A Tutorial on the Maths behind Conditional Random Fields for Sequential Labelling

Tran The Truyen, Dinh Phung
Department of Computing
Curtin University of Technology
thetruyen.tran@postgrad.curtin.edu.au, d.phung@curtin.edu.au

April 8, 2008

Contents

1 Introduction

This note describes the maths behind the Conditional Random Fields (CRFs) [4] for *sequential labelling*. In this type of problems, we are given a data sequence (e.g. a sentence, a Web page, or a sequence of signal measures) and we need to provide labels for each data point in the sequence. Often labels are drawn from a fixed set. For example, the set may include {`DATE,PERSON,LOCATION,JOB`} for a typical named entity recognition (NER) task; {`GOOD,PRICE,SIZE`} for a commercial information extraction application.

CRFs are a powerful machinery with wide range of applications. It is a right combination of several well-known components from different fields

- Dynamic programming, a subject of discrete maths
- Markov chains, an intersection between probability theory and graph theory
- Conditional logistic regression, a subject of statistics

- Large-scale numerical optimisation
- Computer - without this, CRFs would not be implemented efficiently for real world applications

We assume readers are familiar with college maths and computational techniques. We will approach the CRFs from its components first and then build up the final model last.

2 Dynamic Programming

Dynamic programming refers to a practice in which large problem is decomposed into smaller parts, and we reuse the part's results to compute the final output. Here, we are interested in two problems: finding (i) the sum and (ii) the max of the product of functions.

2.1 The Sum-product Problem

A simple example

Let us consider the following problem. We are given three variables x_1, x_2, x_3 and two *positive*[1] functions $\psi_1(x_1, x_2)$ and $\psi_2(x_2, x_3)$. Each of the three variables can receive 10^5 discrete values. The task is to compute

$$Z = \sum_{x_1}\sum_{x_2}\sum_{x_3} \psi_2(x_2, x_3)\psi_1(x_1, x_2)$$

It can be seen that, brute-force approach would require $10^{5\times 3}$ sum operations, which is clearly quite expensive. However, since we do not have to deal with three variables at once, we can exploit the distributive property of the summation

$$\begin{aligned} Z &= \sum_{x_3}\sum_{x2} \psi_2(x_2, x_3)\left(\sum_{x_1} \psi_1(x_1, x_2)\alpha_1[x_1]\right) \\ &= \sum_{x_3}\left(\sum_{x2} \psi_2(x_2, x_3)\alpha_2[x_2]\right) \\ &= \sum_{x_3} \alpha_3[x_3] \end{aligned}$$

where $\alpha_1[x_1] = 1$ for all x_1, $\alpha_2[x_2] = \sum_{x_1} \psi_1(x_1, x_2)\alpha_1[x_1]$ and $\alpha_3[x_3] = \sum_{x2} \psi_2(x_2, x_3)\alpha_2[x_2]$.

Here we do three summations one-by-one, each step takes 10^5 operations, thus the total time is $10^5 + 2 \times 10^5 \times 10^5$ operations.

This illustrates the main idea behind dynamic programming: we first deal with the sub-problems $\alpha_1[x_1]$, $\alpha_2[x_2]$ and $\alpha_3[x_3]$ and then pass the result up to the large problems.

Generalisation

This trick can generalise easily given that we want to sum over more than three variables and to deal with the product of more than two *pairwise* functions. Note that the functions must be pairwise for the trick to work.

For simplicity, assume that the variables receive values from the same set S. Each variable can be assigned to one of $|S|$ values, and we have T variables. We want to compute

$$Z = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_T}\left(\prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1})\right) \quad (1)$$

[1]Although we do not use this positivity, it is included here to be consistent with those used in CRFs.

We need an extra variable to simplify the presentation. Let $\alpha_t[x_t]$ be variables to hold results of the sub-problem at step t. Let $\alpha_1[x_1] = 1$. For $t > 1$, we have

$$\begin{aligned}
\alpha_2[x_2] &= \sum_{x_1} \psi_1(x_1, x_2)\alpha_1[x_1] && (2)\\
\alpha_3[x_3] &= \sum_{x_2} \psi_2(x_2, x_3)\alpha_2[x_2] && (3)\\
&\vdots \\
\alpha_t[x_t] &= \sum_{x_{t-1}} \psi_{t-1}(x_{t-1}, x_t)\alpha_{t-1}[x_{t-1}] && (4)
\end{aligned}$$

At the end of the sequence

$$Z = \sum_{x_T} \alpha_T[x_T]$$

Thus, the overall cost is $|S| + (T-1) \times |S|^2$ operations.

Indeed $\alpha_t[x_t]$ are sometimes called the ***forward*** variables. Since the chain is undirected, we may want to use a symmetric version of the forward which we will call the ***backward*** variables, denoted by $\beta_t[x_t]$. That is, $\beta_T[x_T] = 1$ for all $x_T \in S$ and the recursive relations for $t < T$ are almost identical to those in Equations 2-4

$$\beta_t[x_t] = \sum_{x_{t+1}} \psi_t(x_t, x_{t+1})\beta_{t+1}[x_{t+1}]$$

2.2 The Max-product Problem

We are now interested in a slightly different problem. Instead of computing the sum of product of pairwise functions, we want to find the maximal set of variable assignments, that is

$$(x_1^*, x_2^*, ..., x_T^*) = \arg \max_{x_1, x_2, ..., x_T} \left(\prod_{t \in [1, T-1]} \psi_t(x_t, x_{t+1}) \right)$$

Some of you may have recognised, this is actually a discrete combinatorial optimisation problem. This can be rewritten in the min-sum form

$$(x_1^*, x_2^*, ..., x_T^*) = \arg \min_{x_1, x_2, ..., x_T} \left(\sum_{t \in [1, T-1]} -\log \psi_t(x_t, x_{t+1}) \right)$$

The three variable problem

Let us start with the base-case of this problem in that there are only three variables x_1, x_2, x_3. Let $\alpha_1^*[x_1] = 1$ for all $x_1 \in S$. Now we first find the maximal value of the product of functions

$$\begin{aligned}
\Phi^* &= \max_{x_1} \max_{x_2} \max_{x_3} \left(\psi_2(x_2, x_3)\psi_1(x_1, x_2) \right) \\
&= \max_{x_3} \max_{x_2} \left(\psi_2(x_2, x_3) \left(\max_{x_1} \psi_1(x_1, x_2)\alpha_1^*[x_1] \right) \right) \\
&= \max_{x_3} \left(\max_{x_2} \left(\psi_2(x_2, x_3)\alpha_2^*[x_2] \right) \right) \\
&= \max_{x_3} \alpha_3^*[x_3] && (5)
\end{aligned}$$

where $\alpha_2^*[x_2] = \max_{x_1}\Big(\psi_1(x_1,x_2)\alpha_1^*[x_1]\Big)$ and $\alpha_3^*[x_3] = \max_{x_2}\Big(\psi_2(x_2,x_3)\alpha_2^*[x_2]\Big)$.

However, what we are really interested in is not the maximum product, but the maximal set of variable assignments x_1^*, x_2^*, x_3^*. Since $\max_{x_3}\alpha_3^*[x_3]$ is the maximum of the product, the maximal assignment of x_3 must be

$$x_3^* = \arg\max_{x_3}\alpha_3^*[x_3]$$

The next question is how to get x_1^*, x_2^*. Let us return back, and let us introduce some bookkeepers

$$\begin{aligned} Y_2[x_2] &= \arg\max_{x_1}\Big(\psi_1(x_1,x_2)\alpha_1^*[x_1]\Big) \\ Y_3[x_3] &= \arg\max_{x_2}\Big(\psi_2(x_2,x_3)\alpha_2^*[x_2]\Big) \end{aligned}$$

where $Y_2[x_2]$ and $Y_3[x_3]$ may store more than one variables.

It follows that $x_2^* \in Y[x_3^*]$. To see why, let assume that $x_2^* \notin Y[x_3^*]$ then

$$\begin{aligned} \alpha_3^*(x_3^*) &= \max_{x_2}\Big(\psi_2(x_2,x_3^*)\alpha_2^*[x_2]\Big) \\ &= \psi_2(Y_3[x_3^*],x_3^*)\alpha_2^*[Y_3(x_3^*)] \\ &< \psi_2(x_2^*,x_3^*)\alpha_2^*(x_2^*) \\ &= \psi_2(x_2^*,x_3^*)\max_{x_1}\Big(\psi_1(x_1,x_2^*)\alpha_1^*[x_1]\Big) \\ &= \psi_2(x_2^*,x_3^*)\psi_1(x_1^*,x_2^*) \\ &= \Phi^* \end{aligned}$$

which clearly contradicts Equation 5 that says $\Phi^* = \alpha_3^*[x_3^*]$. A similar argument leads to $x_1^* \in Y[x_2^*]$.

Generalisation

The procedure above can be generalised into arbitrary number of variables. There are two steps

- The *maximisation* step, where the forward variables are computed and bookkeepers are filled.
- The *backtracking* step, where we decode the maximal variable assignments

Let $\alpha_t^*[x_t]$ be variables to hold results of the sub-problem at step t. For $t > 1$, we have

$$\alpha_t^*[x_t] = \max_{x_{t-1}}\Big(\psi_{t-1}(x_{t-1},x_t)\alpha_{t-1}^*[x_{t-1}]\Big) \tag{6}$$

and the bookkeepers

$$Y_t[x_t] = \arg\max_{x_{t-1}}\Big(\psi_{t-1}(x_{t-1},x_t)\alpha_{t-1}^*[x_{t-1}]\Big)$$

We now backtrack to decode the maximal variables

$$\begin{aligned} x_T^* &= \arg\max_{x_T}\alpha_T^*[x_T] \\ x_t^* &\in Y_{t+1}[x_{t+1}^*], \quad \text{for } t \in [1, T-1] \end{aligned}$$

3 Markov Chains

3.1 Markov Assumption

Say we want to understand the evolution of some system over discrete time steps. At a particular time step $t-1$, the system is characterised by a state x_{t-1}. At the next time step t, the state changes to x_t. Normally, the new state should depend on all of the previous states $x_1, x_2, ..., x_{t-1}$ and the states after $x_{t+1}, x_{t+2}, ...$. However, this is a too strong assumption, because we never know how long the evolution will last. Modelling such dependency will be too hard for even simplest problems. The simplest assumption is to say that x_t only depends on x_{t-1}, x_{t+1} and *nothing else*! This is called the Markov assumption. Although it is simplistic, we will stick with it because luckily, it has actually been proven to be very useful.

A Markov chain is a representation of temporal evolution with Markov assumption. Given that we have T states, then we assume that x_t only depends on x_{t-1} and x_{t+1}. Formally, this is realised by

$$P(x_t|x_1, x_2, ..., x_{t-1,} x_{t+1}, ..., x_T) = P(x_t|x_{t-1,} x_{t+1})$$

Let $x = (x_1, x_2, ..., x_T)$ be the *joint* variable. If each the element variable takes discrete values from the set S of size $|S|$ then x takes values from the exponential set of size $|S|^T$, which is the number of all possible assignments of the sequence $(x_1, x_2, ..., x_T)$.

There exists a theorem of Hammersley-Clifford [5] that basically says that in order to satisfy the Markov assumption, the system must respect the following distribution

$$\begin{aligned} P(x) &= \frac{1}{Z}\Phi(x) \\ &= \frac{1}{Z}\prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1}) \end{aligned}$$

where $\psi_t(x_t, x_{t+1})$ are non-negative functions

$$\Phi(x) = \prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1}) \tag{7}$$

and

$$Z = \sum_x \prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1}) \tag{8}$$

Thus Z is the normalisation constant, also known as the partition function. Note that Z is a sum over exponentially many variable assignments. Fortunately, Z has the sum-product form, which we have computed in $\mathcal{O}(T|S|^2)$ steps in Section 2.1.

3.2 Joint Probability and Marginalisation

Given $P(x)$ as a distribution, the probability of a subset x_c of x is given as

$$P(x_c) = \sum_{x\backslash x_c} P(x)$$

where $x\backslash x_c$ denotes the remaining element variables beside x_c. This summation is sometimes called marginalisation. Computing probability of arbitrary subset x_c may not easy, but we often need only the singleton marginal $P(x_t)$ and the pairwise marginal $P(x_t, x_{t+1})$. Let us start with $P(x_t, x_{t+1})$ and leave $P(x_t)$ for readers.

Let us return to the set of Equations 2-4. Substituting $\alpha_{t-1}[x_{t-1}]$ into $\alpha_t[x_t]$, and $\alpha_{t-2}[x_{t-2}]$ into $\alpha_{t-1}[x_{t-1}]$, and so on we have

$$\alpha_t[x_t] = \sum_{x_{1:t-1}} \left[\prod_{i\in[2,t]} \psi_{i-1}(x_{i-1}, x_i) \right]$$

A similar trick gives us

$$\beta_t[x_t] = \sum_{x_{t+1:T}} \left[\prod_{j\in[t+1,T]} \psi_{j-1}(x_{j-1}, x_j) \right]$$

Denote by $x_{i:j}$ as a shorthand for $(x_i, x_{i+1}, ..., x_j)$. Now we proceed to compute $P(x_t)$

$$\begin{aligned} P(x_t, x_{t+1}) &= \sum_{x_{1:t-1}, x_{t+2:T}} \Pr(x) \\ &\propto \sum_{x_{1:t-1}, x_{t+2:T}} \Phi(x_{1:T}) \end{aligned} \tag{9}$$

Let $Z(x_t, x_{t+1}) = \sum_{x_{1:t-1}, x_{t+2:T}} \Phi(x_{1:T})$. By rearranging the factors in the RHS of Equation 7, we have

$$\Phi(x) = \left[\prod_{i\in[2,t]} \psi_{i-1}(x_{i-1}, x_i) \right] \psi_t(x_t, x_{t+1}) \left[\prod_{j\in[t+2,T]} \psi_{j-1}(x_{j-1}, x_j) \right]$$

then $Z(x_t, x_{t+1})$ can be written as

$$\begin{aligned} Z(x_t, x_{t+1}) &= \left(\sum_{x_{1:t-1}} \left[\prod_{i\in[2,t]} \psi_{i-1}(x_{i-1}, x_i) \right] \right) \times \psi_t(x_t, x_{t+1}) \times \\ &\quad \times \left(\sum_{x_{t+2:T}} \left[\prod_{j\in[t+2,T]} \psi_{j-1}(x_{j-1}, x_j) \right] \right) \\ &\propto \alpha_t[x_t] \psi_t(x_t, x_{t+1}) \beta_{t+1}[x_{t+1}] \end{aligned} \tag{10}$$

In other words, we have

$$P(x_t, x_{t+1}) = \kappa_t \alpha_t[x_t] \psi_t(x_t, x_{t+1}) \beta_{t+1}[x_{t+1}] \tag{11}$$

where κ_t are appropriate normalisation constants to ensure $\sum_{x_t, x_{t+1}} P(x_t, x_{t+1}) = 1$.

4 Conditional Logistic Regression

Let x and z be the random variables of interest, and $f_1(x, z), f_2(x, z), ..., f_K(x, z)$ are some real functions known as *features*. In our cases, we are mainly interested in discrete x while z can be anything and is always observed. In our applications, z plays the role of the raw data (e.g. text), while x plays the role of semantics (e.g. text categories like `SCIENCE`, `SPORT` or `CAR`) extracted from the data.

Assume that (x, z) are jointly generated by some unknown distribution $P(x, z)$. From the standard probability theory, we have the following factorisation

$$P(x, z) = P(x|z)P(z)$$

Here, we are not really interested in modelling $P(z)$. We want to model $P(x|z)$ as a distribution of the following parametric form

$$P(x|z,\mathbf{w}) = \frac{1}{Z(z)} \exp(\sum_{k\in[1,K]} w_k f_k(x,z)) \tag{12}$$

where $Z(z) = \sum_x \exp(\sum_{k\in[1,K]} w_k f_k(x,z))$ is the normalisation constant and $\mathbf{w} = (w_1, w_2, ..., w_K)$ are the weights of corresponding features $(f_1(x,z), f_2(x,z), ..., f_K(x,z))$. The weight w_k specifies how much the feature $f_k(x,z)$ contributes to the model probability.

How do we get the weights? If we have a good understanding of the domain, then we may manually specify the weights. But it is not generally the case. Fortunately, what we often have a set of observed values of the random variable $\mathcal{D} = (\tilde{x}^1, z^1); (\tilde{x}^2, z^2); ...; (\tilde{x}^D, z^D)$. For the modelling purpose, we assume that these values are *independently randomly* drawn from the distribution $P(x,z)$.

A theoretically sound method to estimate w_k is to find the optimal w_k^* that helps to maximise the (log) likelihood of the observation

$$\begin{aligned}
\mathbf{w}^* &= \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}), \text{ where} \\
\mathcal{L}(\mathbf{w}) &= \frac{1}{D} \log \prod_{d\in[1,D]} P(\tilde{x}^d|z^d,\mathbf{w}) \\
&= \frac{1}{D} \sum_{d\in[1,D]} \log P(\tilde{x}^d|z^d,\mathbf{w}) \\
&= \frac{1}{D} \sum_{d\in[1,D]} \left(\sum_{k\in[1,K]} w_k f_k(\tilde{x}^d, z^d) - \log Z(z^d) \right) \qquad (13)
\end{aligned}$$

There exists several efficient methods to find the maximum likelihood solution including the scaling methods like Generalised Iterative Scaling (GIS) [2], the Improved Iterative Scaling (IIS) [1], and gradient-based methods like Conjugate Gradients (CG) [3] and L-BFGS [6]. See [7] for a comparison. The most efficient methods are gradient-based, in that they seek to solve the zero gradient problem

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_k} &= \frac{1}{D} \sum_{d\in[1,D]} \left(f_k(\tilde{x}^d, z^d) - \frac{1}{Z(z^d)} \sum_x \exp(\sum_{k\in[1,K]} w_k f_k(x, z^d)) f_k(x, z^d) \right) \\
&= \frac{1}{D} \sum_{d\in[1,D]} \left(f_k(\tilde{x}^d, z^d) - \sum_x P(x|z^d) f_k(x, z^d) \right) \qquad (14) \\
&= \tilde{\mathbb{E}}[f_k] - \mathbb{E}[f_k] \\
&= 0
\end{aligned}$$

where

$$\begin{aligned}
\tilde{\mathbb{E}}[f_k] &= \frac{1}{D} \sum_{d\in[1,D]} f_k(\tilde{x}^d, z^d) \\
\mathbb{E}[f_k] &= \frac{1}{D} \sum_{d\in[1,D]} \sum_x P(x|z^d) f_k(x, z^d)
\end{aligned}$$

Since $\tilde{\mathbb{E}}[f_k]$ depends only on the observed data, it is sometimes called the empirical feature expectation. Likewise, $\mathbb{E}[f_k]$ is known as model feature expectation.

5 Conditional Random Fields

We now have enough ingredients to build up the Conditional Random Fields (CRFs). A CRF, in essence, is a combination of undirected Markov chain and conditional logistic regression. Undirected Markov chain is used to model the sequential structure of x and conditional logistic regression is used to derive the parametric forms. As before, we are given z as observed raw data, and x to be the semantics (or patterns) to be extracted from x.

More specifically, for each z, we assume that x when conditioned on z is a undirected Markov chain

$$P(x|z) = \frac{1}{Z(z)} \prod_{t \in [1,T-1]} \psi_t(x_t, x_{t+1}, z) \tag{15}$$

where $Z(z) = \sum_x \prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1}, z)$ and all of these conditional distributions share the same parameter set $\mathbf{w}$

$$\psi_t(x_t, x_{t+1}, z) = \exp(\sum_{k\in[1,K]} w_k f_k(x_t, x_{t+1}, z))$$

Let $F_k(x,z) = \sum_{t\in[1,T-1]} f_k(x_t, x_{t+1}, z)$. Equation 15 reduces to

$$\begin{aligned} P(x|z) &= \frac{1}{Z(z)} \exp(\sum_{t\in[1,T-1]} \sum_{k\in[1,K]} w_k f_k(x_t, x_{t+1}, z)) \\ &= \frac{1}{Z(z)} \exp(\sum_{k\in[1,K]} w_k F_k(x_t, x_{t+1}, z)) \end{aligned}$$

which is essentially the conditional logistic regression in Equation 12.

There are two main problems associated with CRFs (and any statistical pattern recognition methods in general)

- *Learning*: this is to estimate the parameter $\mathbf{w}$ using an observed dataset
- *Decoding*: this is to find the optimal pattern $x^* = (x_1^*, x_2^*, ..., x_T^*)$ for a given raw data z.

5.1 Learning

As before, given a dataset $\mathcal{D} = (\tilde{x}^1, z^1); (\tilde{x}^2, z^2); ...; (\tilde{x}^D, z^D)$, we want to estimate the parameter $\mathbf{w}$ using the maximum likelihood method. Since the CRF is actually a conditional logistic regression, learning using gradients requires to evaluate two main quantities

- The log-likelihood of the data as in Equation 13.
- The gradient of the log-likelihood as in Equation 14.

Computing the log-likelihood in turns requires the normalisation constants $Z(z^d)$ for each observable z^d. We have

$$Z(z) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_T} \prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1}, z)$$

which is essentially the sum-product form in Equation 1.

The gradient of the log-likelihood has two parts: the empirical expectation of feature $\tilde{\mathbb{E}}[F_k]$ and the model expectation $\mathbb{E}[F_k]$. The empirical expectation is straightforward to compute

$$\tilde{\mathbb{E}}[F_k] = \frac{1}{D} \sum_{d\in[1,D]} \sum_{t\in[1,T-1]} f_k(\tilde{x}_t, \tilde{x}_{t+1}, z^d)$$

The model expectation can be expanded as

$$
\begin{aligned}
\mathbb{E}[f_k] &= \frac{1}{D}\sum_{d\in[1,D]}\sum_{x} P(x|z^d) \sum_{t\in[1,T-1]} f_k(x_t, x_{t+1}, z^d) \\
&= \frac{1}{D}\sum_{d\in[1,D]}\sum_{x} P(x|z^d) \sum_{t\in[1,T-1]} f_k(x_t, x_{t+1}, z^d) \\
&= \frac{1}{D}\sum_{d\in[1,D]}\sum_{t\in[1,T-1]}\sum_{x_t,x_{t+1}} P(x_t, x_{t+1}|z^d) f_k(x_t, x_{t+1}, z^d)
\end{aligned}
$$

Thus the model expectation depends on $P(x_t, x_{t+1}|z^d)$ which we know how to compute efficiently in Section 3.2.

5.2 Decoding

Once the parameter has bee estimated, at test time, we are given raw data z and need to find the optimal x^* in the following sense

$$x^* = \arg\max_x P(x|z)$$

which translates to

$$x^* = \arg\max_x \prod_{t\in[1,T-1]} \psi_t(x_t, x_{t+1}, z)$$

which is essentially the max-product problem described in Section 2.2.

6 Final Notes

CRFs are not an overnight invention. It is basically a conditional version of Markov random fields (MRFs) [5] - a well-known representation scheme for stochastic patterns. CRFs are not limited to sequential problems. It can be applied to anything where MRFs are able to represent. If we use MRFs to represent relational data, then we will get the Relational Markov Networks [9][8].

References

[1] Adam Berger. The Improved Iterative Scaling algorithm: A gentle introduction. URL: http://www.cs.cmu.edu/afs/cs/user/aberger/ www/ps/scaling.ps, 1997.

[2] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 42:1470–1480, 1972.

[3] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[4] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine learning (ICML)*, pages 282–289, 2001.

[5] S.L. Lauritzen. *Graphical Models.* Oxford Science Publications, 1996.

[6] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization methods. *Mathematical Programming*, 45:503–528, 1989.

[7] Robert Malouf. A comparison of algorithms for Maximum Entropy parameter estimation. In Dan Roth and Antal van den Bosch, editors, *Proceedings of the 6th Conference on Natural Language Learning (CoNLL)*, pages 49–55, Taipei, 2002.

[8] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 4, pages 93–128. MIT Press, 2006.

[9] B. Taskar, A. Pieter, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–49. Morgan Kaufmann, 2002.