# TildeCRF: Conditional Random Fields for Logical Sequences

Bernd Gutmann and Kristian Kersting

University of Freiburg, Institute for Computer Science, Machine Learning Lab,
Georges-Koehler-Allee, Building 079, 79110 Freiburg, Germany
{bgutmann,kersting}@informatik.uni-freiburg.de

**Abstract.** Conditional Random Fields (CRFs) provide a powerful instrument for labeling sequences. So far, however, CRFs have only been considered for labeling sequences over flat alphabets. In this paper, we describe TildeCRF, the first method for training CRFs on logical sequences, i.e., sequences over an alphabet of logical atoms. TildeCRF's key idea is to use relational regression trees in Dietterich et al.'s gradient tree boosting approach. Thus, the CRF potential functions are represented as weighted sums of relational regression trees. Experiments show a significant improvement over established results achieved with hidden Markov models and Fisher kernels for logical sequences.

## 1 Introduction

Sequential data are ubiquitous and are of interest to many communities. Such data can be found in virtually all application areas of machine learning including computational biology, user modeling, speech recognition, empirical natural language processing, activity recognition, information extractions, etc. Therefore, it is not surprising that sequential data has been the subject of active research for decades. One of the many problems investigated concerns assigning labels to sequences of objects. For example, in protein secondary structure prediction, the task is to assign a secondary structure class to each amino acid residue in the protein sequence [13]. Dietterich *et al.* [4] have called this general class of problems *sequential supervised learning* (SSL), which can be formalized as follows.

**Definition 1 (Sequential Supervised Learning). Given** *a finite set of training examples of the form* $\{(X_i, Y_i)\}_{i=1}^m$*, where each* $X_i$ *is a sequence* $\langle x_{i,j}\rangle_{j=1}^{T_i} \in \bigotimes \mathcal{X}$ *of elements in the input space* $\mathcal{X}$ *and each* $Y_i$ *is the corresponding sequence* $\langle y_{i,j}\rangle_{j=1}^{T_i} \in \bigotimes \mathcal{Y}$ *of elements in the output space* $\mathcal{Y}$*,* ***find*** *a function* $H : \otimes\mathcal{X} \rightarrow \otimes\mathcal{Y}$ *with low approximation error on the training data as well as on unseen examples.*

One appealing approach to SSL are probabilistic sequence models as they take uncertainty into account explicitly. A probabilistic sequence model assumes the $X_i$'s and $Y_i$'s to be sampled from some random variables $X$ and $Y$ and attempt to learn the statistical dependency $P(X, Y)$ between them. Hidden Markov models (HMMs) [14] are among the most popular probabilistic sequence models. An

HMM models a sequence $X_i$ by assuming that there is an underlying sequence of states $Y_i$ drawn from a finite set of states $S$. To model the joint distribution $P(X, Y)$ tractably, HMMs make two independency assumptions: each state depends only on its immediate predecessor and each observation sequence $x_{i,j}$ depends only on the current state $y_{i,j}$ . Given this, it is relatively straightforward to estimate their parameters. Furthermore, HMMs are relatively easy to understand by humans. Despite of their success, HMMs have two major weaknesses:

**(A)** they are able to only handle sequences over flat alphabets, and
**(B)** it is cumbersome to model arbitrary dependencies in the input space.

To overcome **(A)**, *logical hidden Markov models* (LoHMMs) [6] have recently been introduced as an extension of HMMs. They allow for *logical sequences*, i.e., sequences of atoms in a first order logic. In [6], LoHMMs have been applied to the problem of discovering structural signatures of protein folds and led to more compact models. The trained LoHMM consisted of 120 parameters corresponding to an HMM with more than 62000 parameters. However, LoHMMs still suffer from limitation **(B)**, i.e., the difficulty to model arbitrary dependencies in the input space. One way to address this problem is to explicitly model these dependencies by using complex LoHMM structures. Selecting a structure of a LoHMM, however, is a significant problem [8]. Whereas HMMs are commonly learned by estimating the ML parameters of a fixed, fully connected model, this is not feasible for LoHMMs: different abstraction levels have to be explored.

To overcome **(B)**, i.e., to easily model arbitrary dependencies in the input space, conditional random fields [9] (CRFs) have become popular in language processing, computer vision, and information extraction. They have outperformed HMMs on language processing tasks such as information extraction and shallow parsing. CRFs are undirected graphical models that represent the conditional probability distribution $P(Y|X)$. Instead of the generatively trained (Lo)HMM, the discriminatively trained CRF is designed to handle non-independent input features, which can be beneficial in complex domains. For example, we would like to exploit other features of an amino acid, such as its molecular weight or its neighboring words.

Many sequences occurring in real-world problems such as in computational biology, planning, and user modeling, however, exhibit internal structure. The elements of such sequences can be seen as atoms in a relational logic (see e.g. [10] for an introduction to logic). For example, the secondary structure of the Ribosomal protein L4 can be represented as

$$\mathtt{st}(\mathtt{null}, 2), \mathtt{he}(\mathtt{h}(\mathtt{right}, \mathtt{alpha}), 6), \mathtt{st}(\mathtt{plus}, 2), \mathtt{he}(\mathtt{h}(\mathtt{right}, \mathtt{alpha}), 4), \ldots$$

Here, helices of a certain type and length `he`(*HelixType*,*Length*) and strands of a certain orientation and length `st`(*Orientation*,*Length*) are essentially structured symbols, i.e., atoms over logical predicates. The application of traditional CRFs to such sequences requires one to either ignore the structure of helices and strands, which results in a loss of information, or to take all possible combinations (of arguments such as orientation and length) into account, which leads to a combinatorial explosion in the number of parameters.

The main contribution of this paper is TildeCRF, the first method for training CRFs for logical sequences, i.e., sequences over an alphabet of logical atoms. The key idea of TildeCRF is to use relational regression trees in Dietterich *et al.*'s gradient tree boosting approach [4] to make relational abstraction through logical variables and unification. Thus, the TildeCRF potential functions are represented as weighted sums of relational regression trees. Experiments show a significant improvement over previous results achieved with hidden Markov models and Fisher kernels for logical sequences.

The outline of the paper is as follows. After discussing related work, we will briefly review CRFs in Section 3. In Section 4, we devise TildeCRFs for logical sequences. Before concluding, we experimentally evaluate TildeCRF in Section 5.

## 2 Related Work

CRFs for logical sequences combine two different research directions. On the one hand, they are related to several extensions of HMMs and CRFs. On the other hand, they are related to the recent interest in combining relational learning with probabilistic models such as Markov random fields [3].

In the first type of approaches, the underlying idea is to upgrade HMMs and CRFs to represent more structured state spaces. Sutton and McCallum's Factorial CRFs [17], Quattoni *et al.*'s and Dietterich *et al.*'s tree-shaped CRFs [12, 4], and Sutton *et al.*'s dynamic CRFs [18] decompose the state variables into smaller units. The key differences with TildeCRF is that these approaches do not consider learning the features and that they do not employ the logical concepts of variables and unification. Both are essential because variables allows one to group states together and unification allows one to share knowledge between abstract states via abstract transitions. LoHMMs [6] and relational Markov models [1] extend (H)MMs to handle sequences of logical atoms. Consequently, both suffer from the same difficulty to model arbitrary dependencies in the input space.

In the second type of approaches, most attention has been devoted to developing highly expressive formalisms. Taskar *et al.* 's relational Markov networks (RMN) [19] extend Markov random fields by providing a relational language for describing clique structures and enforcing parameter sharing at the template level. RMNs have been applied to computer vision and natural language problems. Domingos and Richardson [15] introduced Markov logic networks (MLNs). MLNs also upgrade Markov random fields to the relational case. In contrast to RMNs, MLNs view logical formulas as soft constraints on the set of possible worlds: if a world violates one formula, it is less probable but not necessarily impossible as in classical logic. This is essentially realized by representing potentials as weighted sets of logical formulas; the weights reflect how strong the constraints are. Both approaches are not specifically designed for analyzing logical sequences. Recently, Shanghei *et al.* [16] introduced dynamic probabilistic relational models. In contrast to TildeCRF, they extend directed dynamic models.

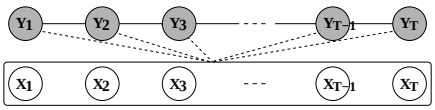



**Fig. 1.** Graphical representation of linear-chain CRF.

TildeCRF can be seen as an attempt towards downgrading such highly expressive frameworks for handling logical sequences.

## 3 Conditional Random Fields

In recent years, conditional random fields [9] (CRFs) turned out to be a suitable representation for SSL. CRFs are undirected graphical models that encode a conditional probability distribution using a given set of features. CRFs are defined as follows. Let $G$ be an undirected graphical model over sets of random variables $X$ and $Y$. As a special case, consider a *linear-chain* CRF, that is $X = \langle x_{i,j} \rangle_{j=1}^{T_i}$ and $Y = \langle Y_{i,j} \rangle_{j=1}^{T_i}$, so that $Y$ is a labeling of an observed sequence $X$. Then, CRFs define the conditional probability of a state sequence given the observed sequence as

$$P(Y|X) = Z(X)^{-1} \exp \sum\nolimits_{t=1}^{T} \Psi_t(y_t, X) + \Psi_{t-1,t}(y_{t-1}, y_t, X).$$

where $\Psi_t(y_t, X)$ and $\Psi_{t-1,t}(y_{t-1}, y_t, X)$ are potential functions and $Z(X)$ is a normalization factor over all state sequences $X$. A *potential function* is a real-valued function that captures the degree to which the assignment $y_t$ to the output variable fits the transition from $y_{t-1}$ and $X$. Due to the global normalization by $Z(X)$, each potential has an influence on the overall probability.

To apply CRFs to SSL problems, one must choose a representation for the potentials. Typically, it is assumed that the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that $\Psi(y_t, X) = \sum \alpha_k g_k(y_t, X)$ and $\Psi(y_{t-1}, y_t, X) = \sum \beta_k f_k(y_{t-1}, y_t, X)$ respectively. The model parameters are now a set of real-valued weights $\alpha_k, \beta_k$, one weight for each feature. In linear-chain CRF, a first-order Markov assumption is made on the hidden variables. A graphical model for this is shown in Figure 1. In this case, there are features for each label transition. Feature functions can be arbitrary such as a binary test that has value 1 if and only if $y_{t-1}$ has the label $a$.

## 4 TildeCRF: CRFs for Logical Sequences

Originally, Lafferty *et al.* introduced CRFs as an essentially propositional representation: symbols used to represent states and outputs are flat. So far, CRFs have not been considered for sequences of logical (ground) atoms. Here, we will describe how to lift CRFs to the relational case. More precisely, we consider the following variant of the SSL problem in Definition 1.

**Definition 2 (Relational-propositional SSL (RP-SSL)). Given** *a set of training examples* $(X_i, Y_i)$*, where each* $X_i$ *is a sequences of logical atoms and each* $Y_i$ *is a corresponding sequence of class labels* $y_{i,j} \in \{c_1, \ldots, c_n\}$*,* **find** *a classifier* $H$ *with low approximation error on the training data as well as on unseen examples.*

The idea underlying TildeCRF, i.e., a CRF for solving RP-SSL, is now to pick up the idea of MLNs and to represent potentials as sets of weighted logical formulas.

### 4.1 Relational Logic and Relational Potentials

Based on the representation of the Ribosomal protein $L4$ given in the introduction, we describe the necessary concepts of relational logic. The symbols `st`, `null`, `2`, `he`, `h`, ... are distinguished into predicate and function symbols. Associated with every symbol is the *arity*, i.e., number of arguments. In the example, `st/2` and `he/2` are predicates of arity 2, `h/2` is a function of arity 2, and `plus/0`, `1/0`, ... are functions of arity 0, i.e., constants. The *alphabet* $\Sigma$ consists of predicates, functions, and variables (e.g., `X`). A *term* is a variable or a function symbol followed by its arguments in brackets such as $\mathtt{h(right, X)}$ or `4`; an *atom* is a predicate symbol followed by its arguments in brackets such as $\mathtt{he(h(right, X), 4)}$. Valid arguments of functions and predicates are terms. A *ground term* or *atom* is one that does not contain any variables. In the protein example $\mathtt{st(null, 2)}$, $\mathtt{he(h(right, alpha), 6)}$, ... are ground atoms and `null`, `2`, $\mathtt{h(right, alpha)}$, `right`, `alpha`, ... are ground terms. A substitution $\sigma = \{\mathtt{X/plus}\}$ is an assignment of terms `plus` to variables `X`. Applying a substitution $\sigma$ to a term or an atom $e$ yields the instantiated term or atom $e\sigma$ where all occurrences of the variables `X` are simultaneously replaced by the term `plus`, e.g., $(\mathtt{st(X, 12), st(X, 10)})$ yields $\mathtt{st(plus, 12), st(plus, 10)}$. A substitution $\sigma$ is a *unifier* of a set of atoms $S$ if $S\sigma$ is singleton; if furthermore for every unifier $\sigma'$ of $S$ there is a substitution $\sigma''$ such that $\sigma = \sigma'\sigma''$ then $\sigma$ is the *most general unifier* (MGU) of $S$. A conjunction $A$ is $\theta$-subsumed by a conjunction $B$, denoted by $A \preceq_\theta B$, if there exists a substitution $\theta$ such that $B\theta \subseteq A$.

Relational abstraction within potentials offers a great compactness. Consider

$$0.938 \ : \ \mathtt{outPrevIs(city(c)), containsAt(1, a(X, f)), containsAt(4, a(X, a))}$$

taken from the regression tree shown in Figure 2. It groups all ground instances, where `X` is substituted by some term such as $\{\mathtt{X/1}\}, \{\mathtt{X/2}\}, \ldots$ Therefore, relational abstraction makes useful prediction possible in very large state spaces, where many of the states are never observed in the training data.

The compactness and even comprehensibility, however, comes at the expense of a more complex parameter estimation problem: they are non-parametric functional representations. Therefore, gradient-based optimization techniques such as McCallum's MALLET [11], which assume a parameterized representation, cannot be applied. Instead, we follow Dietterich *et al.*'s gradient tree boosting technique [4], called TreeCRF. In TreeCRF, the potential functions are represented by sums of traditional regression trees, which are grown stage-wise in

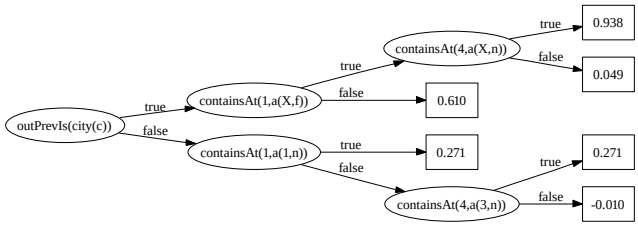



**Fig. 2.** A relational regression tree (taken from the *job scheduling* experiment of Section 5.2). An inner node represents a literal, a path constitutes a conjunction, and a leave represents the regression value (mean) of all examples sorted in this leave. As explained in Section 4.2, not the complete input $X$ but only windows $w_d(X)$ at time steps $d$ of fixed size $s$ are used. $\texttt{outPrevIs}(\texttt{Y})$ denotes the output $\texttt{Y}$ at time step $d-1$ and $\texttt{containsAt}(\texttt{P},\texttt{X})$ the input $\texttt{X}$ at position $\texttt{P}$ in the current window $w_d(X)$.

the manner. Each regression tree can be viewed as defining several new feature combinations one corresponding to each path in the tree from the root to a leaf. The resulting potential functions still have the form of a linear combination of features, but the features can be quite complex.

### 4.2 Model Selection via Functional Gradient Ascent

(Conditional) maximum likelihood parameter estimation is a common framework to determine the parameter $\Theta$ of a CRF. The likelihood of the training data given the current parameter $\Theta_{m-1}$ is used to improve the parameter. Normally, one uses some sort of gradient search for doing this. The parameter in the next iteration are the current plus the gradient of the likelihood function: $\Theta_m = \Theta_0 + \delta_1 + \ldots + \delta_m$ where $\delta_m = \eta_m \cdot \partial/\partial\Theta_{m-1} \sum_i \log P\left(y_i|x_i;\Theta_{m-1}\right)$ is the gradient multiplied by a constant $\eta_m$, which is obtained by doing a line search along the gradient. In our non-parametric case, the potential can be arbitrarily chosen. One starts with some initial potential $\Psi_0$, e.g. the zero function, and adds iteratively corrections $\Psi_m = \Psi_0 + \Delta_1 + \ldots + \Delta_m$, cf. TreeBoost in Alg. 1. In contrast to the standard gradient approach, $\Delta_i$ here denotes the so-called functional gradient, i.e., $\Delta_m = \eta_m \cdot E_{x,y}\left[\partial/\partial\Psi_{m-1} \log P(y|x;\Psi_{m-1})\right]$. Since the joint distribution $P(x,y)$ is unknown, one cannot evaluate the expectation $E_{x,y}$. Dietterich *et al.* suggested to evaluate the gradient function at every position in every training example and fit a regression tree to these derived examples, cf. GenExamples in Alg. 1. In our case, these regression trees are relational.

**Relational Regression Trees** Relational regression trees upgrade the attribute value representation used within classical regression trees: every test is a relational conjunction of atoms; a variable introduced in some node cannot appear in its right subtree, i.e., variables are bounded along left-tree paths. Consider the relational regression tree shown in Figure 2. The set of ground atoms $\{\texttt{outPrevIs}(\texttt{city}(\texttt{c})), \texttt{containsAt}(\texttt{1},\texttt{a}(\texttt{2},\texttt{f})), \texttt{containsAt}(\texttt{4},\texttt{a}(\texttt{2},\texttt{n}))\}$ is sorted

into the left most leaf, i.e., the value 0.938 is assigned. In contrast, changing the last atom to $\texttt{containsAt}(4, \texttt{a}(4, \texttt{n}))$ yields 0.049 as value.

Now, to induce a relational regression tree, we essentially employ Blockeel and De Raedt's TILDE [2], which also explains the name of our approach: Tilde-CRF. TILDE learns relational trees in the *learning from interpretations* setting, i.e., examples are sets of ground atoms. It basically follows Quinlan's well-known C4.5 algorithm. The only point where TILDE differs from C4.5 is in the computation of the tests to be placed in a node. To this aim, it employs a classical *refinement operator* under $\theta$-subsumption. The operator basically adds a literal, unify variables, and grounds variables. When a node is to be splitted, the set of all refinements are computed and evaluated. That is one starts with the empty tree and repeatedly searches for the best test for a node according to some splitting criterion. Next, the examples $D$ in the node are split into $D_1$ (success) and $D_2$ (failure) according to the test. For each split, the procedure is recursively applied, obtaining subtrees for the respective splits. As splitting criterion, we use the weighted variance on $D_1$ and $D_2$. The procedure stops if the variance in one node is small enough or the depth limit was reached. In leaves, the average regression value is predicted.

Using relational trees, Dietterich *et al.*'s TreeCRF can be adapted as follows.

**Relational Functional Gradients** Following Dietterich *et al.*'s notation, we define $F^{y_t}(y_{t-1}, X) = \Psi(y_t, X) + \Psi(y_{t-1}, y_t, X)$. Then, the gradient $\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))}$ can be evaluated quite easily as Corollary 1 (see below) shows. By evaluating the gradient at every known position in our training data and fitting a regression model to this values, we get an approximation of the expectation of the gradient. In order to simplify the derivation of the gradient and afterwards the evaluation, we do not use the complete input $X$ but a window $w_d(X) = x_{d-s}, \ldots, x_d, \ldots, x_{d+s}$, where $s$ is a fixed window size. This is exactly the learning setting of TILDE: each window, i.e., each regression example is a (weighted) set of ground atoms.

**Corolla 1** *The functional gradient with respect to $F^v(u, w_d(X))$ is*

$$\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} = I(y_{d-1} \subseteq_\Theta u, y_d \subseteq_\Theta v) - P(y_{d-1} \subseteq_\Theta u, y_d \subseteq_\Theta v | w_d(X))$$

*where $I$ is the identity function, $\subseteq_\Theta$ denotes that $u$ $\theta$-subsumes $y$, and $P(y_{d-1} \subseteq_\Theta u, y_d \subseteq_\Theta v | w_d(X))$ is the probability that class labels $u, v$ fit the class labels at positions $d, d-1$. It is calculated as shown in* GENEXAMPLES *in Alg. 1.*

*Proof.* This is a straightforward adaption of the proof of proposition 1 in [4].

All the rest of TreeCRF remains unchanged. That is, we can use the forward-backward algorithm as proposed by [4] to compute $Z(X)$. The forward recursion is defined as $\alpha(k, 1) = \exp F^k(\bot, w_1(X))$ and $\alpha(k, t) = \sum_{k' \in K} \left[\exp F^k(k', w_t(X))\right] \cdot \alpha(k', t-1)$. The backward recursion is defined as $\beta(k, T) = 1$ and $\beta(k, t) = \sum_{k' \in K} \left[\exp F^{k'}(k, W_{t+1}(X))\right] \cdot \beta(k', t+1)$.

We will now turn over to how to use CRFs for making predictions.

**Algorithm 1** Gradient Tree Boosting for SSL as introduced by [4].

**function** TreeBoost($Data, L$)
  **for** $1 \leq m \leq M$ **do** ▷ Iterate Functional Gradient
    **for** $1 \leq k \leq K$ **do** ▷ Iterate through the class labels
      $Sk :=$GenExamples$(k, Data, Pot_{m-1})$ ▷ Generate examples
      $\Delta_m(k) :=$FitRelRegressTree$(S(k), L)$ ▷ Functional gradient
      $F_m^k := F_{m-1}^k + \Delta_m(k)$ ▷ Update Models
  **return** $Pot_M$ ▷ Return Relational Potential
**function** GenExamples($k, Data, Pot_m$)
  $S := \emptyset$ ▷ Initialize relational regression examples
  **for all** $(X_i, Y_i) \in Data$ **do** ▷ Iterate over all training examples
    $\big(\alpha, \beta, Z(X_i)\big) =$ ForwardBackward$(X_i, T, K)$ ▷ Compute forward and backward probabilities
    **for** $1 \leq t \leq T_i$ **do** ▷ Iterate over all positions
      **for** $1 \leq k' \leq K$ **do** ▷ Iterate over all class labels
        ▷ Compute value of gradient at position $t$ for class label $k$
        $P(y_{t-1} = k', y_t = k|X_i) := \dfrac{\alpha(k', t-1) \cdot \exp(F_m^k(k', w_t(X)) \cdot \beta(k, t)}{Z(X_i)}$
        $\Delta(k, k', t) := I(y_{t-1} \subseteq_\Theta k', y_t \subseteq_\Theta k) - P(y_{t-1} \subseteq_\Theta k', y_t \subseteq_\Theta k|X_i)$
        $S := S \cup \{((w_t(X_i), k'), \Delta(k, k', t))\}$ ▷ Update set of relational regression examples
  **return** $S$

### 4.3 Making Predictions

There are several ways for getting a classifier from a trained CRF. We can predict the output sequence $Y$ with the highest probability: $H(X) = \arg\max_Y P(Y|X)$. The Viterbi algorithm [14] can be used for this. Another option is to predict every atom $y_t$ in the output sequence individually. This makes sense when we want to maximize the number of correctly tagged input atoms: $H_t(X) = \arg\max_{k \in K} P(y_t = k|X)$. Finally, one can also use a CRF for sequence classification, i.e., to predict a single label for the entire sequence. To do so, we can simply make a kind of majority vote. That is, we first predict $H(X)$. Next, we count the number of times each class atom was predicted, i.e., $count(c, Y) := |\{i \in \{1, \ldots, T\} \mid y_i = c\}|$. Then, the sequence $X$ is assigned to class $c$ with probability $P(c|X) = T^{-1} \cdot count(c, H(X))$.

## 5 Experiments

Our intention here is to investigate to which extent TildeCRF for logical sequences is competitive with related approaches. To this aim, we implemented our system in Yap 5.1.0 prolog and investigate the following questions:

**(Q1)** Does TildeCRF perform equally well as traditional CRFs?
**(Q2)** If so, are there cases where TildeCRF leads to better results?
**(Q3)** If so, are there real-world datasets on which TildeCRFs performs better than established methods?

In the following, we will describe the experiments carried out to investigate **Q1**–**Q3** and the results.

### 5.1 (Q1) Protein Secondary Structure Prediction

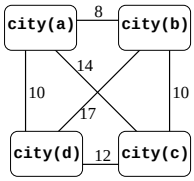
To show that CRFs for logical sequences perform equally well as traditional CRFs, we evaluated our gradient relational tree boosting algorithm on the protein secondary structure predication benchmark considered by Dietterich et. al [4]. The protein secondary structure benchmark was originally published by Qian and Sejnowski [13]. A protein consists of a sequence of amino acid residues. Each residue is represented by a single feature with 20 possible values (corresponding to the 20 standard amino acids). There are three classes: alpha helix, beta sheet, and coil (everything else). There is a training set of 111 sequences and a test set of 17 sequences.

The input features consisted of an 11-residue sliding window and we allowed the regression trees of up to 32 leafs. Dietterich et. al 's TreeCRF attained a test set performance of 64.7%. Our TildeCRF achieved a 64.2% test set accuracy. Qian and Sejnowski's method attained 64.5%, whereas McCallum's Mallet (a gradient based optimization approach for traditional CRFs) reached 62.9%. Thus, TildeCRF is in the range of TreeCRFs. Completely reproducing the TreeCRF results was difficult because we did not know the tree size used by Dietterich et al.. Overall, the results affirmatively answers **Q1**.

### 5.2 (Q2) Job Scheduling

To see whether there are cases where TildeCRF leads to better results than propositional approaches such as TreeCRF, we considered the task of job scheduling. Many jobs in industry and elsewhere require completing a collection of jobs while satisfying resource constraints. Thus, the goal is to arrange a total order among the jobs satisfies all the constraints while taking as little overall time as possible. Here, we will consider a version of the classical *travel salesman problem.*

There are a number of cities $\mathcal{C}$ given and different types of activities $\mathcal{A}$, which can have some parameters. E.g. an activity can be done with normal speed or fast. There is a cost function for traveling from city to city $c_{\text{travel}} : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}_+$, and there is a cost function $c_{\text{act}} : \mathcal{A} \times \mathcal{C} \rightarrow \mathbb{R}_+$ which gives for every city and every activity the costs of doing this activity in that city. It might be the case, that a special activity isn't possible in some cities, therefore $c_{\text{act}}$ can be a partial function. The task within this domain looks now as follows: given is a sequence of activities $a_1, \ldots, a_T$ goal is to find a sequence of cities $c_1, \ldots, c_T$ such that the overall costs are minimized: $\mathit{costs}(c_1, \ldots, c_T) = \sum_{t=1}^{T} c_{\text{travel}}(a_i, a_{i+1}) + c_{\text{act}}(c_i, a_i)$, where $a_{T+1} = a_1$. In the experiments, we considered the instance with 4 cities and 8 possible activities. Each activity `act(Type,Speed)` can be executed with normal speed and fast, therefore `Speed`$\in \{\texttt{normal}, \texttt{fast}\}$ and `Type`$\in \{\texttt{1}, \ldots, \texttt{8}\}$. The travel cost are listed in Figure 3 and the activity costs consists of two parts, namely $c_{\text{act}} = c_{\text{act'}} + c_{\text{speedcosts}}$ as listed in the same figure. To generate a data set, we randomly generated 100 independent activity sequences of length 15 and




| act | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| city(a) | - | 7 | 2 | 1 | 10 | - | - | 2 |
| city(b) | 11 | - | 3 | - | 5 | - | - | - |
| city(c) | 12 | 8 | - | - | 5 | 8 | 10 | 3 |
| city(d) | 13 | 9 | - | - | 5 | 8 | - | 10 |

| | Extra |
|---|---|
| city(a) | 22 |
| city(b) | 50 |
| city(c) | 12 |
| city(d) | 10 |

**Fig. 3.** Job scheduling: Instance with 4 cities and 8 activities, that was used in the experiment. (left) The map for the job scheduling domain. Nodes represent cities and edges transitions with associated costs. (middle) Costs of activities (right) Costs of doing an activity fast in one city.

searched brute force for an optimal travel sequence. This yield sequences such as $X = \langle \texttt{act}(4, \texttt{normal}), \texttt{act}(1, \texttt{fast}), \texttt{act}(8, \texttt{normal}), \texttt{act}(7, \texttt{normal}), \ldots \rangle$ with $Y = \langle \texttt{city(a)}, \texttt{city(d)}, \texttt{city(c)}, \texttt{city(c)}, \ldots \rangle$ We ran two experiments. At first we allowed just ground atoms as tests in the regression trees. This equals to the propositional approach of TreeCRF. In the second experiment we allowed atoms with variables as tests. Figure 4(b) shows the 10-fold cross-validated accuracy of predicted output symbols after each training iteration. One can readily see that TildeCRF outperforms TreeCRF. This affirmatively answers **Q2**.

### 5.3 (Q3) Protein Fold Classification

This experiment is concerned with how proteins fold up in nature. This is an important problem, as the biological functions of proteins depend on the way they fold up. A common approach to protein fold recognition is to start from a protein with unknown structure and search for the most similar protein (fold) with known structure in the database. This approach has been followed by Kersting *et al.* [6] where LoHMMs with the plug-in estimate were able to achieve a cross-validate predictive accuracy of 75%. Notice that the number of parameters of the LoHMMs used were by an order of magnitude smaller than the number of an equivalent HMM (120 vs. approx. 62000). Based on these results, Kersting and Gärtner [7] devised Fisher kernels for logical sequences and achieved a cross-validated accuracy of about 84%.

The data consists of logical sequences of the secondary structure of protein domains. The task is to predict one of the five most populated SCOP folds of alpha and beta proteins (a/b): TIM beta/alpha-barrel (c1), NAD(P)-binding Rossmann-fold domains (c2), Ribosomal protein L4 (c23), Cysteine hydrolase (c37), and Phosphotyrosine protein phosphatases I-like (c55). The class of a/b proteins consists of proteins with mainly parallel beta sheets (beta-alpha-beta units). Overall, the class distribution is as follows (class,#sequences): (c1, 721), (c2,360), (c23,274), (c37,441), (c55,290). Thus, this is a multiclass problems with 5 different classes. Although, CRFs are indeed able to treat multiclass problems, a round robin approach [5] worked better in our experiments. That is, each pair of

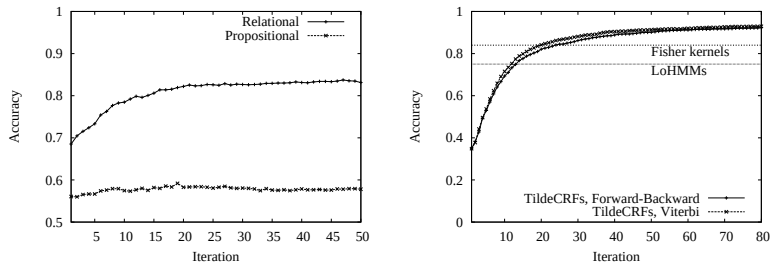



**Fig. 4.** Cross-validated classification accuracy (y axis) vs. number of iterations (x axis). (Left) Job scheduling: TildeCRF achieved 83.13 whereas TreeCRF achieves 57.8%. The difference is significant (one-tailored t-test, $p = 0.05$). (Right) Protein Fold Classification: TildeCRF achieved 92.96%, HMMs (resp. Fisher kernels) for logical sequences achieved 75% (resp. 84%) as indicated by the vertical lines. The differences are significant (one-tailored t-test, $p = 0.05$).

classes is treated as a separate classification problem. The overall classification of an example instance is the majority vote among all pairwise classification problems.

Figure 4(b) summarizes the experimental results. It shows the 10-fold cross-validated accuracy learning curves. The accuracy converges around 92.96% (using Viterbi labeling). Thus, compared to LoHMMs, the error rate of CRFs is about 3 times smaller. The CRF also performed better than Fisher kernels; the error rate dropped about half. The differences are significant (one-tailored t-test, $p = 0.05$). This finally affirmatively answers **Q3**.

## 6 Conclusions

So far, Conditional Random Fields (CRFs) have only been considered for sequences of flat symbols. In this paper, CRFs for logical sequences, i.e., sequences over an alphabet of logical atoms have been introduced and experimentally investigated. Experiments have demonstrated that CRFs can handle logical sequences, the learning algorithm presented performs well in practice, and CRFs for logical sequences can indeed lead to significantly better results than flat CRFs, and HMMs respectively Fisher kernels for logical sequences.

The approach presented suggest a very interesting line of future research, namely to address a more general labeling problem: labeling of sequences of sets of ground atoms with ground atoms. Many problems in learning relational actions and within relational reinforcement learning are of this type.

**Acknowledgments:** The authors thank Alan Fern for useful comments, Tom Dietterich for providing his version of Qian and Sejnowski's benchmark, and Luc De Raedt for his support. The research was supported by the European Union IST programme, contract no. FP6-508861, *Application of Probabilistic ILP II*.

## References


1. C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In *Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD-02)*, pages 143–152, 2002.
2. H. Blockeel and L. De Raedt. Top-down Induction of First-order Logical Decision Trees. *Artificial Intelligence*, 101(1–2):285–297, 1998.
3. L. De Raedt and K. Kersting. Probabilistic Inductive Logic Programming. In *Proc. 15th Int. Conf. on Algorithmic Learning Theory (ALT-04)*, pages 19–36, 2004.
4. T. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proc. 21st International Conf. on Machine Learning*, pages 217–224. ACM, 2004.
5. J. Fürnkranz. Round Robin Classification. *Journal of Machine Learning Research (JMLR)*, 2:721–747, 2002.
6. K. Kersting, L. De Raedt, and T. Raiko. Logial Hidden Markov Models. *Journal of Artificial Intelligence Research (JAIR)*, 25:425–456, 2006.
7. K. Kersting and T. Gärtner. Fisher Kernels for Logical Sequences. In *Proc. of 15th European Conference on Machine Learning (ECML-04)*, pages 205 – 216, 2004.
8. K. Kersting and T. Raiko. 'Say EM' for Selecting Probabilistic Models for Logical Sequences. In *Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence (UAI-05)*, pages 300–307, 2005.
9. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int. Conf. on Machine Learning (ICML-01)*, pages 282–289, 2001.
10. J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 2. edition, 1989.
11. A. McCallum. Effciently inducing features of conditional random fields. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-03)*, 2003.
12. A. Quanttoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17*, pages 1097–1104, 2005.
13. N. Quian and T. J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *JMB*, 202:865–884, 1988.
14. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–285, 1989.
15. M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62:107–136, 2006.
16. S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proc. of the 8th Int. Joint Conference on Artificial Intelligence (IJCAI-03 )*, pages 992–997, 2003.
17. C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proc. of the 21. Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 2005.
18. C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. 21st International Conf. on Machine Learning*. ACM, 2004.
19. B. Taskar, P. Abbeel, and D. Koller. Discriminative Probabilistic Models for Relational Data. In *Proc. of the 8th Conf. on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, 2002.