6.864: Lecture 21 (November 29th, 2005)

Global Linear Models: Part II

Overview

- Log-linear models for parameter estimation

- Global and local features

  - The perceptron revisited
  - Log-linear models revisited

Three Components of Global Linear Models

- $\mathbf{\Phi}$ is a function that maps a structure (x, y) to a **feature vector** $\mathbf{\Phi}(x, y) \in \mathbb{R}^d$

- **GEN** is a function that maps an input x to a set of **candidates** $\mathbf{GEN}(x)$

- $\mathbf{W}$ is a parameter vector (also a member of $\mathbb{R}^d$)

- Training data is used to set the value of $\mathbf{W}$

Putting it all Together

- $\mathcal{X}$ is set of sentences, $\mathcal{Y}$ is set of possible outputs (e.g. trees)

- Need to learn a function $F : \mathcal{X} \to \mathcal{Y}$
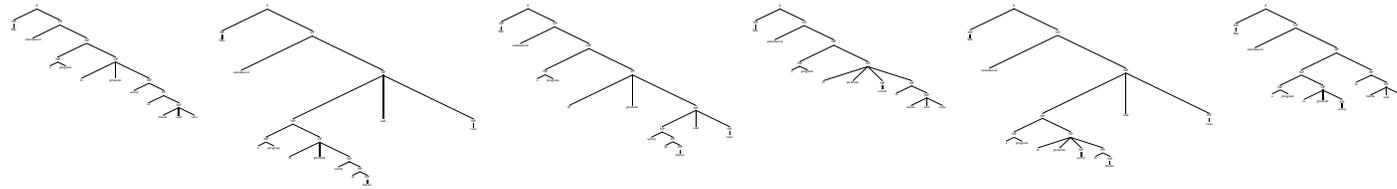
- **GEN**, $\mathbf{\Phi}$, **W** define

$$F(x) = \arg\max_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$$

Choose the highest scoring candidate as the most plausible structure

- Given examples (x_i, y_i), how to set $\mathbf{W}$?

She announced a program to promote safety in trucks and vans

$\Downarrow$ **GEN**

$\Downarrow \mathbf{\Phi}$ $\Downarrow \mathbf{\Phi}$ $\Downarrow \mathbf{\Phi}$ $\Downarrow \mathbf{\Phi}$ $\Downarrow \mathbf{\Phi}$ $\Downarrow \mathbf{\Phi}$

$\langle 1, 1, 3, 5 \rangle$ $\langle 2, 0, 0, 5 \rangle$ $\langle 1, 0, 1, 5 \rangle$ $\langle 0, 0, 3, 0 \rangle$ $\langle 0, 1, 0, 5 \rangle$ $\langle 0, 0, 1, 5 \rangle$

$\Downarrow \mathbf{\Phi} \cdot \mathbf{W}$ $\Downarrow \mathbf{\Phi} \cdot \mathbf{W}$ $\Downarrow \mathbf{\Phi} \cdot \mathbf{W}$ $\Downarrow \mathbf{\Phi} \cdot \mathbf{W}$ $\Downarrow \mathbf{\Phi} \cdot \mathbf{W}$ $\Downarrow \mathbf{\Phi} \cdot \mathbf{W}$

13.6 12.2 12.1 3.3 9.4 11.1

$\Downarrow \arg\max$
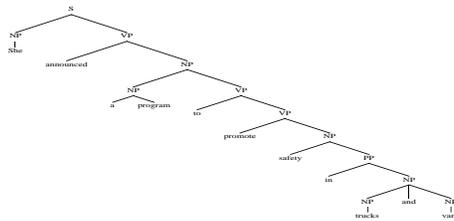
A Variant of the Perceptron Algorithm

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \mathrm{argmax}_{y \in \mathbf{GEN}(x)} \ \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T, \ i = 1 \ldots n$
$\quad z_i = F(x_i)$
$\quad$ If $(z_i \neq y_i) \quad \mathbf{W} = \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

Output: Parameters $\mathbf{W}$

Overview

- Recap: global linear models

- Log-linear models for parameter estimation

- Global and local features

  - The perceptron revisited
  - Log-linear models revisited

Back to Maximum Likelihood Estimation
[Johnson et. al 1999]

- We can use the parameters to define a probability for each parse:

$$P(y \mid x, \mathbf{W}) = \frac{e^{\mathbf{\Phi}(x,y) \cdot \mathbf{W}}}{\sum_{y' \in \mathbf{GEN}(x)} e^{\mathbf{\Phi}(x,y') \cdot \mathbf{W}}}$$

- Log-likelihood is then

$$L(\mathbf{W}) = \sum_i \log P(y_i \mid x_i, \mathbf{W})$$

- A first estimation method: take maximum likelihood estimates, i.e.,

$$\mathbf{W}_{ML} = \operatorname{argmax}_{\mathbf{W}} L(\mathbf{W})$$

Adding Gaussian Priors
[Johnson et. al 1999]

- A first estimation method: take maximum likelihood estimates, i.e., $\mathbf{W}_{ML} = \mathrm{argmax}_{\mathbf{W}} L(\mathbf{W})$

- Unfortunately, very likely to "overfit":
  could use feature selection methods, as in boosting

- Another way of preventing overfitting: choose parameters as

$$\mathbf{W}_{MAP} = \mathrm{argmax}_{\mathbf{W}} \left(L(\mathbf{W}) - C \sum_k \mathbf{W}_k^2 \right)$$

  for some constant C

- Intuition: adds a penalty for large parameter values

The Bayesian Justification for Gaussian Priors

- In *Bayesian* methods, combine the log-likelihood $P(data \mid \mathbf{W})$ with a prior over parameters, $P(\mathbf{W})$

$$P(\mathbf{W} \mid data) = \frac{P(data \mid \mathbf{W})P(\mathbf{W})}{\int_{\mathbf{W}} P(data \mid \mathbf{W})P(\mathbf{W})d\mathbf{W}}$$

- The **MAP** (Maximum A-Posteriori) estimates are

$$\begin{aligned}
\mathbf{W}_{MAP} &= \text{argmax}_{\mathbf{W}} P(\mathbf{W} \mid data) \\
&= \text{argmax}_{\mathbf{W}} \left(\underbrace{\log P(data \mid \mathbf{W})}_{\text{Log-Likelihood}} + \underbrace{\log P(\mathbf{W})}_{\text{Prior}} \right)
\end{aligned}$$

- Gaussian prior: $P(\mathbf{W}) \propto e^{-C \sum_k \mathbf{W}_k^2}$
  $\Rightarrow \log P(\mathbf{W}) = -C \sum_k \mathbf{W}_k^2 + C_2$

Summary

Choose parameters as:

$$\mathbf{W}_{MAP} = \text{argmax}_{\mathbf{W}} \left(L(\mathbf{W}) - C \sum_k \mathbf{W}_k^2 \right)$$

where

$$
\begin{aligned}
L(\mathbf{W}) &= \sum_i \log P(y_i \mid x_i, \mathbf{W}) \\
&= \sum_i \log \frac{e^{\mathbf{\Phi}(x_i, y_i) \cdot \mathbf{W}}}{\sum_{y' \in \mathbf{GEN}(x_i)} e^{\mathbf{\Phi}(x_i, y') \cdot \mathbf{W}}}
\end{aligned}
$$

Can use (conjugate) gradient ascent
(see previous lectures on log-linear models)

Overview

- Recap: global linear models

- Log-linear models for parameter estimation

- Global and local features

  - The perceptron revisited
  - Log-linear models revisited

Global and Local Features

- So far: algorithms have depended on size of **GEN**

- Strategies for keeping the size of **GEN** manageable:

  - Reranking methods: use a baseline model to generate its top N analyses

Global and Local Features

- Global linear models are "global" in a couple of ways:

  - Feature vectors are defined over entire structures
  - Parameter estimation methods explicitly related to errors on entire structures

- Next topic: **global** training methods with **local features**

  - Our "global" features will be defined through *local* features
  - Parameter estimates will be global
  - **GEN** will be large!
  - Dynamic programming used for search and parameter estimation: **this is possible for some combinations of GEN and Φ**

Tagging Problems

TAGGING: Strings to Tagged Sequences

$$a\ b\ e\ e\ a\ f\ h\ j \Rightarrow a/C\ b/D\ e/C\ e/C\ a/D\ f/C\ h/D\ j/C$$

Example 1: Part-of-speech tagging

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

Example 2: Named Entity Recognition

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

Tagging

Going back to tagging:

- Inputs x are sentences $w_{[1:n]} = \{w_1 \ldots w_n\}$

- **GEN**$(w_{[1:n]}) = \mathcal{T}^n$ i.e. all tag sequences of length n

- Note: **GEN** has an exponential number of members

- How do we define $\mathbf{\Phi}$?

Representation: Histories

- A **history** is a 4-tuple $\langle t_{-1}, t_{-2}, w_{[1:n]}, i \rangle$

- t_{-1}, t_{-2} are the previous two tags.

- $w_{[1:n]}$ are the n words in the input sentence.

- i is the index of the word being tagged

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- $t_{-1}, t_{-2} = \text{DT, JJ}$

- $w_{[1:n]} = \langle Hispaniola, quickly, became, \ldots, Hemisphere, . \rangle$

- $i = 6$

Local Feature-Vector Representations

- Take a history/tag pair (h, t).

- $\phi_s(h, t)$ for $s = 1 \ldots d$ are **local features** representing tagging decision t in context h.

Example: POS Tagging

- **Word/tag features**

$$\phi_{100}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } t = \texttt{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{101}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

- **Contextual Features**

$$\phi_{103}(h, t) = \begin{cases} 1 & \text{if } \langle t_{-2}, t_{-1}, t \rangle = \langle \text{DT, JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

A tagged sentence with n words has n history/tag pairs

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/NN

History				Tag
t_{-2}	t_{-1}	$w_{[1:n]}$	i	t
*	*	$\langle Hispaniola, quickly, \ldots, \rangle$	1	NNP
*	NNP	$\langle Hispaniola, quickly, \ldots, \rangle$	2	RB
NNP	RB	$\langle Hispaniola, quickly, \ldots, \rangle$	3	VB
RB	VB	$\langle Hispaniola, quickly, \ldots, \rangle$	4	DT
VP	DT	$\langle Hispaniola, quickly, \ldots, \rangle$	5	JJ
DT	JJ	$\langle Hispaniola, quickly, \ldots, \rangle$	6	NN

A tagged sentence with n words has n history/tag pairs

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/NN

History				Tag
t_{-2}	t_{-1}	$w_{[1:n]}$	i	t
*	*	$\langle Hispaniola, quickly, \dots, \rangle$	1	NNP
*	NNP	$\langle Hispaniola, quickly, \dots, \rangle$	2	RB
NNP	RB	$\langle Hispaniola, quickly, \dots, \rangle$	3	VB
RB	VB	$\langle Hispaniola, quickly, \dots, \rangle$	4	DT
VP	DT	$\langle Hispaniola, quickly, \dots, \rangle$	5	JJ
DT	JJ	$\langle Hispaniola, quickly, \dots, \rangle$	6	NN

Define global features through local features:

$$\mathbf{\Phi}(t_{[1:n]}, w_{[1:n]}) = \sum_{i=1}^{n} \phi(h_i, t_i)$$

where t_i is the i'th tag, h_i is the i'th history

Global and Local Features

- Typically, local features are indicator functions, e.g.,

$$\phi_{101}(h, t) \;=\; \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

- and global features are then counts,

$\boldsymbol{\Phi}_{101}(w_{[1:n]}, t_{[1:n]}) =$ Number of times a word ending in $\texttt{ing}$ is tagged as $\texttt{VBG}$ in $(w_{[1:n]}, t_{[1:n]})$

Putting it all Together

- **GEN**$(w_{[1:n]})$ is the set of all tagged sequences of length n

- **GEN**, $\mathbf{\Phi}$, **W** define

$$
\begin{aligned}
F(w_{[1:n]}) \;&=\; \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \mathbf{W} \cdot \mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) \\[2mm]
&=\; \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \mathbf{W} \cdot \sum_{i=1}^{n} \phi(h_i, t_i) \\[2mm]
&=\; \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \sum_{i=1}^{n} \mathbf{W} \cdot \phi(h_i, t_i)
\end{aligned}
$$

- Some notes:

  - Score for a tagged sequence is a sum of local scores
  - **Dynamic programming can be used to find the** `argmax`**!**
    (because history only considers the previous two tags)

A Variant of the Perceptron Algorithm

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \mathrm{argmax}_{y \in \mathbf{GEN}(x)} \, \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$
$z_i = F(x_i)$
If $(z_i \neq y_i)$ $\mathbf{W} = \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

Output: Parameters $\mathbf{W}$

Training a Tagger Using the Perceptron Algorithm

Inputs: Training set $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \ldots n$.

Initialization: $\mathbf{W} = 0$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$$z_{[1:n_i]} = \arg \max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \mathbf{W} \cdot \mathbf{\Phi}(w^i_{[1:n_i]}, u_{[1:n_i]})$$

$z_{[1:n_i]}$ can be computed with the dynamic programming (Viterbi) algorithm

If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then

$$\mathbf{W} = \mathbf{W} \quad + \quad \mathbf{\Phi}(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \mathbf{\Phi}(w^i_{[1:n_i]}, z_{[1:n_i]})$$

Output: Parameter vector $\mathbf{W}$.

An Example

Say the correct tags for i'th sentence are

<p style="text-align:center">the/DT man/NN bit/VBD the/DT dog/NN</p>

Under current parameters, output is

<p style="text-align:center">the/DT man/NN bit/NN the/DT dog/NN</p>

Assume also that features track: (1) all bigrams; (2) word/tag pairs

Parameters incremented:

$$\langle \text{NN, VBD} \rangle, \langle \text{VBD, DT} \rangle, \langle \text{VBD} \rightarrow \text{bit} \rangle$$

Parameters decremented:

$$\langle \text{NN, NN} \rangle, \langle \text{NN, DT} \rangle, \langle \text{NN} \rightarrow \text{bit} \rangle$$

Experiments

- Wall Street Journal part-of-speech tagging data

  Perceptron = 2.89%, Max-ent = 3.28%
  (11.9% relative error reduction)

- [Ramshaw and Marcus, 1995] NP chunking data

  Perceptron = 93.63%, Max-ent = 93.29%
  (5.1% relative error reduction)

How Does this Differ from Log-Linear Taggers?

- Log-linear taggers (in an earlier lecture) used very similar *local representations*

- How does the perceptron model differ?

- Why might these differences be important?

Log-Linear Tagging Models

- Take a history/tag pair (h, t).

- $\phi_s(h, t)$ for $s = 1 \ldots d$ are **features**
  $\mathbf{W}_s$ for $s = 1 \ldots d$ are **parameters**

- Conditional distribution:

$$P(t|h) = \frac{e^{\mathbf{W} \cdot \phi(h,t)}}{Z(h, \mathbf{W})}$$

where $Z(h, \mathbf{W}) = \sum_{t' \in \mathcal{T}} e^{\mathbf{W} \cdot \phi(h,t')}$

- Parameters estimated using maximum-likelihood
  e.g., iterative scaling, gradient descent

Log-Linear Tagging Models

- Word sequence $w_{[1:n]} = [w_1, w_2 \ldots w_n]$
- Tag sequence $t_{[1:n]} = [t_1, t_2 \ldots t_n]$
- Histories $h_i = \langle t_{i-1}, t_{i-2}, w_{[1:n]}, i \rangle$

$$\log P(t_{[1:n]} \mid w_{[1:n]})$$

$$= \sum_{i=1}^{n} \log P(t_i \mid h_i) = \underbrace{\sum_{i=1}^{n} \mathbf{W} \cdot \phi(h_i, t_i)}_{\text{Linear Score}} - \underbrace{\sum_{i=1}^{n} \log Z(h_i, \mathbf{W})}_{\substack{\text{Local Normalization} \\ \text{Terms}}}$$

- Compare this to the perceptron, where $\mathbf{GEN}$, $\mathbf{\Phi}$, $\mathbf{W}$ define

$$F(w_{[1:n]}) = \arg\max_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \underbrace{\sum_{i=1}^{n} \mathbf{W} \cdot \phi(h_i, t_i)}_{\text{Linear score}}$$

Problems with Locally Normalized models

- "Label bias" problem [Lafferty, McCallum and Pereira 2001] See also [Klein and Manning 2002]

- Example of a conditional distribution that locally normalized models can't capture (under bigram tag representation):

$$
\text{a b c} \Rightarrow
\begin{array}{ccc}
\text{A} & - \text{B} - & \text{C} \\
| & | & | \\
\text{a} & \text{b} & \text{c}
\end{array}
\quad \text{with } P(\text{A B C} \mid \text{a b c}) = 1
$$

$$
\text{a b e} \Rightarrow
\begin{array}{ccc}
\text{A} & - \text{D} - & \text{E} \\
| & | & | \\
\text{a} & \text{b} & \text{e}
\end{array}
\quad \text{with } P(\text{A D E} \mid \text{a b e}) = 1
$$

- Impossible to find parameters that satisfy

$$P(A \mid a) \times P(B \mid b, A) \times P(C \mid c, B) = 1$$

$$P(A \mid a) \times P(D \mid b, A) \times P(E \mid e, D) = 1$$

Overview

- Recap: global linear models, and boosting

- Log-linear models for parameter estimation

- An application: LFG parsing

- Global and local features

    – The perceptron revisited

    – Log-linear models revisited

Global Log-Linear Models

- We can use the parameters to define a probability for each tagged sequence:

$$P(t_{[1:n]} \mid w_{[1:n]}, \mathbf{W}) = \frac{e^{\sum_i \mathbf{W} \cdot \phi(h_i, t_i)}}{Z(w_{[1:n]}, \mathbf{W})}$$

where

$$Z(w_{[1:n]}, \mathbf{W}) = \sum_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} e^{\sum_i \mathbf{W} \cdot \phi(h_i, t_i)}$$

is a **global** normalization term

- This is a global log-linear model with

$$\mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) = \sum_i \phi(h_i, t_i)$$

Now we have:

$$\log P(t_{[1:n]} \mid w_{[1:n]})$$

$$= \underbrace{\sum_{i=1}^{n} \mathbf{W} \cdot \phi(h_i, t_i)}_{\text{Linear Score}} - \underbrace{\log Z(w_{[1:n]}, \mathbf{W})}_{\substack{\text{Global Normalization} \\ \text{Term}}}$$

When finding highest probability tag sequence, the global term is irrelevant:

$$\operatorname{argmax}_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \sum_{i=1}^{n} \left(\mathbf{W} \cdot \phi(h_i, t_i) - \log Z(w_{[1:n]}, \mathbf{W}) \right)$$

$$= \operatorname{argmax}_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \sum_{i=1}^{n} \mathbf{W} \cdot \phi(h_i, t_i)$$

Parameter Estimation

- For parameter estimation, we must calculate the gradient of

$$\log P(t_{[1:n]} \mid w_{[1:n]}) = \sum_{i=1}^{n} \mathbf{W} \cdot \phi(h_i, t_i) - \log \sum_{t'_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} e^{\sum_i \mathbf{W} \cdot \phi(h'_i, t'_i)}$$

   with respect to $\mathbf{W}$

- Taking derivatives gives

$$\frac{dL}{d\mathbf{W}} = \sum_{i=1}^{n} \phi(h_i, t_i) - \sum_{t'_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} P(t'_{[1:n]} \mid w_{[1:n]}, \mathbf{W}) \sum_{i=1}^{n} \phi(h'_i, t'_i)$$

- Can be calculated using dynamic programming!
  (very similar to forward-backward algorithm for EM training)

Summary of Perceptron vs. Global Log-Linear Model

- Both are global linear models, where

$$\mathbf{GEN}(w_{[1:n]}) \;=\; \text{the set of all possible tag sequences for } w_{[1:n]}$$

$$\mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) \;=\; \sum_i \phi(h_i, t_i)$$

- In both cases,

$$F(w_{[1:n]}) \;=\; \operatorname{argmax}_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \mathbf{W} \cdot \mathbf{\Phi}(w_{[1:n]}, t_{[1:n]})$$

$$\;=\; \operatorname{argmax}_{t_{[1:n]} \in \mathbf{GEN}(w_{[1:n]})} \sum_i \mathbf{W} \cdot \phi(h_i, t_i)$$

can be computed using dynamic programming

- Dynamic programming is also used in training:

  - Perceptron requires highest-scoring tag sequence for each training example

  - Global log-linear model requires gradient, and therefore "expected counts"

Results

From [Sha and Pereira, 2003]

- Task = shallow parsing (base noun-phrase recognition)

Model	Accuracy
SVM combination	94.39%
Conditional random field (global log-linear model)	94.38%
Generalized winnow	93.89%
Perceptron	94.09%
Local log-linear model	93.70%

References

[Abney 1997]   Abney

[ABR64]   Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. *Automation and Remote Control*, 25:821–837.

[Bod 98]   Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications/Cambridge University Press.

[Zhang 2002]   covering number paper

[Littlestone and Warmuth 1986]   Littlestone and Warmuth.

[CHA97]   Charniak, E. (1997). Statistical techniques for natural language parsing. In *AI Magazine*, Vol. 18, No. 4.

[COL00]   Collins, M. (2000). Discriminative Reranking for Natural Language Parsing. *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

[CD01]   Collins, M. and Duffy, N. (2001). Parsing with a Single Neuron: Convolution Kernels for Natural Language Problems. Technical report UCSC-CRL-01-01, University of California at Santa Cruz.

[CV95]   Cortes, C. and Vapnik, V. (1995). Support–Vector Networks. *Machine Learning*, 20(3):273–297.

[Work with Rob Schapire, Yoram Singer at COLT 2000]

[Friedman, Hastie, and Tibshirani 1998]   Friedman.

[Lafferty 1999]   lafferty.

[Lafferty, McCallum and Pereira 2001]   CRFs

[Della Pietra, Della Pietra, and Lafferty 1997]   Lafferty

[Freund and Schapire 99]   Freund, Y. and Schapire, R. (1999). Large Margin Classification using the Perceptron Algorithm. In *Machine Learning*, 37(3):277–296.

[FISS98] Freund, Y., Iyer, R.,Schapire, R.E., & Singer, Y. (1998). An effi cient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*. San Francisco: Morgan Kaufmann.

[HAU99] Haussler, D. (1999). *Convolution Kernels on Discrete Structures.* Technical report, University of Santa Cruz.

[Helmbold and Warmuth 95] Helmbold, D., and Warmuth, M. On Weak Learning.

[LCSW01] Lodhi, H., Cristianini, N., Shawe-Taylor, J., and Watkins, C. (2001). Text Classifi cation using String Kernels. To appear in *Advances in Neural Information Processing Systems 13*, MIT Press.

[Johnson et. al 1999] Johnson, M., Geman, S., Canon, S., Chi, S., & Riezler, S. (1999). Estimators for stochastic 'unifi cation-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

[Johnson, Geman, Canon, Chi, and Riezler 1999] Johnson, M., Geman, S., Canon, S., Chi, S., & Riezler, S. (1999). Estimators for stochastic 'unifi cation-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

[MSM93] Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics, 19*, 313-330.

[Prince and Smolensky] Prince and Smolensky. Optimality theory.

[Ramshaw and Marcus 95] Prince and Smolensky. Optimality theory.

[Ratnaparkhi 96]

[SSM99] Scholkopf, B., Smola, A.,and Muller, K.-R. (1999). Kernel principal component analysis. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods – SV Learning, pages 327-352. MIT Press, Cambridge, MA.

[Walker et al. 2001] Walker, M., Rambow, O., and Rogati, M. (2001). SPoT: a trainable sentence planner. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*.

[WAT00] Watkins, C. (2000). Dynamic alignment kernels. In A.J. Smola, P.L. Bartlett, B. Schlkopf, and D. Schuurmans, editors, Advances in Large Margin Classifi ers, pages 39-50, MIT Press.