# FEATURE INDUCTION OF LINEAR-CHAIN CONDITIONAL RANDOM FIELDS

## *A Study Based on a Simulation*

Dapeng Zhang, Bernhard Nebel
*Department of Computer Science, University Of Freiburg, Georges-Khler-Allee Geb. 52, Germany*
*zhangd, nebel@informatik.uni-freiburg.de*

Keywords: Conditional Random Fields; CRF Queue; Feature Induction.

Abstract: Conditional Random Fields (CRFs) is a probabilistic framework for labeling sequential data. Several approaches were developed to automatically induce features for CRFs. They have been successfully applied in real-world applications, e.g. in natural language processing. The work described in this paper was originally motivated by processing the sequence data of table soccer games. As labeling such data is very time consuming, we developed a sequence generator (simulation), which creates an extra phase to explore several basic issues of the feature induction of linear-chain CRFs. First, we generated data sets with different configurations of overlapped and conjunct atomic features, and discussed how these factors affect the induction. Then, a reduction step was integrated into the induction which maintained the prediction accuracy and saved the computational power. Finally, we developed an approach which consists of a queue of CRFs. The experiments show that the CRF queue achieves better results on the data sets in all the configurations.

## 1 Introduction

In natural language processing, a sentence (a sequence of words) needs to be "understood" by a computer. An important task is to label the phrases with e.g. noun, verb, or preposition in the sentences, mapping the segments of the words to the labels. In robotics, agents are equipped with sensors in order to acquire the environment using measurements of the surroundings. The task of labeling is to identify the states according to the temporal sensor data. The states are normally encoded in a vector of variables with discrete values. Similar applications can be found in image processing and in computational genetics.

Researchers have developed several approaches for the sequential labeling tasks. Hidden Markov Models (HMMs), for example, is a well-developed generative model suitable for such a task. The inference of HMMs is based upon joint probabilities (Rabiner, 1990). Compared to HMMs, Conditional Random Fields (CRFs) has a shorter history. It was first proposed by (Lafferty et al., 2001), then gained popularity quickly. CRFs is a discriminative model based on conditioned probabilities. In CRFs, a hidden label is globally conditioned on all the observations in the sequence. CRFs outperformed HMMs in the experiments on the benchmarks in natural language processing (Lafferty et al., 2001).

Feature induction of CRFs was first introduced by (McCallum, 2003). As training CRFs requires considerable computational power, the induction is mainly about how to define some more efficient evaluations for incrementally inducing the feature functions of CRFs. The method of McCallum was tested on name entity recognition and noun phrases segmentation. It resulted in comparable prediction accuracy to the approaches other than CRFs(McCallum, 2003). There are a few further works on the topic. In (Dietterich et al., 2004), the boosting algorithm is embodied for simultaneously inducing features and training CRFs. All these works experimented on the same synthetic data, which served as a testbed for the comparison.

Our research was originally motivated by the explanation of the data of table soccer games. The sequential game data is made available via a game recorder (Zhang and Hornung, 2008). The labels are

the actions of human players e.g. *lock*, *attack*, *block*, *pass*, and *dribble*. These data considerablly differ from the synthetic data used by the authors mentioned above. At each time slice, the measurements are encoded in a vector of Boolean variables. Each skill of human players consists of hundreds of such time slices. The annotation of the data is very exhausting. We have spent about 80 hours annotating 200 sequences, which are not enough for feature induction and supervised learning of CRFs. This difficulty motivated the idea of the simulation.

A sequence generator was built to create data sequences, and to label them automatically, simulating the data of the table soccer games. The core idea is to create the first CRFs, with the feature functions and the parameters generated randomly. Then, the second CRFs can be obtained from the feature induction methods as mentioned above. The first CRFs can thus provide the information to estimate the second one and the induction algorithm. This approach creates an extra phase to explore several basic properties of the feature induction of linear-chain CRFs. Consequently, it fosters several further developments. The highlights of this work are summarized as follows:

- We integrate a novel reduction step in the induction, which can keep the accuracy of the prediction and decrease the number of feature functions, thus making the learning more efficient.
- We develop a method to train a queue of CRF models from the data. CRF queues guarantees a no worse prediction accuracy than the single CRFs. It outperformed the single ones on the data sets in all the configurations. To the best of our knowledge, we are the first who propose the idea of CRF queues.

## 1.1 Related Works

*Variable and feature selection* is a well-developed research area. Guyon summarized the issues and the main approaches in the area in (Guyon and Elisseeff, 2003). If we put the specific CRFs problem into a more general context, many ideas and methods can be used. For example, feature reduction is widely employed in this area. To our knowledge, it is not yet applied to CRFs.

We found only a few works about feature induction of CRFs. Chen et al. compared a gradient based approach (Chen et al., 2009) to the McCallum method. Both approaches use the framework shown in Figure 1. In *candidate evaluation*, the gradient based approach searches for the candidates that make the objective function decrease fastest. Instead

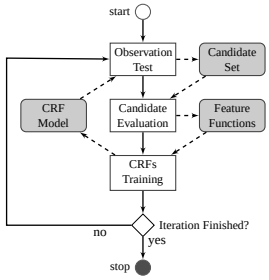

Figure 1: The Feature Induction of CRFs

of simply counting in the *observation test*, some researchers integrated the boosting method (Dietterich et al., 2004). The approaches in this direction can simultaneously induce features and train CRFs, which have the more compact model, and therefore being efficient in the computation.

Our implementation is based on a CRF training algorithm - Stochastic Meta Descent (SMD) (Vishwanathan et al., 2006), and the feature induction framework by McCallum. The experiment platform was implemented according to the descriptions in the publications. We did not use any existing source code from the authors or the open source toolkit via Internet. The main reason is that the sequential data in this work are very different from the data in the synthetic benchmarks. In addition, building a platform from scratch creates more chances to find unique and novel ideas.

# 2 Preliminaries

A game recorder was developed to record table soccer games of human (Zhang and Hornung, 2008). The data are collected from 14 sensors, which are mounted on a regular game table. They measure the position and angle of each game rod, and the position of the game ball. The Frequency of the recorder is about $200Hz$. The sensor data were transferred to 52 Boolean variables via a discretization method. The labeling task is to identify the skills of human.

In this work, we define the data by using a typical notation in data classification. The sequential data has the form $(X,Y)$, where $X$ is an observation sequence $(B_1,B_2,...,B_I)$ and $Y$ is the state sequence. $Y=(y_1,y_2,...,y_I)$, where $I$ is the length. At each state $y_i$, a corresponding $B_i$ can be observed, which is a vector of Boolean variables, $B_i=(b_{i,1},b_{i,2},...,b_{i,C})$, where $C$ is the number of the variables.

Conditional Random Fields is an undirected graphic model in the exponential family. The clique decomposition of CRFs supports the inference of the

distributes in an arbitrary graph structure. We focus on linear-chain structure in this paper. In CRFs, the probabilities of a sequence of labels $Y$ given the observations $X$ are defined in the following equation.

$$p(Y|X) = \frac{exp(\sum_{i=1}^{I} \Theta \cdot F(y_{i-1}, y_i, X))}{Z(X)} \quad (1)$$

The parameters $\Theta$ of CRFs can be estimated by a training process, in which $F$ is assumed to be known. Given the training data $D = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = \{X_1, X_2, ..., X_N\}$, $\mathbf{Y} = \{Y_1, Y_2, ..., Y_N\}$, the training algorithm maximizes the likelihood of the CRF model.

$$\Theta^* = \arg\max_{\Theta} \sum_{n=1}^{N} p(Y_n|X_n) \quad (2)$$

Feature induction is a difficult problem because training CRFs requires considerable computational power. In McCallum's approach, there are mainly three layers of the evaluations (McCallum, 2003), shown as three rectangles in the center of Figure 1. From top to bottom, each step to the lower layer requires more computational powers of several levels of significance. In *candidate evaluation*, each candidate is evaluated by measuring how much it can increase the gain $G(f_{K+1})$, given in Equation 3.

$$G(f_{K+1}) = \max_{\theta_{K+1}} \sum_{n=1}^{N} (p_{f_{K+1}}(Y_n|X_n) - p(Y_n|X_n)) \quad (3)$$

Here $p_{f_{K+1}}$ is from the CRFs that includes an extra candidate $f_{K+1}$. Its weight $\theta_{K+1}$ can be calculated by fixing $\Theta$ so that the evaluation can be done much faster than training the whole CRFs.

# 3 Simulation

A CRF model describes a stochastic process, which reveals the relations among the observations and the hidden labels. In the training process, the success of the CRFs hints that the acquired stochastic process matches the patterns in the data. First the training data are avaiable, then the CRFs is trained from them. The idea of the simulation goes the retrograde way. First a CRFs is generated, then it can be used to compute the hidden labels of any randomly generated observations. The following is the assumption which bridges the simulation and the simulated process.

- **The stochastic processes in the target system can be described as a CRF model.**

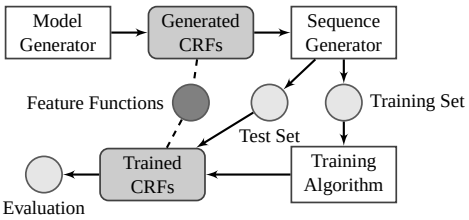

Figure 2: Trainning of CRFs with the shared features

The simulation is shown as the upper row of the boxes in Figure 2. There are mainly two algorithms. A CRF model is generated by the *model generator*. The Simulation provides a platform for studying a wide range of CRFs. After we exploring on different situations, five configurations $\{S_1, S_2, S_3, S_4, S_5\}$ are carefully chosen, which are challenging for the induction issues, being not too hard or too easy. The configurations are designed for the comparisons of the different levels of the conjunctions and the interdependencies.

The simulation was run to create the following data: 100 data sets for each configuration $S_i$. Each set contains a training set of 1000 sequences, and a test set of 500 sequences.

## 3.1 The Trained CRFs

The experiments are based on 8 computers, each with 8 AMD cores at $2.3GHz$ and $32G$ memory. These CPUs are driven by a grid system, on which 64 tasks can be run in parallel. The experiments described in this paper altogether took about 14 days in the grid system.

In the **first experiment**, we assume $F$ is known. Figure 2 shows the scenario. The training algorithm (SMD) was run for maximum 10000 iterations (batches) on each data set, so that the resulted CRFs are well-trained. The trained model and the simulated model are compared in three aspects. The results are shown in Figure 3. We denote the correct sequence probability as $p_r$, and $p_w$ is the wrong probability. The prediction accuracy is the rate of the correct sequences. The values in the figure are the average values over all 500 models and the data sets.

We can summarize the observations as follows: The simulated CRFs cannot be cloned via supervised learning. With the shared feature function, the trained CRFs can achieve an average accuracy of about 90%.

The detailed information on the trained models over the configurations are shown in Figure 4. We inspect the prediction accuracy of the trained models. By using each configuration, 100 simulated models were generated. "Minimum" means the trained model

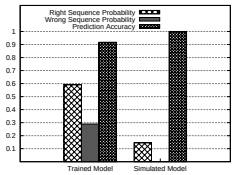


Figure 3: A comparison to the simulated Models

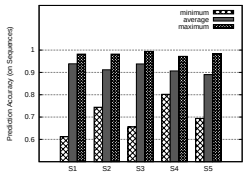


Figure 4: The accuracy of the trained CRFs

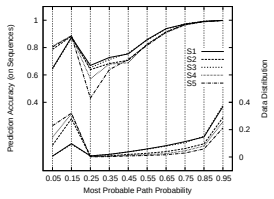


Figure 5: The distributions of the accuracy of the prediction over the probabilities of the most probable path.

performed worst in the estimation, while "maximum" is the best. The average performances of the trained models are roughly the same over the five configurations. The performance of a trained CRFs depends on both $F$ and $\Theta$ of the simulated model. $\Theta$ itself has significant impact on the performance.

The trained CRFs can not only compute the most possible explanation for an input $X$, but also associate this explanation with a probability value. In Figure 5, we show the relations between the prediction accuracy and the probability of the most probable paths. The figure consists of two parts. The upper part is about the accuracy of the prediction. The lower part is the distributions of the data in each grid.

These distributions are interesting. If there is an axis for the probability of the most probable path, the training based on maximum likelihood pushes a large number of sequences in the training set towards the higher value direction of the axis. If the axis is divided into two parts at the middle point, the higher part has a higher accuracy than the lower part.

# 4 Feature Induction

The induction process iterates over three steps. In this section, we develop the forth step: feature reduction. A subset of features $F_r$ is to be removed from $F$, where $F$ is the set of so far induced features, $F_r \subset F$. For each $f_k \in F$, a gain value $G_r$ is defined as a measurement for the reduction. We modified the gain $G(f_{K+1})$ in Equation 3 for the reduction. The difference is $f_{K+1} \notin F$, while $f_k \in F$. $G$ is calculated in iterations before the traing of the CRFs; $G_r$ can be

Table 1: Algorithm: Inducing Features with Reduction

| | |
|---|---|
| | **input:** Training Examples $(\mathbf{X}, \mathbf{Y})$ |
| | **output:** CRFs: $(F, \Theta)$ |
| 1 | $F_{0,I_0} = \emptyset$ |
| 2 | **for** $i_0 = 1...I_0$ **do** |
| 3 | $F_{i_0,0} = F_{i_0-1,I_0}$ |
| 4 | **for** $i_1 = 1...I_1$ **do** |
| 5 | $F_{i_0,i_1} = F_{i_0,i_1-1} \cup$ {new Features from the *Observation Test*} |
| 6 | Compute $\Theta_{i_0,i_1}$ on $F_{i_0,i_1}$ via Equation 2 |
| 7 | **end** |
| 8 | ***Reducing*** $F_{i_0,I_1}$ ***via Equation 5*** |
| 9 | **end** |
| 10 | Choose $F = F_{i_0,i_1}$ where $F_{i_0,i_1}$ yield to the best performance on the $(\mathbf{X}, \mathbf{Y})$ |
| 11 | Compute $\Theta$ on $\{F_{i_0,i_1}, \Theta_{i_0,i_1}\}$ via Equation 2 |
| 12 | return $(F, \Theta)$ |

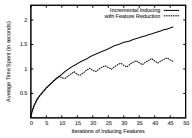


Figure 6: Time spent on the training

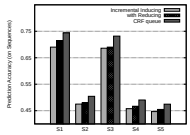


Figure 7: The Evaluation

calculated without any iteration after the training. In the reduction, the features with a $G_r$ lower than a predefined threshold $C_0$ can be removed.

$$G_r(f_k) = \sum_{n=1}^{N} (p(Y_n|X_n) - p_{\theta_k=0}(Y_n|X_n)) \quad (4)$$

$$F' = F - \{f_k | G_r(f_k) < C_0\} \quad (5)$$

The induction algorithm with feature reduction is written in pseudo-code in Table 1. The reduction is called after several iterations of observation test, candidate evaluation, and CRFs training. The algorithm stops after some iterations of the reduction steps.

In the training process, each feature function has a weight. Intuitively, the feature reduction can reduce the number of parameters. Consequently, it should save the computational power required by the training. The **second experiment** is designed for this comparison. The feature induction algorithms are run independently with and without the reduction for 50 iterations. Figure 6 shows the results.

In the figure, the horizontal is the function calls of the training. The vertical is average time spent on processing 100 batches of SMD. During the induction process, features are added to $F$ incrementally. The training thus requires more and more time to compute the weights of the feature functions. The dotted curve shows the performance of the algorithm with feature reduction. It is serrated because the reduction step is not called in every iteration. The reduction step can

save the computational power more than 30% in the long run.

Although the reduction makes the induction process faster, does it decrease the prediction accuracy of the resulted CRFs? The **third experiment** is designed to investigate this issue. The induction algorithms with and without the reduction were run independently over the $5 \times 100$ training sets. The models which yield to the best performance on the training set are selected for the evaluation. The results are illustrated in Figure 7. Based on the average results, the algorithm with the reduction outperformed the one without reduction in all the configurations.

In Figure 7, the performances of the models can be roughly classified into two categories: $\{S_1, S_3\}$ the configurations with single features, and $\{S_2, S_4, S_5\}$ the configurations with conjunctive features. The conjunctive features of the simulated models make the induction tasks more difficult. The feature overlapping of the simulated models [only] slightly affects the difficulties of the induction in the experiments.

As illustrated in Figure 6, the algorithm with the reduction runs faster because the number of feature functions is lower. How many features were induced in the experiments? In the upper part of Figure 8, we show the results. The "13" comes from the simulated models, which serves as a based line for the comparison. The learning induced the features several times more than the target features in the simulated models. Surprisingly, it did not cause a severe overfitting problem. For the induction with the reduction, compared to the configurations with conjunctive features, more features were induced in the configurations with a single feature.

## 4.1 CRF Queues

In Section 3.1, the experiments showed that along the axis of the probabilities of the most probable paths, a higher value has a higher accuracy. The basic idea of CRF queues is to build a queue of CRF models, and each model uses the higher probability part to do the prediction. If the probability of a sequence is lower than a threshold $t$, the data are passed to next model.

$$r(D,t) = \{(X',Y') | (X',Y') \in D, p(Y''|X') > t)\}$$

If $D(\mathbf{X}, \mathbf{Y})$ is the training set, a filter function is defined as follows, where $Y''$ is the most probable explanation of $X'$. We define $D' \subset r(D,t)$ as the set of the sequences which are correctly explained. The threshold $t^*$ can be calculated via:

$$t^* \cong arg_t \frac{|D'|}{|r(D,t)|}) = C_1 \qquad (6)$$

Table 2: Algorithm: Inducing CRF Queue from Data

| | |
|---|---|
| | **input:** Training Examples $D(\mathbf{X}, \mathbf{Y})$ |
| | **output:** Learned CRF Queue $Q((F_m, \Theta_m), t_m), m = 1...M$ |
| 1 | $D_1 = D$; $Q_0 = (\emptyset, \perp)$ |
| 2 | **for** m=1...M **do** |
| 3 | Compute $(F_m, \Theta_m)$ on $D_m$ via table 1 |
| 4 | Compute $t_m$ on $D_m$ via Equation 6 |
| 5 | $Q_m = Q_{m-1} \cup ((F_m, \Theta_m), t_m)$ |
| 6 | Compute $D_{m+1}$ by Applying $((F_m, \Theta_m), t_m)$ on $D_m$ via Equation 7 |
| 7 | **end** |
| 8 | return $Q_M$ |

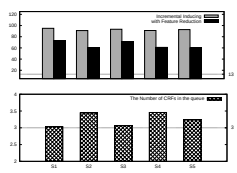


Figure 8: feature functions and CRFs in the queue

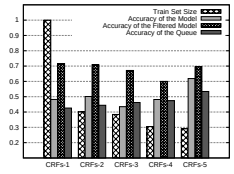


Figure 9: An Example of CRF Queue

In the equation, $C_1$ is a selected accuracy higher than the accuracy of the first CRFs in the queue. In order to build the queue, assume the first CRF model is already induced via the algorithm shown in Table 1 – we can then use $t^*$ to filter the training set for the next model in the queue. The sequences with a probability of the most probable explanation higher than $t^*$ are removed from the training set. The rests are used to induce the next model in the queue.

$$D_{m+1} = D_m - r(D_m, t^*) \qquad (7)$$

The algorithm of inducing CRF queues is written in pseudo-code in Table 2. In each iteration, a CRF model is built; the threshold is computed; and the training set is filtered. The algorithm is run until no CRFs with the required accuracy ($C_1$) can be induced from the filtered data.

A sequence $X$ can be explained by the queue in the following way. $X$ is explained by the first model $p_1$ in the queue. If $p_1(Y'|X) > t_1$, where $t_1$ is the threshold, then $Y'$ is the explanation of the $X$. Otherwise, $X$ is passed to the second model. If $X$ cannot be explained by any of the models in the queue, the model with the highest accuracy is chosen to explain the sequence. Figure 9 shows an example of the CRF queue.

There are 5 models altogether in the queue. The first column in the figure shows the training set used for the model. In the experiment, each training set consists of 1000 sequences. 1.0 means all of them are used to induce CRF-1. Along the queue, fewer and fewer data are passed to the next model. CRF-5 is trained by less than 300 sequences.

The second column in the figure illustrates the performance of each model on the training set. To our surprise, so many models can be induced with the reduced training sets. Their performances on the training data can be improved by inducing a new set of the features. The third column is the accuracy of using the model with the threshold $t_m$, we call it the filtered model. $t_m$ defines the higher part along the axis of the probability of the most probable paths. CRF queues provide a more accurate prediction because the third column is higher than the second one. The fourth column is the performance of the CRF queue. It is computed by using the current acquired models. For example, in CRF-3, the first, second, and third models build the queue. Along the queue, the evaluations are better and better. In CRF-1, the second column is higher than the fourth one because of overfitting. The performance of the model is better on the training set than on the test set.

In the example, if $X$ cannot be explained with a probability higher than the threshold $t_m$ by all the models in the queue, CRFs-5 should be chosen to explain $X$. The reason is that its second column is the highest one over the second columns of all the models. To summarize, the third columns of the first four models and the second column of the fifth model are chosen to explain $X$. Their overall performance, the fourth column in CRFs-5, is lower than any of these columns because of overfitting. From another point of view, the values of the chosen columns are based on the training data; the overall estimation of the queue is the evaluation on the test set.

The **fourth experiment** was designed to evaluate CRF queue. The algorithm shown in Table 2 was run on all data sets. The average results over 5 configurations are shown as the third columns in Figure 7. The CRF queue outperformed the single model approaches for about 4% on average in all configurations. We show the number of models in the queue in the lower parts in Figure 8. The results are averaged over the 100 sets in each configurations. The number of models is above 3, which hints that the queue works well in most cases. $\{S1, S3\}$ has a shorter queue because the performances of the single model approaches in these configurations are better, as shown in Figure 7. The CRF queue is shorter when the single model approaches work better.

## 5 Conclusion

In this paper, we constructed a simulation framework to investigate the issues of inducing features of linear-chain CRFs. The simulation helps to gain a new phase to compare the simulated CRFs and the induced CRFs. We used a large amount of experiments to explore the properties of the learned CRFs. Moreover, we developed a feature reduction method that can be integrated into the induction process, and a queue of CRF models can be constructed which yields a better performance. CRF queues guarantees accuracy no worse than the single model approaches.

We did not use the open source CRF toolkit and did not yet experiment on the benchmarks. In the future, we will adapt our code to process the benchmark data. The simulation framework sets a basis for interesting research on CRFs in several directions. It would be interesting to explore the bootstrap issues. In CRF queue, we defined a method to filter the training set. Another method could be to construct a decision tree to first classify the training set, then use the data in each class to induce CRFs.

## REFERENCES


Chen, M., Chen, Y., Brent, M. R., and Tenney, A. E. (2009). Gradient-based feature selection for conditional random fields and its applications in computational genetics. In *ICTAI '09: Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 750–757, Washington, DC, USA. IEEE Computer Society.

Dietterich, T. G., Ashenfelter, A., and Bulatov, Y. (2004). Training conditional random fields via gradient tree boosting. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 28, New York, NY, USA. ACM.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*, pages 282–289.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *UAI*, pages 403–410.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 969–976, New York, NY, USA. ACM.

Zhang, D. and Hornung, A. (2008). A table soccer game recorder. In *Video Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.