technische universität
dortmund

Classical Probabilistic Models and Conditional Random Fields

Roman Klinger
Katrin Tomanek

Algorithm Engineering Report
TR07-2-013
December 2007
ISSN 1864-4503

Classical Probabilistic Models and Conditional Random Fields

Roman Klinger*
Fraunhofer Institute for
Algorithms and Scientific Computing (SCAI)
Schloss Birlinghoven
53754 Sankt Augustin, Germany

Dortmund University of Technology
Department of Computer Science
Chair of Algorithm Engineering (Ls XI)
44221 Dortmund, Germany

roman.klinger@scai.fhg.de
roman.klinger@udo.edu

Katrin Tomanek*
Jena University
Language & Information Engineering (JULIE) La
Fürstengraben 30
07743 Jena, Germany

katrin.tomanek@uni-jena.de

Contents

* The authors contributed equally to this work.

1 Introduction

Classification is known as the assignment of a class $y \in \mathcal{Y}$ to an observation $x \in \mathcal{X}$. This task can be approached with probability theory by specifying a probability distribution to select the most likely class y for a given observation x.

A well-known example (Russell and Norvig, 2003) is the classification of weather observations into categories, such as *good* or *bad*, $\mathcal{Y} = \{good, bad\}$. For instance, let x be the weather observation on a special day, $\mathcal{X} = \{Monday, Tuesday, \ldots\}$. Now, x can be described by a set of features such as $f_{cloudy}(x) = 1$, if and only if it is cloudy on day x, $f_{cloudy}(x) = 0$ otherwise. Other features might be f_{sunny} or f_{rainy}. In general, features do not necessarily have to be binary.

Modeling all dependencies in a probability distribution is typically very complex due to interdependencies between features. The Naïve Bayes assumption of all features being conditionally independent is an approach to address this problem (see Section 2.1). In nearly all probabilistic models such independence assumptions are made for some variables to make necessary computations manageable.

In the structured learning scenario, multiple and typically interdependent class and observation variables are considered which implicates an even higher complexity in the probability distribution. This is the case for image or music data as well as for natural language text. As for images, pixels near to each other are very likely to have a similar color or hue. In music, different succeeding notes follow special laws, they are not independent, especially when they sound simultaneously. Otherwise, music would not be pleasant to the ear. In text, words are not an arbitrary accumulation, the order is important and grammatical constraints hold.

A typical task in natural language processing is known as text segmentation which means the classification of units of a textual sequence. Such units are words or other symbols. For each unit a category $y \in \mathcal{Y}$ has to be assigned. Categories might be linguistically motivated, such as part-of-speech tags, or person names or cities, as in the following text snippet:

```
[...]  Mister George W. Bush arrived in Rome together with [...]
```

Here, the task is to assign a fitting label (also called output) sequence such as

```
[...]  O name name name O O city O O [...]
```

where each label represents an entity class.[1]

One approach for modeling linear sequence structures, as can be found in natural language text, are Hidden Markov Models (Rabiner, 1989). For the sake of complexity reduction, strong independence assumptions between the observation variables are made. This impairs the accuracy of the model. Conditional Random Fields (CRFs, Lafferty et al. (2001)) are developed exactly to fill that gap: While CRFs make similar assumptions on the dependencies among the class variables, no assumptions on the dependencies among observation variables need to be made (see Section 4).

[1] In this example the classes are `name`, `city`, and `O` where the latter is defined as not being an entity of interest.

CRFs have found application in many domains which deal with structured data. Despite the frequent application of linear-chain CRFs, also other underlying structures have been used to model the respective dependencies among the class variables. Especially in natural language processing, CRFs are currently a state-of-the-art technique for many of its subtasks including basic text segmentation (Tomanek et al., 2007), part-of-speech tagging (Lafferty et al., 2001), shallow parsing (Sha and Pereira, 2003), the resolution of elliptical noun phrases (Buyko et al., 2007). CRFs have been proven to be very useful in named entity recognition, especially on documents from the biomedical domain (Settles, 2004; McDonald and Pereira, 2005; Klinger et al., 2007a,b; McDonald et al., 2004). Furthermore, CRFs have been applied to gene prediction (DeCaprio et al., 2007), image labeling (He et al., 2004) and object recognition (Quattoni et al., 2005), and also in telematics for intrusion detection (Gupta et al., 2007) and sensor data management (Zhang et al., 2007).

This paper aims at giving an overview of the basic theory behind Conditional Random Fields and illustrates how these are related to other probabilistic models. In Section 2, a brief overview of three classical and well-established probabilistic models is given: Naïve Bayes, Hidden Markov, and Maximum Entropy. The relations between and graphical representations of these different approaches are discussed in Section 3. In Section 4, the basic concepts of CRFs (4.1) are explained. This section is mainly focused on the special case of linear-chain CRFs (4.2) and methods for training (4.2.1) and inference (4.2.2). Moreover, building upon these explanations, a generalization to arbitrarily structured CRFs is given in Section 4.3.

For further reading we recommend the tutorials of Wallach (2004) and Sutton and McCallum (2007). They approach the theory behind CRFs from a different perspective.

2 Probabilistic Models

In this section, some well-known probabilistic models are discussed. Conditional Random Fields are founded on the underlying ideas and concepts of these approaches.

The Naïve Bayes Model is an approach to classify single class variables in dependence of several feature values. In that model, the input values are assumed to be conditionally independent. It is a so called generative approach, modeling the joint probability $p(y, \vec{x})$ of the input values $\vec{x}$ and the class variable y. The Hidden Markov Model is an extension to the Naïve Bayes Model for sequentially structured data also representing the dependencies of the variables $\vec{x}$ and $\vec{y}$ as a joint probability distribution.

Modeling joint probabilities has disadvantages due to computational complexity. The Maximum Entropy Model, in contrast, is based on modeling the conditional probability $p(y|x)$. Like the Naïve Bayes Model, it is an approach to classify a single class variable in dependence of several feature values. The difference is the consideration of conditional probability $p(y|x)$ instead of the joint probability.

While a Hidden Markov Model is a sequential extension to the Naïve Bayes Model, Conditional Random Fields can be understood as a sequential extension to the Maximum Entropy Model. Both Maximum Entropy Models and Conditional Random Fields are

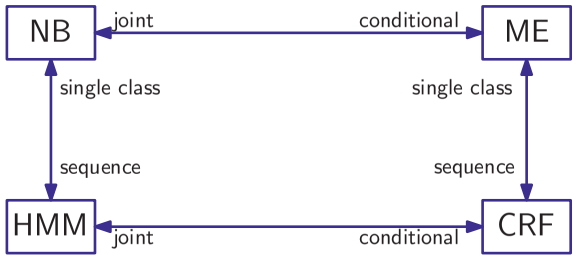


Figure 1: Overview of probabilistic models: Naïve Bayes Model (NB), Hidden Markov Model (HMM), Maximum Entropy Model (ME), and Conditional Random Field (CRF). Depicted aspects are joint versus conditional probability, single class prediction versus prediction on sequential data.

known as discriminative approaches.

A graphical comparison of these models is given in Figure 1. In the following, a detailed explanation of these models is given based on Bishop (2006) and Russell and Norvig (2003).

2.1 Naïve Bayes

A conditional probability model is a probability distribution $p(y|\vec{x})$ with an input vector $\vec{x} = (x_1, \ldots, x_m)$, where x_i $(1 \leq i \leq m)$ are features and y is the class variable to be predicted. That probability can be formulated with Bayes' law:

$$p(y|\vec{x}) = \frac{p(y)p(\vec{x}|y)}{p(\vec{x})}\,. \tag{1}$$

The denominator $p(\vec{x})$ is not important for classification as it can be understood as a normalization constant which can be computed by considering all possible values for y. The numerator can also be written as a joint probability

$$p(y)p(\vec{x}|y) = p(y, \vec{x})\,, \tag{2}$$

which can be too complex to be computed directly (especially when the number of components in $\vec{x}$ is high). A general decomposition of that probability can be formulated applying the chain rule $p(x_1, \ldots, x_m) = \prod_{i=2}^{m} p(x_i|x_{i-1}, \ldots, x_1)$:

$$p(y, \vec{x}) = p(y) \prod_{i=2}^{m} p(x_i|x_{i-1}, \ldots, x_1, y)\,. \tag{3}$$

In practice, it is often assumed, that all input variables x_i are conditionally independent of each other which is known as the Naïve Bayes assumption (Hand and Yu, 2001). That

means that $p(x_i|y, x_j) = p(x_i|y)$ holds for all $i \neq j$. Based on this simplification, a model known as the Naïve Bayes classifier is formulated as[2]

$$p(y|\vec{x}) \propto p(y, \vec{x}) = p(y) \prod_{i=1}^{m} p(x_i|y)\,. \tag{4}$$

This probability distribution is less complex than the one formulated in equation 3. Dependencies between the input variables $\vec{x}$ are not modeled, probably leading to an imperfect representation of the real world. Nevertheless, the Naïve Bayes Model performs surprisingly well in many real world applications, such as email classification (Androutsopoulos et al., 2000a,b; Kiritchenko and Matwin, 2001).

2.2 Hidden Markov Models

In the Naïve Bayes Model, only single output variables have been considered. To predict a sequence of class variables $\vec{y} = (y_1, \ldots, y_n)$ for an observation sequence $\vec{x} = (x_1, \ldots, x_n)$, a simple sequence model can be formulated as a product over single Naïve Bayes Models. Dependencies between single sequence positions are not taken into account. Note, that in contrast to the Naïve Bayes Model there is only one feature at each sequence position, namely the identity of the respective observation:

$$p(\vec{y}, \vec{x}) = \prod_{i=1}^{n} p(y_i) \cdot p(x_i|y_i)\,. \tag{5}$$

Each observation x_i depends only on the class variable y_i at the respective sequence position[3]. Due to this independence assumption, transition probabilities from one step to another are not included in this model. In fact, this assumption is hardly ever met in practice resulting in limited performance of such models. Thus, it is reasonable to assume that there are dependencies between the observations at consecutive sequence positions. To model this, state transition probabilities are added:[4]

$$p(\vec{y}, \vec{x}) = \prod_{i=0}^{n} p(y_i|y_{i-1})p(x_i|y_i)\,. \tag{6}$$

This leads to the well-known *Hidden Markov model* (HMM, Rabiner (1989))

$$P(\vec{x}) = \sum_{y \in \mathcal{Y}} \prod_{i=0}^{n} p(y_i|y_{i-1})p(x_i|y_i)\,, \tag{7}$$

[2] $A \propto B$ indicates that A is proportional to B. Here, proportionality is given because of omission the denominator.

[3] Recall that x_i are different observations at different sequence positions. In equation 4, in contrast, x_i specifies different observations at the same position.

[4] The initial probability distribution is assumed to be included as $p(y_0|y_{-1}) = p(y_0)$

where $\mathcal{Y}$ is the set of all possible label sequences $\vec{y}$.

Dependencies between output variables $\vec{y}$ are modeled. A shortcoming is the assumption of conditional independence (see equation 6) between the input variables $\vec{x}$ due to complexity issues. As we will see later, CRFs address exactly this problem.

2.3 Maximum Entropy Model

The two models introduced in Section 2.1 and 2.2 are trained to maximize the joint likelihood. In the following, the *Maximum Entropy Model*[5] is discussed in more detail as it is fundamentally related to CRFs. The Maximum Entropy Model is a conditional probability model. It is based on the *Principle of Maximum Entropy* (Jaynes, 1957) which states that if incomplete information about a probability distribution is available, the only unbiased assumption that can be made is a distribution which is as uniform as possible given the available information. Under this assumption, the proper probability distribution is the one which maximizes the entropy given the constraints from the training material. For the conditional model $p(y|x)$ the conditional entropy $H(y|x)$ (Korn and Korn, 2000; Bishop, 2006) is applied, which is defined as

$$H(y|x) = -\sum_{(x,y)\in\mathcal{Z}} p(y,x)\log p(y|x)\,. \tag{8}$$

The set $\mathcal{Z} = \mathcal{X}\times\mathcal{Y}$ consists of $\mathcal{X}$, the set of all possible input variables x, and $\mathcal{Y}$, the set of all possible output variables y. Note that $\mathcal{Z}$ contains not only the combinations of x and y occurring in the training data, but all possible combinations.

The basic idea behind Maximum Entropy Models is to find the model $p^*(y|x)$ which on the one hand has the largest possible conditional entropy but is on the other hand still consistent with the information from the training material. The objective function, later referred to as *primal problem*, is thus

$$p^*(y|x) = \underset{p(y|x)\in P}{\operatorname{argmax}}\, H(y|x)\,, \tag{9}$$

where P is the set of all models consistent with the training material. What is meant with "consistent" will be explained in detail later on page 8.

The training material is represented by features. Here, these are defined as binary-valued functions[6] $f_i(x,y)\in\{0,1\}$ $(1\le i\le m)$ which depend on both the input variable x and the class variable y. An example for such a function is:

$$f_i(x,y) = \begin{cases} 1 & \text{if } y = \texttt{name} \text{ and } x = \texttt{Mister} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

[5] These explanations of Maximum Entropy Models are based on the Maximum Entropy tutorial by Berger et al. (1996).

[6] Such indicator functions are referred to as *features* instead of *feature functions* throughout the rest of the paper.

The expected value of each feature f_i is estimated from the empirical distribution $\tilde{p}(x,y)$. The empirical distribution is obtained by simply counting how often the different values of the variables occur in the training data:

$$\tilde{E}(f_i) = \sum_{(x,y)\in\mathcal{Z}} \tilde{p}(x,y) f_i(x,y)\,. \tag{11}$$

All possible pairs (x,y) are taken into account here. As the empirical probability for a pair (x,y) which is not contained in the training material is 0, $\tilde{E}(f_i)$ can be rewritten as

$$\tilde{E}(f_i) = \frac{1}{N}\sum_{(x,y)\in\mathcal{T}} f_i(x,y)\,. \tag{12}$$

The size of the training set is $N = |\mathcal{T}|$. Thus, $\tilde{E}(f_i)$ can be calculated by counting how often a feature f_i is found with value 1 in the training data $\mathcal{T}\subseteq\mathcal{Z}$[7] and dividing that number by the size N of the training set.

Analogously to equation 11, the expected value of a feature on the model distribution is defined as

$$E(f_i) = \sum_{(x,y)\in\mathcal{Z}} p(x,y) f_i(x,y)\,. \tag{13}$$

In contrast to equation 11 (the expected value on the empirical distribution), the model distribution is taken into account here. Of course, $p(x,y)$ cannot be calculated in general because the number of all possible $x\in\mathcal{X}$ can be enormous. This can be addressed by rewriting $E(f_i)$ by

$$E(f_i) = \sum_{(x,y)\in\mathcal{Z}} p(x)p(y|x) f_i(x,y) \tag{14}$$

and substituting $p(x)$ with the empirical distribution $\tilde{p}(x)$. This is an approximation to make the calculation of $E(f_i)$ possible (see Lau et al. (1993) for a more detailed discussion). This results in

$$E(f_i) \approx \sum_{(x,y)\in\mathcal{Z}} \tilde{p}(x)p(y|x) f_i(x,y)\,, \tag{15}$$

which can (analogously to equation 12) be transformed into

$$E(f_i) = \frac{1}{N}\sum_{x\in\mathcal{T}}\sum_{y\in\mathcal{Y}} p(y|x) f_i(x,y)\,. \tag{16}$$

Only x values occurring in the training data are considered $(x\in\mathcal{T})$ while all possible y values are taken into account $(y\in\mathcal{Y})$. In many applications the set $\mathcal{Y}$ typically contains

[7] In fact, $\mathcal{T}$ is a multiset as the elements from $\mathcal{Z}$ can be contained several times. So the subset relation $\mathcal{T}\subseteq\mathcal{Z}$ only holds in a special case.

only a small number of variables. Thus, summing over y is possible here and $E(f_i)$ can be calculated efficiently.

Equation 9 postulates that the model $p^*(y|x)$ is consistent with the evidence found in the training material. That means, for each feature f_i its expected value on the empirical distribution must be equal to its expected value on the particular model distribution, these are the first m constraints

$$E(f_i) = \tilde{E}(f_i)\,. \tag{17}$$

Another constraint is to have a proper conditional probability ensured by

$$p(y|x) \geq 0 \text{ for all } x, y \qquad \sum_{y \in \mathcal{Y}} p(y|x) = 1 \text{ for all } x. \tag{18}$$

Finding $p^*(y|x)$ under these constraints can be formulated as a constrained optimization problem. For each constraint a Lagrange multiplier λ_i is introduced. This leads to the following Lagrange function $\Lambda(p, \vec{\lambda})$:

$$\Lambda(p,\vec{\lambda}) = \underbrace{H(y|x)}_{\substack{\text{primal problem}\\ \text{equation 9}}} + \sum_{i=1}^{m} \lambda_i \underbrace{\Big(E(f_i) - \tilde{E}(f_i)\Big)}_{\substack{\stackrel{!}{=} 0\\ \text{constraints from}\\ \text{equation 17}}} + \lambda_{m+1} \underbrace{\left(\sum_{y \in \mathcal{Y}} p(y|x) - 1\right)}_{\substack{\stackrel{!}{=} 0\\ \text{constraint from}\\ \text{equation 18}}} \tag{19}$$

This is maximized to get the model formulation $p^*_{\vec{\lambda}}(y|x)$ in equation 28 on page 10. In the following, a detailed derivation is given.

In the same manner as done for the expectation values in equation 15, $H(y|x)$ is approximated:

$$H(y|x) \approx -\sum_{(x,y) \in \mathcal{Z}} \tilde{p}(x) p(y|x) \log p(y|x)\,. \tag{20}$$

The derivation of equation 20 is given by

$$\begin{aligned}\frac{\partial}{\partial p(y|x)} H(y|x) &= -\tilde{p}(x)\left(\log p(y|x) + \frac{p(y|x)}{p(y|x)}\right)\\ &= -\tilde{p}(x)\left(\log p(y|x) + 1\right)\,.\end{aligned} \tag{21}$$

The derivation of the first m constraints in equation 19 is

$$\begin{aligned}\frac{\partial}{\partial p(y|x)} \sum_{i=1}^{m} \lambda_i \Big(E(f_i) - \tilde{E}(f_i)\Big) &= \\ = \frac{\partial}{\partial p(y|x)} \sum_{i=1}^{m} \lambda_i \left(\sum_{(x,y) \in \mathcal{Z}} \tilde{p}(x) p(y|x) f_i(x,y) - \left(\sum_{(x,y) \in \mathcal{Z}} \tilde{p}(x,y) f_i(x,y)\right)\right)&\\ = \sum_{i=1}^{m} \lambda_i \tilde{p}(x) f_i(x,y)\,.&\end{aligned} \tag{22}$$

The complete derivation of the Lagrange function from equation 19 is then:

$$\frac{\partial}{\partial p(y|x)} \Lambda(p, \vec{\lambda}) = -\tilde{p}(x)(1 + \log p(y|x)) + \sum_{i=1}^{m} \lambda_i \tilde{p}(x) f_i(x,y) + \lambda_{m+1}\,. \tag{23}$$

Equating this term to 0 and solving by $p(y|x)$ leads to

$$\begin{aligned}0 &= -\tilde{p}(x)(1 + \log p(y|x)) + \sum_{i=1}^{m} \lambda_i \tilde{p}(x) f_i(x,y) + \lambda_{m+1}\\ \tilde{p}(x)(1 + \log p(y|x)) &= \sum_{i=1}^{m} \lambda_i \tilde{p}(x) f_i(x,y) + \lambda_{m+1}\\ 1 + \log p(y|x) &= \sum_{i=1}^{m} \lambda_i f_i(x,y) + \frac{\lambda_{m+1}}{\tilde{p}(x)}\\ \log p(y|x) &= \sum_{i=1}^{m} \lambda_i f_i(x,y) + \frac{\lambda_{m+1}}{\tilde{p}(x)} - 1\\ p(y|x) &= \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right) \cdot \exp\left(\frac{\lambda_{m+1}}{\tilde{p}(x)} - 1\right)\,.\end{aligned} \tag{24}$$

The second constraint in equation 18 is given as

$$\sum_{y \in \mathcal{Y}} p(y|x) = 1\,. \tag{25}$$

Substituting equation 24 into 25 results in

$$\begin{aligned}\sum_{y \in \mathcal{Y}} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right) \cdot \exp\left(\frac{\lambda_{m+1}}{\tilde{p}(x)} - 1\right) &= 1\\ \exp\left(\frac{\lambda_{m+1}}{\tilde{p}(x)} - 1\right) &= \frac{1}{\sum_{y \in \mathcal{Y}} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right)}\,.\end{aligned} \tag{26}$$

Substituting equation 26 back into equation 24 results in

$$p(y|x) = \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right) \cdot \frac{1}{\sum_{y \in \mathcal{Y}} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right)}\,. \tag{27}$$

This is the general form the model needs to have to meet the constraints. The Maximum Entropy Model can then be formulated as

$$p^*_{\vec{\lambda}}(y|x) = \frac{1}{Z_{\vec{\lambda}}(x)} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right), \tag{28}$$

and $Z_{\vec{\lambda}}(x)$ then is

$$Z_{\vec{\lambda}}(x) = \sum_{y \in \mathcal{Y}} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right). \tag{29}$$

This formulation of a conditional probability distribution as a log-linear model and a product of exponentiated weighted features is discussed from another perspective in Section 3. In Section 4, the similarity of Conditional Random Fields, which are also log-linear models, with the conceptually closely related Maximum Entropy Models becomes evident.

For a more detailed discussion of Maximum Entropy Models and related approaches we recommend the book by Pearl (1988) and the Maximum Entropy Tutorial by Berger et al. (1996).

In this section, two kinds of probabilistic models have been introduced. On the one hand *generative* models, such as Naïve Bayes and Hidden Markov Models, which are based on joint probability distributions. As can be seen in formula 4 and 6, in such models the observation variables x_i topologically precede, also called "*generate*", the input variables y. This characteristic can be seen in the graphical representation (see Section 3), of these models in Figures 3(a) and 4(a). On the other hand, *discriminative* models, such as Maximum Entropy Models, are based on conditional probability distributions. In the next section, both groups of models are reviewed from a different perspective: their graphical representations.

3 Graphical Representation

The underlying probability distributions of probabilistic models can be represented in a graphical form, this is why they are often called probabilistic *graphical* models. The following explanations are inspired by Bishop (2006).

A *probabilistic graphical model* is a diagrammatic representation of a probability distribution. In such a graph there is a node for each random variable. The absence of an edge between two variables represents *conditional independence* between those variables. Conditional independence means that two random variables a and b are independent given a third random variable c if they are independent in their conditional probability distribution, formally $p(a,b|c) = p(a|b)p(b|c)$.[8] From such graphs, also

[8] Note that in contrast two random variables a and b are *statistically* independent if and only if $p(a,b) = p(a)p(b)$.

called *independency graphs*, one can read the conditional independence properties of the underlying distribution. Note that a fully connected independency graph does not contain any information about the probability distribution, only the absence of edges is informative: Conditional independence in the probability distribution does not induce the absence of the edge in the graph.[9]

Conditional independence is an important concept as it can be used to decompose complex probability distributions into a product of factors, each consisting of the subset of corresponding random variables. This concept makes complex computations (which are for example necessary for learning or inference) much more efficient. In general, the *decomposition*, in fact a *factorization* of a probability distribution, is written as the product of its factors Ψ_s, with $\vec{v}_s$ the subset of the respective random variables constituting such a factor

$$p(\vec{v}) = \prod_s \Psi_s(\vec{v}_s). \tag{30}$$

Let $G = (V, E)$ be a graph with vertexes V and edges E. In an *independency graph* (for example the one shown in Figure 2(a)), the vertexes $V = X \cup Y$, with X and Y sets of random variables, are depicted by circles. X will typically be considered as the set of input or observation variables (shaded circles), and Y as the set of output variables (empty nodes). An independency graph can have directed or undirected edges, depending on the kind of graphical model it represents (see Sections 3.1 and 3.2).

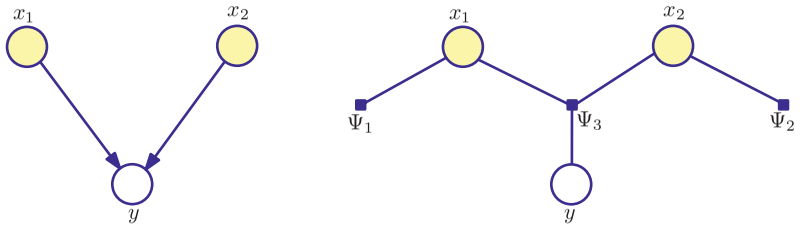
In a *factor graph* (Kschischang et al., 2001), such as the one shown in Figure 2(b), the circles represent, as in an independency graph, the random variables of the underlying distribution, depicted by circles. Further, a factor graph contains factor nodes, depicted by small, filled squares, which represent the factors Ψ_s (compare with equation 30).[10] In a factor graph, the edges are always undirected, linking the random variables to the factor nodes. A factor Ψ_s includes all random variables to which the respective factor node is directly connected by an edge. Thus, a factor graph represents more explicitly the factorization of the underlying probability distribution. Independency graphs of both directed and undirected graphical models can be transformed into factor graphs.

As an example, assume a probability distribution $p(x_1, x_2, y)$ to factorize as $p(\vec{x}) = p(x_1)p(x_2)p(y|x_1,x_2)$. It has the factors $\Psi_1(x_1) = p(x_1)$, $\Psi_2(x_2) = p(x_2)$, and $\Psi_3(y) = p(y|x_1,x_2)$. Here, x_1 and x_2 are conditionally independent given y. Figure 2 shows an independency graph and a factor graph representing this distribution.

In the following, directed and undirected graphical models are discussed. Naïve Bayes and Hidden Markov Models fall into the first group, the Maximum Entropy Model falls into the second group of graphical models.

[9] A graph is called dependency graph if the independence of two variables implicates separability of the corresponding nodes in the graph (Beierle and Kern-Isberner, 2003, pp. 337f.).

[10] A factor graph consists of two sets of nodes: variable and factor nodes. There are no edges between nodes of the same set, so a factor graph is always bipartite.



(a) Independency graph (b) Factor graph

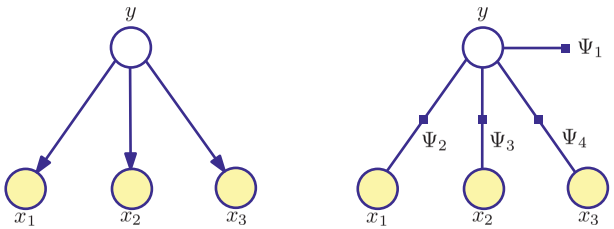
Figure 2: Directed graphical model

3.1 Directed Graphical Models

A joint distribution $p(\vec{v})$ can be factorized into the product of conditional distributions for each node v_k, so that each such conditional distribution is conditioned on its set of parent nodes v_k^p:

$$p(\vec{v}) = \prod_{k=1}^{K} p(v_k|v_k^p) \tag{31}$$

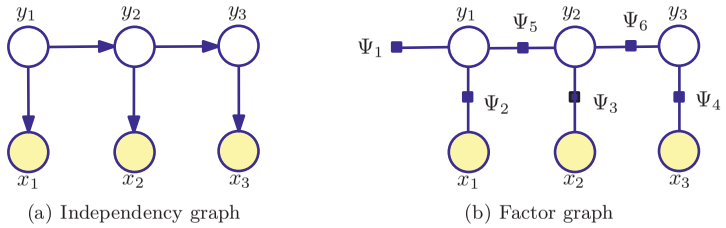
This is the same kind of factorization as shown in Figure 2 for the example distribution $p(x_1, x_2, y)$. As another example, take the Naïve Bayes classifier which is discussed in Section 2.1. Figure 3 graphically represents such a model with three observation variables. The corresponding probability distribution factorizes as $p(y, x_1, x_2, x_3) = p(y) \cdot p(x_1|y) \cdot p(x_2|y) \cdot p(x_3|y)$, following the Naïve Bayes assumption. Analogously, Figure 4 shows an HMM classifier for a sequence of three input variables x_1, x_2, x_3. The factorization is $p(x_1, x_2, x_3, y_1, y_2, y_3) = \Psi_1(y_1) \cdot \Psi_2(x_1, y_1) \cdot \Psi_3(x_2, y_2) \cdot \Psi_4(x_3, y_3) \cdot \Psi_5(y_1, y_2) \cdot \Psi_6(y_2, y_3)$ which corresponds[11] to the HMM (see equation 6).



(a) Independency graph (b) Factor graph

Figure 3: Naïve Bayes classifier

[11] A dedicated start value $y_0 = \perp$ is assumed, so that $\Psi(y_1) = p(y_0 = \perp, y_1))$.



(a) Independency graph (b) Factor graph

Figure 4: Independency and factor graph for the Hidden Markov Model.

3.2 Undirected Graphical Models

A probability distribution can be represented by an undirected graphical model using a product of non-negative functions of the maximal cliques of G. The factorization is performed in a way that conditionally independent nodes do not appear within the same factor, that means that they belong to different *cliques*:

$$p(\vec{v}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(\vec{v}_C)\,. \tag{32}$$

The factors $\Psi_C \geq 0$ are so-called *potential functions* of the random variables $\vec{v}_C$ within a clique $C \in \mathcal{C}$.

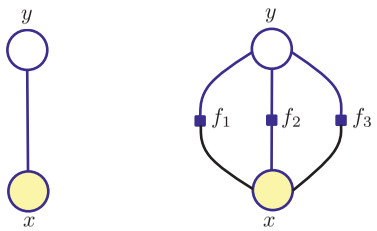
The potential functions may be any arbitrary function. Due to this generality the potential functions do not necessarily have to be probability functions. This is in contrast to directed graphs where the joint distribution is factorized into a product of conditional distributions. Thus, normalization of the product of potential functions is necessary to achieve a proper probability measure. This is yielded by a normalization factor Z. Calculating Z is one of the main challenges during parameter learning as summing over all possible variables is necessary:

$$Z = \sum_{\vec{v}} \prod_{C \in \mathcal{C}} \Psi_C(\vec{v}_C)\,. \tag{33}$$

In Section 2.3 the Maximum Entropy model was discussed which can be formulated by such a product of non-negative potential functions (compare to equation 28)

$$p_{\vec{\lambda}}(y|x) = \frac{1}{Z_{\vec{\lambda}}(x)} \prod_{i=1}^{m} \exp\left(\lambda_i f_i(x, y)\right)\,. \tag{34}$$

In such log-linear models, potential functions are formulated as the exponential function of weighted features. Such a formulation is frequently used because it fulfills the requirement of strict positivity of the potential functions. Figure 5(a) shows the independency graph for a Maximum Entropy classifier with an observation variable x, a corresponding factor graph with three features is shown in Figure 5(b).



(a) Independency graph (b) Factor graph

Figure 5: Maximum Entropy Classifier

Directed and undirected graphical models differ in the way the original probability distribution is factorized. The factorization into a product of conditional probability distributions as done in a directed graphical model is straight-forward. In undirected graphical models a factorization into arbitrary functions is done. This does not require an explicit specification how the variables are related. But it comes at the expense of having to calculate the normalization factor.

4 Conditional Random Fields

In the previous section, some well-known probabilistic models were discussed from a mathematical perspective. Moreover, the graphical representation, which characterizes the underlying probability distribution of the model, was shown.

A Hidden Markov Model can be understood as the sequence version of a Naïve Bayes Model: instead of single independent decisions, a Hidden Markov Model models a linear sequence of decisions. Accordingly, Conditional Random Fields can be understood as the sequence version of Maximum Entropy Models, that means they are also discriminative models. Furthermore, in contrast to Hidden Markov Models, Conditional Random Fields are not tied to the linear-sequence structure but can be arbitrarily structured.

In the following, the idea and theoretical foundation of Conditional Random Fields is illustrated. First, a general formulation of Conditional Random Fields is given followed by an in-depth discussion of the most popular form of CRFs, those with a linear sequence structure. A main focus are aspects of training and inference. This section closes with a short discussion of arbitrarily structured CRFs.

4.1 Basic Principles

Introduced by Lafferty et al. (2001), *Conditional Random Fields* (CRF) are probabilistic models for computing the probability $p(\vec{y}|\vec{x})$ of a possible output $\vec{y} = (y_1, \ldots, y_n) \in \mathcal{Y}^n$ given the input $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$ which is also called the *observation*. A CRF in

general can be derived from formula 32:

$$p(\vec{v}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(\vec{v}_C)\,. \tag{35}$$

The conditional probability $p(\vec{y}|\vec{x})$ can be written as

$$\begin{aligned} p(\vec{y}|\vec{x}) &= \frac{p(\vec{x},\vec{y})}{p(\vec{x})} \\ &= \frac{p(\vec{x},\vec{y})}{\sum_{\vec{y}'} p(\vec{y}',\vec{x})} \\ &= \frac{\frac{1}{Z}\prod_{C\in\mathcal{C}} \Psi_C(\vec{x}_C,\vec{y}_C)}{\frac{1}{Z}\sum_{\vec{y}'}\prod_{C\in\mathcal{C}} \Psi_C(\vec{x}_C,\vec{y}'_C)}\,. \end{aligned} \tag{36}$$

From this, the general model formulation of CRFs is derived:

$$\boxed{p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{C \in \mathcal{C}} \Psi_C(\vec{x}_C, \vec{y}_C)\,.} \tag{37}$$

As described in Section 3, Ψ_C are the different factors corresponding to maximal cliques in the independency graph (see Kschischang et al. (2001)). See Figure 6 for an example of a linear-chain CRF. Each factor corresponds to a potential function which combines different features f_i of the considered part of the observation and the output. The normalization follows from the denominator of equation 36:

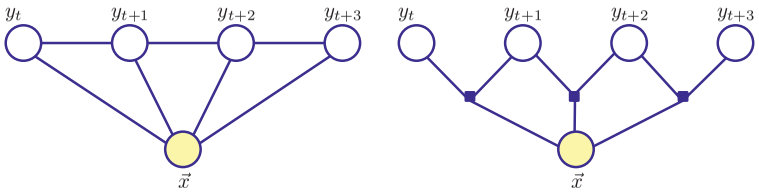
$$Z(\vec{x}) = \sum_{\vec{y}'} \prod_{C \in \mathcal{C}} \Psi_C(\vec{x}_C, \vec{y}')\,. \tag{38}$$

In fact, during both training and inference, for each instance a separate graph is used which is built from so-called *clique templates*. Clique templates make assumptions on the structure of the underlying data by defining the composition of the cliques. Each clique is a set of putatively interdependent variables, namely those contained in the corresponding potential function (Sutton and McCallum, 2007). Examples for clique templates are given in Section 4.2 and 4.3.

4.2 Linear-chain CRFs

A special form of a CRF, which is structured as a linear chain, models the output variables as a sequence. Figure 6 shows the respective independency and factor graphs. The CRF introduced in equation 37 can be formulated as

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y})\,, \tag{39}$$



(a) Independency graph (b) Factor graph

Figure 6: A Linear Chain Conditional Random Field

with

$$Z(\vec{x}) = \sum_{\vec{y}'} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y}')\,. \tag{40}$$

Given the factors $\Psi_j(\vec{x}, \vec{y})$ in the form

$$\Psi_j(\vec{x}, \vec{y}) = \exp\left(\sum_{i=1}^{m} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)\,, \tag{41}$$

and assuming $n+1$ to be the length of the observation sequence[12], a linear-chain CRF can be written as

$$p_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \cdot \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)\,. \tag{42}$$

The index j is needed in comparison to the Maximum Entropy Model because a label sequence is considered instead of a single label to be predicted. In equation 42, j specifies the position in the input sequence $\vec{x}$. Note that the weights λ_i are not dependent on the position j. This technique, known as parameter tying, is applied to ensure a specified set of variables to have the same value.

The normalization to $[0, 1]$ is given by

$$Z_{\vec{\lambda}}(\vec{x}) = \sum_{\vec{y} \in \mathcal{Y}} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)\,. \tag{43}$$

Summation over $\mathcal{Y}$, the set of all possible label sequences, is performed to get a feasible probability.

[12] Note, that the number of factors is n because any two consecutive positions y_{j-1} and y_j are combined in a factor.

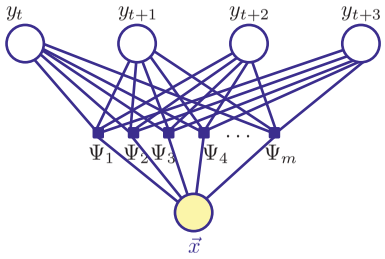


Figure 7: Alternative interpretation of a linear-chain CRF

In equation 42 a formulation of a linear-chain CRF is given. Moving the sum over the sequence positions in front of the exponential function, the actual factorization typically done for a CRF gets more evident:

$$p_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \cdot \prod_{j=1}^{n} \exp\left(\sum_{i=1}^{m} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)\,. \tag{44}$$

The factor graph in Figure 6(b) corresponds to this factorization. One could also move the sum over the different features in front of the exponential function

$$p_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \cdot \prod_{i=1}^{m} \exp\left(\sum_{j=1}^{n} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)\,. \tag{45}$$

In this interpretation, the factors are not "running" over the sequence but over the features. The factor graph with factors $\Psi_i = \exp\left(\sum_{j=1}^{n} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)$ corresponding to features f_i is given in Figure 7. This interpretation is less intuitive but shows the relation to the Maximum Entropy model (in Figure 5).

The model can be interpreted with even more factors by moving both sums to the front of the exponential function

$$p_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} \exp\left(\lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right)\,. \tag{46}$$

The corresponding factor graph is not shown here because of the large number of factors in the graph.

The factorization based on maximal cliques (see equation 44) is the one usually applied for a linear-chain CRF. The other two factorizations (see equations 45 and 46) do not adhere to this maximality. In general, factorizing according to cliques consisting of less variable nodes than the maximal clique lead to inaccuracies because not all dependencies are correctly considered. In this case, however, it leads to redundant computations

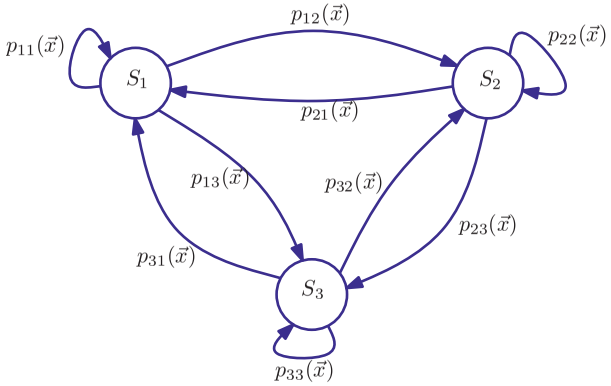


Figure 8: Example for a stochastic finite state automaton

as can be seen in equation 46. The rest of the paper is based on the idea of the first factorization.

Linear chain CRFs have exactly one clique template $C \in \mathcal{C}$: It specifies the independency graph to consist of connections between y_j and y_{j-1} and $\vec{x}$: $C = \{\Psi_j(y_j, y_{j-1}, \vec{x}) \mid \forall j \in \{1, \ldots, n\}\}$. Because of that special structure, it is possible to represent a linear-chain CRF by a stochastic finite state automaton (SFSA) similar to Hidden Markov Models. This is beneficial for implementation purposes. In that automaton the transition probabilities depend on the input sequence $\vec{x}$. Its structure is in general arbitrary but the most straight-forward approach is to use a fully connected automaton with states S_l where $l \in \mathcal{Y}$. One state is used for every symbol in the label alphabet. Such automaton with $|\mathcal{Y}| = 3$ is depicted in Figure 8.

As stated in equation 42, the features are dependent on the label sequence and herewith on the state transitions in the finite state automaton. So it is important to point out that only a subset of all features f_i is used in every transition in the graph.

The strategy to build a linear-chain CRF can now be summarized as follows:

1. Construct an SFSA $\mathcal{S} = (S, T)$ out of the set of states S (with transitions $T = (s, \dot{s}) \in S^2$). It can be fully connected but it is also possible to forbid some transitions.[13]

2. Specify a set of feature templates[14] $F = \{g_1(\vec{x}, j), \ldots, g_h(\vec{x}, j)\}$ on the input sequence. These are not used directly but for the generation of the features f_i.

3. Generate set of features $\mathcal{F} = \{\forall s, \dot{s} \in S.\ \forall g_o \in F : f_k(s, \dot{s}, g_o)\}$

[13] Such a decision might depend on the training data and the transitions contained there.

[14] An example for such a feature template is $g_1(\vec{x}, j) = \begin{cases} 1 & \text{if } x_j = \mathtt{V} \\ 0 & \text{else} \end{cases}$.

Until now, only first-order linear-chain CRFs have been considered. To define linear-chain CRFs with a higher order (see McDonald and Pereira, 2005), the features need to have the form

$$f_i(\vec{y}, \vec{x}, j) = f_i(h_j(\vec{y}), \vec{x}, j) \tag{47}$$

with

$$h_j(\vec{y}) = (y_{j-k+1}, \ldots, y_j)\,. \tag{48}$$

The order is given by k. For higher orders[15] ($k > 2$), the same probabilistic state automaton is used by combining different previous output values y_i in special states. For example, for $k = 3$ the set of states would be $S' = \{(S_i, S_j)\}$ for all $i, j \in \{1, \ldots, |S|\}$ (according to the first-order SFSA in Figure 8).

For that special linear-chain structure of the CRF, training and inference are formulated in a similar way as for Hidden Markov Models as basic problems (Rabiner, 1989):

I) Given observation $\vec{x}$ and a CRF $\mathcal{M}$: How to find the most probably fitting label sequence $\vec{y}$?

II) Given label sequences $\mathcal{Y}$ and observation sequences $\mathcal{X}$: How to find parameters of a CRF $\mathcal{M}$ to maximize $p(\vec{y}|\vec{x}, \mathcal{M})$?

Problem I is the most common application of a conditional random field, to find a label sequence for an observation. Problem II is the question how to train, to adjust the parameters of $\mathcal{M}$ which are especially the feature weights λ_i.

In discriminative approaches, the probability $p(\vec{x}|\mathcal{M})$ is not modeled. Estimating this is another basic question in context of Hidden Markov Models and not considered here.

4.2.1 Training

For all types of CRFs, as well as for Maximum Entropy Models, the maximum-likelihood method can be applied for parameter estimation. That means, that training the model is done by maximizing the log-likelihood $\mathcal{L}$ on the training data $\mathcal{T}$:

$$\begin{aligned} \bar{\mathcal{L}}(\mathcal{T}) &= \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \log p(\vec{y}|\vec{x}) \\ &= \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \left[\log \left(\frac{\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\left(y_{j-1}, y_j, \vec{x}, j\right)\right)}{\sum_{\vec{y}'\in\mathcal{Y}} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\left(y'_{j-1}, y'_j, \vec{x}, j\right)\right)} \right) \right]. \end{aligned} \tag{49}$$

To avoid overfitting the likelihood is penalized with the term $-\sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma^2}$. This technique is established for use in Maximum Entropy Models and can also be applied here (see

[15] Note that first order means $k = 2$.

explanations by Chen and Rosenfeld, 2000). The parameter σ^2 models the trade-off between fitting exactly the observed feature frequencies and the squared norm of the weight vector (McDonald and Pereira, 2005). The smaller the values are, the smaller the weights are forced to be, so that the chance that few high weights dominate is reduced.

For the derivation, the notation of the likelihood function $\mathcal{L}(\mathcal{T})$ is reorganized:

$$\begin{aligned}
\mathcal{L}(\mathcal{T}) &= \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \left[\log \left(\frac{\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y_{j-1},y_j,\vec{x},j\right)\right)}{\sum_{\vec{y}'\in\mathcal{Y}}\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1},y'_j,\vec{x},j\right)\right)} \right) \right] - \sum_{i=1}^{m}\frac{\lambda_i^2}{2\sigma^2} \\
&= \sum_{(\vec{x},\vec{y})\in\mathcal{T}} \left[\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y_{j-1},y_j,\vec{x},j\right) \right) - \right. \\
&\qquad \left. - \log\left(\sum_{\vec{y}'\in\mathcal{Y}} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1},y'_j,\vec{x},j\right)\right)\right)\right] - \sum_{i=1}^{m}\frac{\lambda_i^2}{2\sigma^2} \\
&= \underbrace{\sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y_{j-1},y_j,\vec{x},j\right)}_{\mathcal{A}} - \\
&\qquad \underbrace{\sum_{(\vec{x},\vec{y})\in\mathcal{T}} \log\underbrace{\left(\sum_{\vec{y}'\in\mathcal{Y}} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1},y'_j,\vec{x},j\right)\right)\right)}_{Z_{\vec{\lambda}}(\vec{x})}}_{\mathcal{B}} \underbrace{-\sum_{i=1}^{m}\frac{\lambda_i^2}{2\sigma^2}}_{\mathcal{C}} .
\end{aligned} \tag{50}$$

The partial derivations of $\mathcal{L}(\mathcal{T})$ by the weights λ_k are computed separately for the parts $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$. The derivation for part $\mathcal{A}$ is given by

$$\frac{\partial}{\partial\lambda_k}\sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y_{j-1},y_j,\vec{x},j\right) = \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{j=1}^{n} f_k\left(y_{j-1},y_j,\vec{x},j\right). \tag{51}$$

The derivation for part $\mathcal{B}$, which corresponds to the normalization, is given by

$$\begin{aligned}
\frac{\partial}{\partial\lambda_k}\sum_{(\vec{x},\vec{y})\in\mathcal{T}}\log Z_{\vec{\lambda}}(\vec{x}) &= \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\frac{1}{Z_{\vec{\lambda}}(\vec{x})}\frac{\partial Z_{\vec{\lambda}}(\vec{x})}{\partial\lambda_k} \\
&= \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\frac{1}{Z_{\vec{\lambda}}(\vec{x})}\sum_{\vec{y}'\in\mathcal{Y}}\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1},y'_j,\vec{x},j\right)\right)\cdot \\
&\qquad \cdot\sum_{j=1}^{n} f_k(y'_{j-1},y'_j,\vec{x},j) \\
&= \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{\vec{y}'\in\mathcal{Y}}\underbrace{\frac{1}{Z_{\vec{\lambda}}(\vec{x})}\exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m}\lambda_i f_i\left(y'_{j-1},y'_j,\vec{x},j\right)\right)}_{=p_{\vec{\lambda}}(\vec{y}|\vec{x}) \quad \text{see equation (42)}}\cdot \\
&\qquad \cdot\sum_{j=1}^{n} f_k(y'_{j-1},y'_j,\vec{x},j) \\
&= \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{\vec{y}\in\mathcal{Y}} p_{\vec{\lambda}}(\vec{y}'|\vec{x})\sum_{j=1}^{n} f_k(y'_{j-1},y'_j,\vec{x},j)
\end{aligned} \tag{52}$$

Part $\mathcal{C}$, the derivation of the penalty term, is given by

$$\frac{\partial}{\partial\lambda_k}\left(-\sum_{i=1}^{m}\frac{\lambda_i^2}{2\sigma^2}\right) = -\frac{2\lambda_k}{2\sigma^2} = -\frac{\lambda_k}{\sigma^2}. \tag{53}$$

The log-likelihood function in equation 50 is concave: The first term is linear (see equation 51) the second term belongs to the normalization. Hence, it does not change the concavity of the function and the last term is concave (see equation 53), so is the whole function.

Equation 51, the derivation of part $\mathcal{A}$, is the expected value under the empirical distribution of a feature f_i:

$$\tilde{E}(f_i) = \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{j=1}^{n} f_i\left(y_{j-1},y_j,\vec{x},j\right). \tag{54}$$

Accordingly, equation 52, the derivation of part $\mathcal{B}$, is the expectation under the model distribution:

$$E(f_i) = \sum_{(\vec{x},\vec{y})\in\mathcal{T}}\sum_{\vec{y}'\in\mathcal{Y}} p_{\vec{\lambda}}(\vec{y}'|\vec{x})\sum_{j=1}^{n} f_i(y'_{j-1},y'_j,\vec{x},j). \tag{55}$$

The partial derivations of $\mathcal{L}(\mathcal{T})$ can also be interpreted as

$$\frac{\partial \mathcal{L}(\mathcal{T})}{\partial \lambda_k} = \tilde{E}(f_k) - E(f_k) - \frac{\lambda_k}{\sigma^2}\,. \tag{56}$$

Note the relation of equations 54 and 55 to equations 12 and 16 which were formulated for the Maximum Entropy model. Besides the fact that for the CRF several output variables are considered, these equations correspond. A difference is the absence of the factor $\frac{1}{N}$, which is irrelevant for finding the maximum by the approximation of the first derivation $\tilde{E}(f_k) - E(f_k) - \frac{\lambda_k}{\sigma^2} = 0$.

Computing $\tilde{E}(f_i)$ is easily done by counting how often each feature occurs in the training data (McDonald and Pereira, 2005). Computing $E(f_i)$ directly is impractical because of the high number of possible tag sequences ($|\mathcal{Y}|$). Recall, that for the Maximum Entropy models, $E(f_i)$ can be computed efficiently due to the small number of different output variables y in most applications. In a CRF, sequences of output variables lead to enormous combinatorial complexity. Thus, a dynamic programming approach is applied, known as the Forward-Backward Algorithm originally described for Hidden Markov Models (Rabiner, 1989). This algorithm can also be used for linear-chain Conditional Random Fields in a slightly modified form.

According to McDonald and Pereira (2005), a function $T_j(s)$ is defined, which maps a single state s at an input position j to a set of allowed next states at position $j+1$, and the inverse function $T_j^{-1}(s)$, which maps the set of all states of possible predecessors to s. Special states $\perp$ and $\top$ are defined for start and end of the sequence. An example for the states in Figure 8 is $T_j(S_1) = \{S_1, S_2, S_3\}$. Forward ($\alpha$) and backward ($\beta$) scores will be used, which can be understood in general as messages sent over the network, in the following assumed to be a linear chain (Bishop, 2006):

$$\alpha_j(s|\vec{x}) = \sum_{s' \in T_j^{-1}(s)} \alpha_{j-1}(s'|\vec{x}) \cdot \Psi_j(\vec{x}, s', s) \tag{57}$$

$$\beta_j(s|\vec{x}) = \sum_{s' \in T_j(s)} \beta_{j+1}(s'|\vec{x}) \cdot \Psi_j(\vec{x}, s, s')\,. \tag{58}$$

In relation to the definition of the potentials in equation 41 the features are defined on special states: $\Psi_j(\vec{x}, s, s') = \exp\left(\sum_{i=1}^{m} \lambda_i f_i(y_{j-1} = s, y_j = s', \vec{x}, j)\right)$.

The α functions are messages sent from the beginning of the chain to the end. The β functions are messages sent from the end of the chain to the beginning. They are initialized by

$$\alpha_0(\perp|\vec{x}) = 1 \tag{59}$$

$$\beta_{|\vec{x}|+1}(\top|\vec{x}) = 1\,. \tag{60}$$

With these messages, it is possible to compute the expectation under the model distribution efficiently (Lafferty et al., 2001; McDonald and Pereira, 2005) by

$$E(f_i) = \sum_{(\vec{x},\vec{y}) \in \mathcal{T}} \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \underline{\sum_{j=1}^{n} \sum_{s \in S} \sum_{s' \in T_j(s)}} f_i(s, s', \vec{x}, j) \cdot \underline{\alpha_j(s|\vec{x}) \Psi_j(\vec{x}, s, s') \beta_{j+1}(s'|\vec{x})}\,. \tag{61}$$

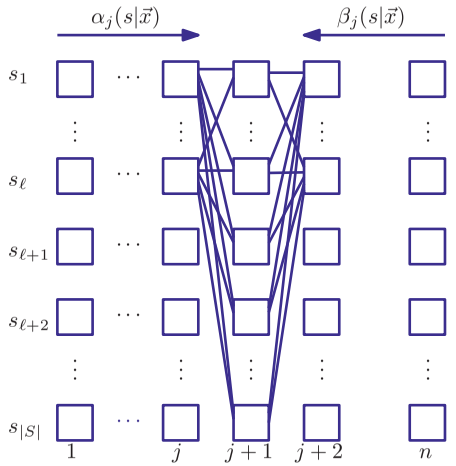


Figure 9: Message passing in the Forward-Backward Algorithm. Each column represents one variable, each box in a row one possible value of that variable.

The underlined part of formula 61 can be understood as computing the potentials in all combinations of state sequences in the training data. A nice visualization is the so-called *lattice diagram* in Figure 9 (Bishop, 2006) in which possible paths of messages sent are depicted. The values for α and β are stored after one iteration so that they have to be computed only once.

The normalization factor is computed by

$$Z_{\vec{\lambda}}(\vec{x}) = \beta_0(\perp|\vec{x})\,. \tag{62}$$

The Forward-Backward Algorithm has a run-time of $O(|S|^2 n)$, so it is linear in the length of the sequence and quadratic in the number of states.

4.2.2 Inference

The problem of inference is to find the most likely sequence $\vec{y}$ for given observations $\vec{x}$. Note, that this is not about to choose a sequence of states, which are individually most likely. That would be the maximization of the number of correct states in the sequence. In contrast, for finding the most likely sequence the Viterbi Algorithm is applied (Rabiner, 1989). The Viterbi Algorithm is similar to the Forward-Backward Algorithm. The main difference is, that instead of summing, a maximization is applied.

The quantity $\delta_j(s|\vec{x})$, which is the highest score along a path, at position j, which ends in state s, is defined as

$$\delta_j(s|\vec{x}) = \max_{y_1, y_2, \ldots, y_{j-1}} p(y_1, y_2, \ldots, y_j = s|\vec{x})\,. \tag{63}$$

The induction step is

$$\delta_{j+1}(s|\vec{x}) = \max_{s' \in S} \delta_j(s') \cdot \Psi_{j+1}(\vec{x}, s, s') \,. \tag{64}$$

The array $\psi_j(s)$ keeps track of the j and s values. The algorithm then works as follows:

1. Initialization:
   The values for all steps from the start state $\perp$ to all possible first states s are set to the corresponding factor value.

$$\begin{aligned} \forall s \in S: \quad & \delta_1(s) = \Psi_1(\vec{x}, \perp, s) \\ & \psi_1(s) = \perp \end{aligned} \tag{65}$$

2. Recursion:
   The values for the next steps are computed from the current value and the maximum values regarding all possible succeeding states s'.

$$\begin{aligned} \forall s \in S: \ 1 \leq j \leq n: \quad & \delta_j(s) = \max_{s' \in S} \delta_{j-1}(s') \Psi(\vec{x}, s', s) \\ & \psi_j(s) = \operatorname*{argmax}_{s' \in S} \delta_{j-1}(s') \Psi(\vec{x}, s', s) \end{aligned} \tag{66}$$

3. Termination:

$$p^* = \max_{s' \in S} \delta_n(s') \tag{67}$$

$$\vec{y}_n^* = \operatorname*{argmax}_{s' \in S} \delta_n(s') \tag{68}$$

4. Path (state sequence) backtracking:
   Recompute the optimal path from the lattice using the track keeping values ψ_t.

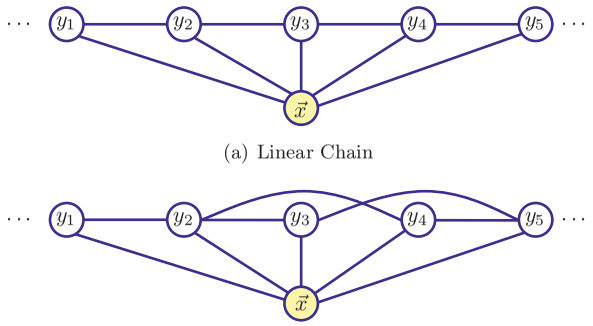
$$\vec{y}_t^* = \psi_{t+1}(\vec{y}_{t+1}^*) \qquad t = n-1, n-2, \ldots, 1 \tag{69}$$

Steps 1-3 are very similar to the Forward-Backward Algorithm. A lattice is filled with the "best" values. Step 4 reads the best path from that lattice.

4.3 Arbitrarily structured CRFs

In Section 4.2, efficient training and inference for the special case of a linear-chain CRF have been discussed. In the following, CRFs with an arbitrary graphical structure, such as a tree or a grid structure, are explained. Different CRF structures have been proposed by Sutton and McCallum (2007), Finkel et al. (2005), Lafferty et al. (2001), Sutton and McCallum (2004), and Tsai et al. (2006).

Moving from a linear-chain CRF to a general CRF basically means to abolish the restrictions that the clique templates (see Section 4.1) model a linear structure. Hence, more general algorithms for training and inference have to be applied.



(a) Linear Chain

(b) Skip Chain

Figure 10: Examples for Structures of Conditional Random Fields

4.3.1 Unrolling the graph

In some publications, different CRF structures are depicted (see citations above and examples shown in Figure 10). It has to be emphasized, that these structures are meant exemplarily because the actual graph is instantiated separately for each instance for training or inference with the help of the clique templates. Hence, the actual graph structure depends on the considered instance and the specific type of the CRF. The potential functions Ψ_j in the model (compare with equation 39) are associated with the clique templates, but *not* to the cliques in the graph. The process of building the graph for a specific instance is called "unrolling the graph".

As already discussed in Section 4.2 the set of clique templates $\mathcal{C}$ for a linear-chain CRF is given by

$$\mathcal{C} = \{C\} \text{ with } C = \left\{\Psi_j(y_j, y_{j-1}, \vec{x}) \mid \forall j \in \{1, \ldots, n\}\right\}. \tag{70}$$

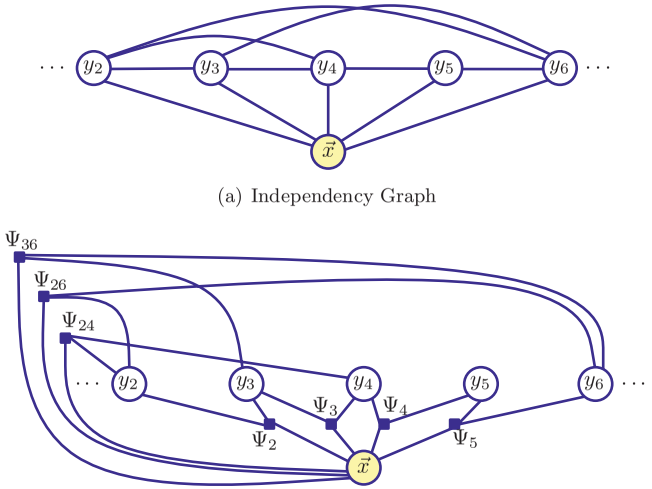
Accordingly, for a linear-chain CRF there is only one clique template resulting in a linear structure for every possible input value. Only because of this linear sequence structure, an SFSA can be used as a basis for an efficient implementation.

Another possible set of clique templates is

$$\mathcal{C} = \{C_1, C_2\} \text{ with } C_1 = \left\{\Psi_j(y_j, y_{j-1}, \vec{x}) \mid \forall j \in \{1, \ldots, n\}\right\} \tag{71}$$

$$\text{and } C_2 = \left\{\Psi_{ab}(y_a, y_b, \vec{x}) \mid (a, b) \in \{1, \ldots, n\}^2\right\}, \tag{72}$$

where a and b are indexes that specify labels with special attributes on the input sequence. For example in a sequence $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{N}^n$ the indexes a and b could specify all items where b is divisible by a. Given a concrete sequence $2, 3, 4, 5, 6$ this leads to a CRF with the structure shown in Figure 11. In real life applications parameter tying is often applied, in this example, the value of the weights λ_j is the same for identical clique templates, independent of the position in the sequence.



(a) Independency Graph

(b) Factor Graph

Figure 11: Example for an unrolled skip-chain CRF for the sequence $\vec{x} = (2, 3, 4, 5, 6)$ according to equations 71 and 72.

4.3.2 Training and Inference

In sequence structures such as the HMM or the linear-chain CRF (which is a simple chain globally conditioned on the input values $\vec{x}$) the Forward-Backward and the Viterbi Algorithm, which are based on sending messages along the chain in the only two possible directions, can be applied. Besides conceptually different algorithms, such as sampling methods, there are generalizations of these algorithms for tree-structured graphs, namely the sum-product and the max-sum algorithm (Kschischang et al., 2001).

The basic idea is that the messages sent along the graph are collected from different directions before forwarding them. This generalization can also be used on arbitrary graphs. The basic idea is to compute a so-called junction tree from the original graph. The algorithms can then be applied in a slightly modified form.

5 Summary

In this tutorial, a detailed overview of Conditional Random Fields and the theory behind and related to it is given. We started with a short recapitulation of well-known probabilistic models. Conditional Random Fields can be considered as an extension to the Maximum Entropy model (a structured learning model instead of a single-position classification model) on the one hand, and the Hidden Markov Model (discriminative model instead of a generative model) on the other hand.

Probabilistic models can be represented graphically by means of independency and factor graphs. We distinguished between directed and undirected graphical models which result in different types of factorization. Such factorizations can be read from the graphical representation of a model. They are often a crucial prerequisite for efficiently performing complex computations, as needed for training or inference. Aspects of how a CRF can be factorized and the resulting graphical representations are a focus of this paper.

Based on the concepts of graphical representation and factorization we have introduced a general formulation of CRFs. For the special case of a CRF based on a linear sequence structure, an in-depth explanation of methods for training and inference was given. These methods were originally developed for HMMs, but can be reused in a slightly modified way for linear-chain CRFs.

Finally, we shortly discussed the issues which arise for CRFs with an arbitrary graphical structure. This includes aspects of how the potential functions of the model can be defined by means of clique templates and why the training and inference algorithms used for a linear-chain CRF cannot be used in a non-sequential scenario. For further reading and detailed explanations on algorithms for training and inference on general probabilistic graphical models the interested reader might refer to Bishop (2006); Lepar and Shenoy (1999); Jordan and Weiss (2002); Mooij and Kappen (2007); Borgelt and Kruse (2002).

Acknowledgements

We would like to thank Christoph M. Friedrich, Juliane Fluck and Ernst Günter Schukat-Talamazzini for discussions and motivations. Special thanks to Günter Rudolph for fruitful suggestions.

Funding for the work of Roman Klinger was provided by the Fraunhofer-Max-Planck Cooperation "Learning and Inference Platform" (http://lip.fml.tuebingen.mpg.de/). The work of Katrin Tomanek was funded by the European Commission within the BOOTStrep project (FP6-028099).

References

[Androutsopoulos et al. 2000a] Androutsopoulos, Ion; Koutsias, John; Chandrinos, Konstantinos V.; Paliouras, George; Spyropoulos, Constantine D.: An evaluation of Naive Bayesian anti-spam filtering. In: Potamias, G.; Moustakis, V.; Someren, M. van (Editors.): *Proceedings of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning*. Barcelona, Spain, 2000, pp. 9–17. – URL http://arxiv.org/abs/cs/0006013v1.

[Androutsopoulos et al. 2000b] Androutsopoulos, Ion; Paliouras, Georgios; Karkaletsis, Vangelis; Sakkis, Georgios; Spyropoulos, Constantine D.; Stamatopoulos, Panagiotis: Learning to Filter Spam E-Mail: A Comparison of

a Naive Bayesian and a Memory-Based Approach. In: Zaragoza, H.; Gallinari, P.; Rajman, M. (Editors.): *Proceedings of the workshop "Machine Learning and Textual Information Access", 4th Conference on Principles and Practice of Knowledge Discovery in Databases*. Lyon, France, 2000, pp. 1–13. – URL http://arxiv.org/abs/cs/0009009v1.

[Beierle and Kern-Isberner 2003] Beierle, Christoph; Kern-Isberner, Gabriele: *Methoden wissensbasierter Systeme*. 2nd. Wiesbaden, Germany: Vieweg, May 2003. – in german.

[Berger et al. 1996] Berger, Adam L.; Pietra, Stephen D.; Pietra, Vincent J. D.: A Maximum Entropy Approach to Natural Language Processing. In: *Computational Linguistics* 22 (1996), No. 1, pp. 39–71.

[Bishop 2006] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, 2006. – ISBN 0-387-31073-8.

[Borgelt and Kruse 2002] Borgelt, Christian; Kruse, Rudolf: *Graphical Models: Methods for Data Analysis and Mining*. Wiley & Sons, 2002.

[Buyko et al. 2007] Buyko, Ekaterina; Tomanek, Katrin; Hahn, Udo: Resolution of Coordination Ellipses in Named Entities in Scientific Texts. In: *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING 2007)*. Melbourne, Australia, 2007, pp. 163–171.

[Chen and Rosenfeld 2000] Chen, Stanley F.; Rosenfeld, Ronald: A Survey of Smoothing Techniques for ME Models. In: *IEEE Transactions on Speech and Audio Processing* 8 (2000), No. 1, pp. 37–50.

[DeCaprio et al. 2007] DeCaprio, David; Vinson, Jade P.; Pearson, Matthew D.; Montgomery, Philip; Doherty, Matthew; Galagan, James E.: Conrad: Gene Prediction using Conditional Random Fields. In: *GENOME RESEARCH* 17 (2007), No. 9, pp. 1389–1396.

[Finkel et al. 2005] Finkel, Jenny R.; Grenager, Trond; Manning, Christopher: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005, pp. 363–370.

[Gupta et al. 2007] Gupta, Kapil K.; Nath, Baikunth; Ramamohanarao, Kotagiri: Conditional Random Fiels for Intrusion Detection. In: *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 203–208. – ISBN 0-7695-2847-3.

[Hand and Yu 2001] Hand, David J.; Yu, Keming: Idiot's Bayes – not so stupid after all? In: *International Statistical Review* 69 (2001), No. 3, pp. 385–399.

[He et al. 2004] He, Xuming; Zemel, Richard S.; Carreira-Perpinan, Migual A.: Multiscale conditional random fields for image labeling. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Vol. 2, 2004, pp. 695–702.

[Jaynes 1957] Jaynes, Edwin T.: Information Theory and Statistical Mechanics. In: *Physical Review* 106 (1957), May, No. 4, pp. 620–630.

[Jordan and Weiss 2002] Jordan, Michael I.; Weiss, Yair.: Chap. Graphical Models: Probabilistic Inference. In: Arbib, Michael A. (Editors.): *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 2002. – URL http://www.cs.berkeley.edu/~jordan/papers/jordan-weiss.ps.

[Kiritchenko and Matwin 2001] Kiritchenko, Svetlana; Matwin, Stan: Email classification with co-training. In: Stewart, Darlene A.; Johnson, J. H. (Editors.): *Proceedings of the conference of the Centre for Advances Studies on Collaborative research*. Toronto, Ontario, Canada, November 2001, pp. 192–201.

[Klinger et al. 2007a] Klinger, Roman; Friedrich, Christoph M.; Fluck, Juliane; Hofmann-Apitius, Martin: Named Entity Recognition with Combinations of Conditional Random Fields. In: *Proceedings of the Second BioCreative Challenge Evaluation Workshop*. Madrid, Spain, April 2007, pp. 89–91.

[Klinger et al. 2007b] Klinger, Roman; Furlong, Laura I.; Friedrich, Christoph M.; Mevissen, Heinz T.; Fluck, Juliane; Sanz, Ferran; Hofmann-Apitius, Martin: Identifying Gene Specific Variations in Biomedical Text. In: *Journal of Bioinformatics and Computational Biology* 5 (2007), December, No. 6, pp. 1277–1296.

[Korn and Korn 2000] Korn, Granino A.; Korn, Theresa M.: *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review*. 2 Revised. New York: Dover Publications Inc., 2000.

[Kschischang et al. 2001] Kschischang, Frank; Frey, Brendan J.; Loeliger, Hans-Andrea: Factor Graphs and the Sum-Product Algorithm. In: *IEEE Transactions on Information Theory* 47 (2001), No. 2, pp. 498–519.

[Lafferty et al. 2001] Lafferty, John D.; McCallum, Andrew; Pereira, Fernando C. N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, Morgan Kaufmann Publishers, 2001, pp. 282–289.

[Lau et al. 1993] Lau, Raymond; Rosenfeld, Ronald; Roukos, Salim: Adaptive language modeling using the maximum entropy principle. In: *HLT '93: Proceedings of the workshop on Human Language Technology*. Morristown, NJ, USA: Association for Computational Linguistics, 1993, pp. 108–113.

[Lepar and Shenoy 1999] Lepar, Vasilica; Shenoy, Prakash P.: A Comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions. In: Cooper, Gregory F.; Moral, Serafin (Editors.): *Uncertainty in Artificial Intelligence* Vol. 14. San Francisco, CA: Morgan Kaufmann, 1999, pp. 328–337.

[McDonald and Pereira 2005] McDonald, Ryan; Pereira, Fernando: Identifying gene and protein mentions in text using conditional random fields. In: *BMC Bioinformatics* 6 (Suppl 1) (2005), May, No. S6.

[McDonald et al. 2004] McDonald, Ryan T.; Winters, R. S.; Mandel, Mark; Jin, Yang; White, Peter S.; Pereira, Fernando: An entity tagger for recognizing acquired genomic variations in cancer literature. In: *Bioinformatics* 20 (2004), pp. 3249–3251.

[Mooij and Kappen 2007] Mooij, Joris M.; Kappen, Hilbert J.: Sufficient conditions for convergence of the Sum-Product Algorithm. In: *IEEE Transactions on Information Theory* 53 (2007), December, No. 21, pp. 4422–4437. – URL http://arxiv.org/abs/cs/0504030v2.

[Pearl 1988] Pearl, Judea: *Probabilistic Reasoning in Intelligent Systems – Networks of Plausible Inference*. Revised Second Printing. Morgan Kaufmann Publishers, Inc., 1988.

[Quattoni et al. 2005] Quattoni, Ariadna; Collins, Michael; Darrell, Trevor: Conditional Random Fields for Object Recognition. In: Saul, Lawrence K.; Weiss, Yair; Bottou, Leon (Editors.): *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005, pp. 1097–1104.

[Rabiner 1989] Rabiner, Lawrence R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proceedings of the IEEE* 77 (1989), No. 2, pp. 257–286.

[Russell and Norvig 2003] Russell, Stuart; Norvig, Peter: *Artificial Intelligence – A Modern Approach*. Prentice Hall, 2003.

[Settles 2004] Settles, Burr: Biomedical Named Entity Recognition using Conditional Random Fields and Rich Feature Sets. In: Collier, Nigel; Ruch, Patrick; Nazarenko, Adeline (Editors.): *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*. Geneva, Switzerland: COLING, 2004, pp. 107–110.

[Sha and Pereira 2003] Sha, Fei; Pereira, Fernando: Shallow parsing with Conditional Random Fields. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 134–141.

[Sutton and McCallum 2004] Sutton, Charles; McCallum, Andrew: Collective Segmentation and Labeling of Distant Entities in Information Extraction / Department of Computer Science, University of Massachusetts. 2004 (TR 04-49). – Technical Report.

[Sutton and McCallum 2007] Sutton, Charles; McCallum, Andrew: An Introduction to Conditional Random Fields for Relational Learning. In: Getoor, Lise; Taskar, Benjamin (Editors.): *Introduction to Statistical Relational Learning*. MIT Press, November 2007, Chap. 4, pp. 93–127.

[Tomanek et al. 2007] Tomanek, Katrin; Wermter, Joachim; Hahn, Udo: A Reappraisal of Sentence and Token Splitting for Life Science Documents. In: *Proceedings of the 12th World Congress on Medical Informatics (MEDINFO 2007)*. Brisbane, Australia, August 2007, pp. 524–528.

[Tsai et al. 2006] Tsai, Richard Tzong-Han; Sung, Cheng-Lung; Dai, Hong-Jie; Hung, Hsieh-Chuan; Sung, Ting-Yi; Hsu, Wen-Lian: NERBio: using selected word conjunctions, term normalization, and global patterns to improve biomedical named entity recognition. In: *BMC Bioinformatics* 7(Suppl 5) (2006), No. S11.

[Wallach 2004] Wallach, Hanna M.: Conditional Random Fields: An Introduction. / Department of Computer and Information Science, University of Pennsylvania. 2004 (MS-CIS-04-21). – Technical Report.

[Zhang et al. 2007] Zhang, Xinhua; Aberdeen, Douglas; Vishwanathan, S. V. N.: Conditional random fields for multi-agent reinforcement learning. In: *ICML '07: Proceedings of the 24th international conference on Machine learning*. New York, NY, USA: ACM, 2007, pp. 1143–1150. – ISBN 978-1-59593-793-3.