anthology/D/D11/D11-1072.pdf

References (4)

[15] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti, "Collective annotation of Wikipedia entities in Web text," in *SIGKDD Conference*, 2009, pp. 457–466. [Online]. Available: http://www.cse.iitb.ac.in/~soumen/doc/CSAW/

[16] A. Globerson, N. Lazic, S. Chakrabarti, A. Subramanya, M. Ringgaard, and F. Pereira, "Collective entity resolution with multi-focal attention," in *ACL Conference*, 2016, pp. 621–631. [Online]. Available: https://www.aclweb.org/anthology/P/P16/P16-1059.pdf

[17] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," *arXiv preprint arXiv:1601.01343*, 2016. [Online]. Available: https://arxiv.org/pdf/1601.01343.pdf

[18] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," *arXiv preprint arXiv:1704.04920*, 2017. [Online]. Available: https://arxiv.org/pdf/1704.04920.pdf

References (5)

[19] N. Lazic, A. Subramanya, M. Ringgaard, and F. Pereira, "Plato: A selective context model for entity resolution," *TACL*, vol. 3, pp. 503–515, 2015. [Online]. Available: http://anthology.aclweb.org/Q/Q15/Q15-1036.pdf

[20] N. Ge, J. Hale, and E. Charniak, "A statistical approach to anaphora resolution," in *Proceedings of the sixth workshop on very large corpora*, vol. 71, 1998, p. 76. [Online]. Available: http://www.aclweb.org/anthology/W98-1119

[21] M. Charikar, V. Guruswami, and A. Wirth, "Clustering with qualitative information," in *FOCS Conference*, 2003, pp. 524–533. [Online]. Available: http://www.cs.mu.oz.au/~awirth/pubs/awirthFocs03.pdf

[22] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning," in *SIGKDD Conference*, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 269–278. [Online]. Available: http://www.cse.iitb.ac.in/~sunita/papers/kdd02.pdf

References (6)

[23] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," in *FOCS Conference*, 2002, p. 238. [Online]. Available: http://www.cs.cmu.edu/~shuchi/papers/clusteringfull.pdf

[24] A. McCallum and B. Wellner, "Conditional models of identity uncertainty with application to noun coreference," in *NIPS Conference*, 2004, pp. 905–912. [Online]. Available: https://papers.nips.cc/paper/2557-conditional-models-of-identity-uncertainty-with-application-to-noun-coreference.pdf

[25] P. Singla and P. Domingos, "Object identification with attribute-mediated dependences," in *PKDD Conference*, Porto, Portugal, 2005, pp. 297–308. [Online]. Available: http://www.cs.washington.edu/homes/parag/papers/object-mediated-pkdd05.pdf

[26] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE PAMI*, vol. 26, no. 2, pp. 147–159, Feb. 2004. [Online]. Available: http://www.cs.cornell.edu/rdz/Papers/KZ-ECCV02-graphcuts.pdf

References (7)

[27] D. M. Greig, B. T. Porteous, and A. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society*, vol. B, no. 51, pp. 271–279, 1989. [Online]. Available: http://jstor.org/stable/2345609

[28] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *International Conference on Computational Linguistics*, vol. 14, 1992, pp. 539–545. [Online]. Available: http://www.aclweb.org/website/old_anthology/C/C92/C92-2082.pdf

[29] O. Etzioni, M. Cafarella *et al.*, "Web-scale information extraction in KnowItAll," in *WWW Conference*. New York: ACM, 2004. [Online]. Available: http://www.cs.washington.edu/research/knowitall/papers/www-paper.pdf

[30] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the Web," in *IJCAI*, M. M. Veloso, Ed., 2007, pp. 2670–2676. [Online]. Available: http://www.ijcai.org/papers07/Papers/IJCAI07-429.pdf

References (8)

[31] H. Poon and P. Domingos, "Unsupervised semantic parsing," in *EMNLP Conference*, 2009, pp. 1–10. [Online]. Available: http://anthology.aclweb.org/D/D09/D09-1001.pdf

[32] L. Yao, A. Haghighi, S. Riedel, and A. McCallum, "Structured relation discovery using generative models," in *EMNLP Conference*, 2011, pp. 1456–1466. [Online]. Available: http://anthology.aclweb.org/D/D11/D11-1135.pdf

[33] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "Relation extraction with matrix factorization and universal schemas," in *NAACL Conference*, 2013, pp. 74–84. [Online]. Available: http://www.anthology.aclweb.org/N/N13/N13-1008.pdf

[34] S. Brin, "Extracting patterns and relations from the World Wide Web," in *WebDB Workshop*, ser. LNCS, P. Atzeni, A. O. Mendelzon, and G. Mecca, Eds., vol. 1590. Valencia, Spain: Springer, Mar. 1998, pp. 172–183. [Online]. Available: http://ilpubs.stanford.edu:8090/421/1/1999-65.pdf

References (9)

[35] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *ICDL*, 2000, pp. 85–94. [Online]. Available: http://www.academia.edu/download/31007490/cucs-033-99.pdf

[36] R. C. Bunescu and R. J. Mooney, "A shortest path dependency kernel for relation extraction," in *EMNLP Conference*. ACL, 2005, pp. 724–731. [Online]. Available: http://acl.ldc.upenn.edu/H/H05/H05-1091.pdf

[37] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning, "Multi-instance multi-label learning for relation extraction," in *EMNLP Conference*, 2012, pp. 455–465. [Online]. Available: http://anthology.aclweb.org/D/D12/D12-1042.pdf

[38] G. Angeli, J. Tibshirani, J. Wu, and C. D. Manning, "Combining distant and partial supervision for relation extraction." in *EMNLP Conference*, 2014, pp. 1556–1567. [Online]. Available: http://www.anthology.aclweb.org/D/D14/D14-1164.pdf

References (10)

[39] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *ACL Conference*, 2011, pp. 541–550. [Online]. Available: http://anthology.aclweb.org/P/P11/P11-1055.pdf

[40] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek, "Distant supervision for relation extraction with an incomplete knowledge base." in *NAACL Conference*, 2013, pp. 777–782. [Online]. Available: http://www.anthology.aclweb.org/N/N13/N13-1095.pdf

[41] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *AAAI Conference*, 2011, pp. 301–306. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/viewFile/3659/3898

[42] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS Conference*, 2013, pp. 2787–2795. [Online]. Available: http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf

References (11)

[43] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix." in *ACL Conference*, 2015, pp. 687–696. [Online]. Available: http://www.aclweb.org/anthology/P/P15/P15-1067.pdf

[44] I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun, "Order-embeddings of images and language," *arXiv preprint arXiv:1511.06361*, 2015. [Online]. Available: https://arxiv.org/pdf/1511.06361

[45] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *EMNLP Conference*, 2015, pp. 1499–1509. [Online]. Available: https://www.aclweb.org/anthology/D/D15/D15-1174.pdf

[46] P. D. Turney, "Mining the Web for synonyms: PMI-IR versus LSA on TOEFL," in *ECML*, 2001.

References (12)

[47] J. Zhu, Z. Nie, B. Zhang, and J.-R. Wen, "Dynamic hierarchical Markov random fields and their application to Web data extraction," in *ICML*, 2007, pp. 1175–1182. [Online]. Available: http://www.machinelearning.org/proceedings/icml2007/papers/215.pdf

[48] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in *ICDE*. IEEE, 2002.

[49] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A system for keyword-based search over relational databases," in *ICDE*. San Jose, CA: IEEE, 2002.

[50] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-style keyword search over relational databases," in *VLDB Conference*, 2003, pp. 850–861. [Online]. Available: http://www.db.ucsd.edu/publications/VLDB2003cr.pdf

[51] G. Jeh and J. Widom, "Scaling personalized web search," in *WWW Conference*, 2003, pp. 271–279. [Online]. Available: http://www2003.org/cdrom/papers/refereed/p185/html/p185-jeh.html

References (13)

[52] T. H. Haveliwala, "Topic-sensitive PageRank," in *WWW Conference*, 2002, pp. 517–526. [Online]. Available: http://www2002.org/CDROM/refereed/127/index.html

[53] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Authority-based keyword queries in databases using ObjectRank," in *VLDB Conference*, Toronto, 2004.

[54] M. J. Cafarella, C. Re, D. Suciu, O. Etzioni, and M. Banko, "Structured querying of web text: A technical challenge," in *CIDR*, 2007, pp. 225–234. [Online]. Available: http://www-db.cs.wisc.edu/cidr/cidr2007/papers/cidr07p25.pdf

[55] S. Chakrabarti, K. Puniyani, and S. Das, "Optimizing scoring functions and indexes for proximity search in type-annotated corpora," in *WWW Conference*, Edinburgh, May 2006, pp. 717–726. [Online]. Available: http://www.cse.iitb.ac.in/~soumen/doc/www2006i

References (14)

[56] T. Cheng, X. Yan, and K. C.-C. Chang, "EntityRank: Searching entities directly and holistically," in *VLDB Conference*, Sep. 2007, pp. 387–398. [Online]. Available: http://www-forward.cs.uiuc.edu/pubs/2007/entityrank-vldb07-cyc-jul07.pdf

[57] S. Chakrabarti, "Dynamic personalized PageRank in entity-relation graphs," in *WWW Conference*, Banff, May 2007. [Online]. Available: http://www.cse.iitb.ac.in/~soumen/doc/netrank/

[58] P. Sarkar, A. W. Moore, and A. Prakash, "Fast incremental proximity search in large graphs," in *ICML*, 2008, pp. 896–903. [Online]. Available: http://icml2008.cs.helsinki.fi/papers/565.pdf

[59] D. Milne and I. H. Witten, "Learning to link with Wikipedia," in *CIKM*, 2008, pp. 509–518. [Online]. Available: http://www.cs.waikato.ac.nz/~dnk2/publications/CIKM08-LearningToLinkWithWikipedia.pdf

References (15)

[60] G. Kasneci, F. M. Suchanek, G. Ifrim, S. Elbassuoni, M. Ramanath, and G. Weikum, "NAGA: harvesting, searching and ranking knowledge," in *SIGMOD Conference*. ACM, 2008, pp. 1285–1288. [Online]. Available: http://www.mpi-inf.mpg.de/~kasneci/naga/

[61] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A core of semantic knowledge unifying WordNet and Wikipedia," in *WWW Conference*. ACM Press, 2007, pp. 697–706. [Online]. Available: http://www2007.org/papers/paper391.pdf

[62] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS Conference*, 2013, pp. 3111–3119. [Online]. Available: https://goo.gl/x3DTzS

[63] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation." in *EMNLP Conference*, vol. 14, 2014, pp. 1532–1543. [Online]. Available: http://www.emnlp2014.org/papers/pdf/EMNLP2014162.pdf

References (16)

[64] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Machine Learning*, vol. 81, no. 1, pp. 53–67, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10994-010-5205-8

[65] S. Sarawagi, "Information extraction," *FnT Databases*, vol. 1, no. 3, 2008. [Online]. Available: http://www.cse.iitb.ac.in/~sunita/papers/ieSurvey.pdf