Regular-pattern-sensitive CRFs for Distant Label Interactions

Sean Papay* **and** **Roman Klinger*** **and** **Sebastian Padó†**

*Fundamentals of Natural Language Procccessing
University of Bamberg, Germany
(sean.papay|roman.klinger)@uni-bamberg.de

†Institute for Natural Language Processing
University of Stuttgart, Germany
sebastian.pado@ims.uni-stuttgart.de

Abstract

While LLMs have grown popular in sequence labeling, linear-chain conditional random fields (CRFs) remain a popular alternative with the ability to directly model interactions between labels. However, the Markov assumption limits them to interactions between adjacent labels. Weighted finite-state transducers (FSTs), in contrast, can model distant label–label interactions, but exact label inference is intractable in general. In this work, we present regular-pattern-sensitive CRFs (RPCRFs), a method of enriching standard linear-chain CRFs with the ability to learn long-distance label interactions through user-specified patterns. This approach allows users to write regular-expression label patterns concisely specifying which types of interactions the model should take into account, allowing the model to learn from data whether and in which contexts these patterns occur. The result can be interpreted alternatively as a CRF augmented with additional, non-local potentials, or as a finite-state transducer whose structure is defined by a set of easily-interpretable patterns. Critically, exact training and inference are tractable for many pattern sets. We detail how an RPCRF can be automatically constructed from a set of user-specified patterns, and demonstrate the model's effectiveness on a sequence of three synthetic sequence modeling datasets.

1 Introduction

Sequence labeling is a common paradigm which has provided a useful frame to modeling many tasks in machine learning, ranging from Natural Language Processing (e.g., part-of-speech (POS) tagging (Schmid, 1994; Chiche and Yitagesu, 2022)) to protein structure prediction (Wang et al., 2016; Mukanov and Takhanov, 2022) and weather pattern prediction (Raje and Mujumdar, 2009).

Sequence labeling is fundamentally a structured prediction task – individual labels are not in general independent from one another, but should form a coherent label sequence. E.g., in weather pattern prediction, while the weather at a specific time point may be uncertain, it should still be highly correlated to the weather at nearby time points. In part-of-speech tagging, where an individual word like "duck" may have ambiguous POS in isolation, models strive to tag all words so that they obtain a grammatical global POS sequence.

In recent years, research in NLP, but also beyond, has been dominated by the impressive developments in the area of neural networks. With the widespread success of LLM encoders such as BERT (Devlin et al., 2019), a common approach is to represent the entire input sequence in the joint latent space of such an LLM encoder, and to make independent predictions for each token conditioned on this joint latent representation.[1] With a sufficiently powerful encoder, models can try to sidestep the issue of modeling interactions between output labels by modeling the interactions at the level of the input sequence.

However, the success of LLMs is predicated on both practical and conceptual factors.

- First, at the practical level, LLMs appear to be a class of learning methods that capitalize very well on the specific properties of natural language – that is, the fact that most (hard) constraints are local, that sequences are fairly predictable, and that symbols are mildly ambiguous. In contrast, research has found that LLM-based models are not such clear success stories when applied to languages with different, properties, notably 'crisper' ones such as logics (Liu et al., 2024) and programming languages (Fang et al., 2024)

- Second, LLMs work best when large

[1] Concretely, this would correspond to e.g. feeding the input into BERT, and using a position-wise softmax output layer.

Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025), pages 26–35
August 1, 2025 ©2025 Association for Computational Linguistics

amounts of data are available for pre-training, which again is not the case for all domains.

- Third, there are conceptual limits according to which even strong encoder-based approaches to sequence modeling often cannot be certain about a prediction. This may be due to underlying ambiguity (e.g. no model can be certain about the POS tags in an ambiguous sentence like “I saw her duck.”), limits imposed by data availability or model complexity, or simply the difficulty of the underlying task. In such cases, while models won’t be able to always guess the correct label sequence, they stand to benefit from explicitly modeling interactions between labels, such that they can exclude unlikely label sequences.

For these reasons, we believe that structured prediction, with its ability to cope with a larger typology of input languages, still warrants investigation as a general approach to modeling interactions between labels.

In this paper, we extend linear-chain conditional random fields (CRFs) (Lafferty et al., 2001), maybe the most established approach to modeling label–label interactions. Within this framework, interactions between adjacent labels are directly modeled, but distant labels are assumed to only interact by proxy of their intervening labels. This conditional independence assumption makes CRFs well-suited for modeling local interactions between labels, but leads to difficulties when long-distance interactions are important, such as in quotation detection (Scheible et al., 2016) but fundamentally unable to account for more global constraints in the interest of computational efficiency.

A related class of models are (neural) weighted finite-state transducers or FSTs (Mohri, 1997; Eisner, 2002; Rastogi et al., 2016). Like CRFs, weighted FSTs define a distribution over label sequences conditioned on an input sequence, but they do so by modeling transitions through latent *states*. FSTs also obey a Markov assumption, but in their case, this is a conditional independence assumption on states, not on labels. While the state at a given time step depends directly only on the states of neighboring time steps, the output label at that time step may not be conditionally independent from distant output labels, depending on the structure and weights of the underlying automaton, and which paths through that automaton might explain those labels.

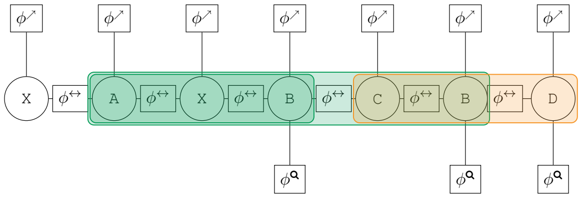


Figure 1: A linear-chain CRF can only model probabilities of labels occurring at particular positions ($\phi^{\nearrow}$), and probabilities for labels being adjacent to one another ($\phi^{\leftrightarrow}$). In particular, linear-chain CRFs cannot encourage or discourage the presence of nonlocal patterns in the label sequence, e.g. the regular expression patterns `A.*B` and `C.*D`. With an RPCRF, a set of such patterns can be specified, and the model can learn the probability of each of those patterns occurring at different positions of the label sequence ($\phi^{🔍}$).

This ability to model distant interactions makes weighted FSTs more powerful than CRFs but also computationally more demanding. When the underlying automaton is nondeterministic, inferring the most probable label sequence is NP-hard (Casacuberta and de la Higuera, 2000). Furthermore, it is often not obvious how to chose the crucial automaton structure in order to be sensitive to specific types of label–label interactions.

In this paper, we propose regular-pattern-sensitive CRFs (RPCRFs), a model architecture combining the strengths of CRFs and FSTs for sequence labeling. An RPCRF can be seen as a linear-chain CRF equipped with the ability to be sensitive to specific types of long-distance interactions between labels. When instantiating a model, a user specifies a set of regular-expression label patterns, such that the resulting model will be able to punish or reward occurrences of those patterns at specific positions in the label sequence. In this way, particular types of long-distance interactions can be chosen in a task-specific manner, while the model is still free to learn how and when those interactions are important for sequence labeling. Figure 1 illustrates how an RPCRF can model long-distance interactions through sensitivity to patterns. Equivalently, RPCRFs are a framework for specifying automaton structures for FSTs in an easily interpretable manner such that the resulting FST will be sensitive to exactly those long-distance interactions the user would like to model. Unlike in the general-case for weighted FSTs, an RPCRF will always define a deterministic automaton, support efficient exact inference like CRFs.

We first characterize RPCRFs formally, and discuss how one can be implemented as a linear-chain CRF defined over an alternative label sequence. We then discuss the time-complexity of parameter estimation and inference. Finally, we perform a number of experiments on synthetic data wherein we compare an RPCRF against a linear-chain, demonstrating different types of nonlocal label structures an RPCRF can be made sensitive to through an appropriate choice of patterns.

2 Model architecture and construction

2.1 Formal description

For a label set Σ, a standard linear-chain CRF, parameterized by θ, defines a distribution over label sequences $\boldsymbol{y} \in \Sigma^*$ conditioned on input sequences $\boldsymbol{x}$ in terms of a *transition potential function* $\phi_\theta^{\leftrightarrow}$ and a *emission potential function* $\phi_\theta^{\nearrow}$:

$$P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z} \prod_i \left(\phi_\theta^{\leftrightarrow}(y_i, y_{i+1}) \cdot \phi_\theta^{\nearrow}(\boldsymbol{x}, y_i, i) \right) \tag{1}$$

Z, the partition function, acts as a constant of proportionality, and is chosen such that all probabilities sum to unity:

$$Z = \sum_{\boldsymbol{y}'} \left(\prod_i \left(\phi_\theta^{\leftrightarrow}(y'_i, y'_{i+1}) \cdot \phi_\theta^{\nearrow}(\boldsymbol{x}, y'_i, i) \right) \right) \tag{2}$$

The transition potential function is applied pairwise to each pair of adjacent labels, and is responsible for modeling label-to-label interactions, while the emission potential function models the interaction between the input sequence and individual labels.

An RPCRF can be understood as standard linear-chain augmented with additional potential functions defined by the set of specified patterns. An RPCRF is additionally hyperparameterized by a set $\mathbb{L}$ of regular-language patterns, and includes a *pattern potential function*, $\phi_\theta^{\text{🔍}}$, to model the likelihood of different label-sequence patterns ending at different positions in the sequence:

$$P_\theta^{\mathbb{L}}(\boldsymbol{y} \mid \boldsymbol{x}) \propto P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) \cdot \prod_{L \in \mathbb{L}} \prod_i \phi_\theta^{\text{🔍}}(L, i)^{\mathcal{I}} \tag{3}$$

$$\text{with } \mathcal{I} = \mathbb{1}(L \text{ matches } \boldsymbol{x} \text{ ending at position } i)$$

In principle, since deciding if an arbitrary regular-language pattern matches ending on a given label index requires looking at all preceding labels, this defines a CRF without linear-chain structure wherein all labels are adjacent to one another. However, as we will show next, the RPCRF distribution can be represented as the distribution over an auxiliary CRF which *does* have a linear-chain structure, allowing for tractable training and exact inference for these models.

2.2 Construction from patterns

This subsection describes how training and inference can be done with RPCRFs. As described, these models are highly cyclic CRFs, for which exact training and inference are infeasible in general. However, we will present a method for defining an auxiliary, linear-chain CRF whose distribution happens to equal the RPCRF distribution. As this auxiliary CRF has a linear-chain structure, parameter estimation and inference can be done with the forward and Viterbi algorithms respectively.

We begin by defining a deterministic finite-state automaton (DFA) Π whose state space captures information about all patterns in $\mathbb{L}$. Specifically, we would like to define Π such that, as Π processes the label sequence $\boldsymbol{y}$, the current state of Π at time step i can tell us which set of patterns in $\mathbb{L}$ match $\boldsymbol{y}$ ending at position i. We achieve this as follows: for each $L \in \mathbb{L}$, we construct a DFA for the language $L' = \Sigma^* \oplus L$, i.e., the language of label sequences with a suffix matching L. We can then construct Π as a product of the automata for these L', whose states are $|\mathbb{L}|$-tuples of the states the constituent automata. While accepting $\boldsymbol{y}$ through Π, we can examine the state-tuple at each time-step, and determine which set of patterns match $\boldsymbol{y}$ ending at that time step by checking which states in that tuple are accepting states in their original automata. We can interpret Π as a state-labeled DFA, where each state is labeled with the set of patterns which match $\boldsymbol{y}$ ending at that time-step when that state is reached. In particular, for each state q in Π, we will notate the set of patterns which label that state as $\mathbb{L}_{[q]} \subseteq \mathbb{L}$.

Once we have constructed Π, we will define an auxiliary linear-chain CRF whose label set is the set A of arcs (labeled arrows) of Π. As Π is deterministic, each possible label sequence $\boldsymbol{y} \in \Sigma^*$ corresponds to exactly one path through Π – as a path through Π can be represented as a sequence of arcs $\boldsymbol{\pi} \in A^*$, that path can be used directly as a label sequence for our auxiliary CRF. We specifically construct our auxiliary CRF such that the probability assigned to each arc sequence $\boldsymbol{\pi}$ is equal to the RPCRF probability for the corresponding label sequence $\boldsymbol{y}$:

$$\begin{aligned} P'_\theta(\boldsymbol{\pi} \mid \boldsymbol{x}) &= \frac{1}{Z}\prod_i \left(\phi'^{\leftrightarrow}_\theta(\pi_i, \pi_{i+1}) \cdot \phi'^{\nearrow}_\theta(\boldsymbol{x}, \pi_i, i)\right) \\ &= P^{\mathbb{L}}_\theta(\boldsymbol{y} \mid \boldsymbol{x}) \end{aligned} \tag{4}$$

We achieve this through suitable definition of our auxiliary CRF's transition function $\phi'^{\leftrightarrow}_\theta$ and emission function $\phi'^{\nearrow}_\theta$:

$$\phi'^{\leftrightarrow}_\theta(\langle q \xrightarrow{\mathrm{a}} r\rangle, \langle s \xrightarrow{\mathrm{b}} t\rangle) = \begin{cases} \phi^{\leftrightarrow}_\theta(a, b) & \text{if } r = s \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\phi'^{\nearrow}_\theta(\boldsymbol{x}, \langle q \xrightarrow{\mathrm{a}} r\rangle, i) = \begin{cases} 0 & \text{if } \mathcal{C} \\ \phi^{\nearrow}_\theta(\boldsymbol{x}, a, i) \cdot & \\ \prod\limits_{L \in \mathbb{L}_{[r]}} \phi^{\text{🔍}}_\theta(L, i) & \text{otherwise} \end{cases} \tag{6}$$

where $\mathcal{C} = \mathbb{1}(i = 1 \text{ and } q \text{ is not initial state of } \Pi)$

These definitions ensure that our auxiliary CRF will only assign nonzero probability to proper paths through Π (which start at the initial state and contain only valid transitions), and, for those paths, will assign a probability to path $\boldsymbol{\pi}$ equal to the RPCRF distribution's probability for the corresponding label sequence $\boldsymbol{y}$. Figure 2 shows a worked example of this construction, illustrating the state-labeled automaton obtained from a set of patterns and the auxiliary CRF computing a probability for a path through that automaton.

As the time- and space-complexity of our learning and inference algorithms will depend on the size of Π, we would like to make Π as small as possible. This can be achieved by minimizing all automata for our L' languages before constructing Π, and pruning unreachable states in Π.

In the worst case, all states in Π will be reachable, and the size of Π equals the product of the minimal number of states for all languages in $\mathbb{L}$, i.e. it is exponential in $|\mathbb{L}|$. However, we observe that in many cases where different patterns "share information," we can do significantly better than this upper bound. For instance, when one pattern is a strict prefix of another, we can include the prefix pattern "for free", without necessitating any additional states, as the product construction has the effect of simply labeling which states in the larger automaton match the prefix. Unfortunately, a full characterization of such synergies falls outside the scope of the current work.

3 Experiments

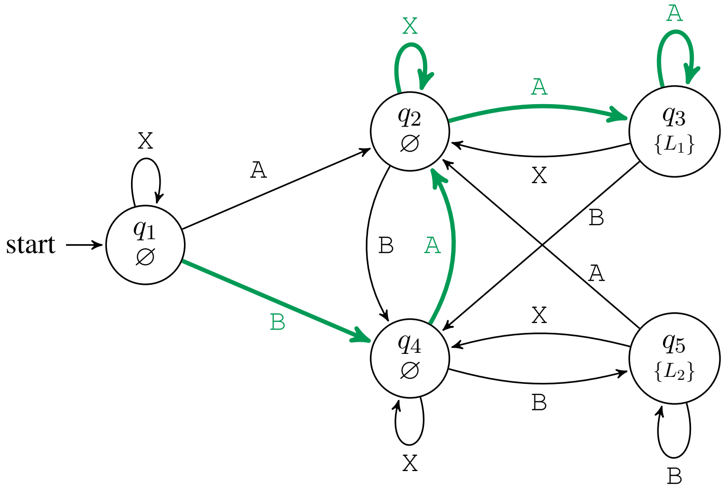
To concretely demonstrate the differences between RPCRFs and linear-chain CRFs, we perform three experiments with synthetic data, each demonstrating a particular class of problem where an RPCRF can model interactions not capturable by a linear-chain CRF. Each experiment will feature a synthetic dataset exhibiting a certain type of label structure, and a pattern set designed to be sensitive to that label structure. As all labels are trivially independent under certainty (i.e. when all label probabilities are either zero or one), all synthetic data tasks are fundamentally underspecified, such that models will always need to "guess" the right answer from some space of possibilities. Thus, for each experiment, in addition to reporting model performance, we will report the highest level of performance possible by a hypothetical model employing an optimal strategy.

For all synthetic data experiments, we will use digits as input symbols, and letters and underscores as output labels, with the specific meanings of these symbols varying by experiment. For all experiments, the emission and pattern potential functions are represented with a biLSTM neural network (Hochreiter and Schmidhuber, 1997), and the transition function is represented as a parameter matrix. All parameters are jointly optimized until convergence using the Adam optimizer (Kingma and Ba, 2015).

We evaluate all tasks via exact-match accuracy. That means that we count a model as correct only when it predicts the label sequence exactly correct, and we don't assign partial credit. This turns out to be quite important, as many less-strict evaluation methods are explicitly insensitive to the global structures we are trying to capture. For instance, when evaluating by token-wise accuracy, models are not rewarded for producing globally plausible label sequences, only for ensuring that each individual label is likely in isolation, something that linear-chain CRFs are already capable of.

3.1 Experiment 1: Cardinality patterns

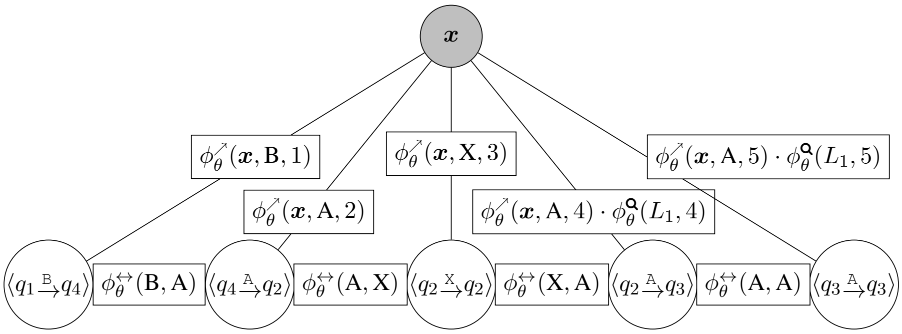
A common source of label interdependencies in sequence labeling is given by global constraints on how often a particular label occurs. Under such constraints, each label can directly depend on each



(a) A DFA for for Π. The path through this automaton for the string BAXAA is marked.

$$A = \{\langle q_1 \xrightarrow{\texttt{X}} q_1\rangle, \langle q_1 \xrightarrow{\texttt{A}} q_2\rangle, \langle q_1 \xrightarrow{\texttt{B}} q_4\rangle, \langle q_2 \xrightarrow{\texttt{X}} q_2\rangle, \langle q_2 \xrightarrow{\texttt{A}} q_3\rangle, \langle q_2 \xrightarrow{\texttt{B}} q_4\rangle, \langle q_3 \xrightarrow{\texttt{X}} q_2\rangle,$$
$$\langle q_3 \xrightarrow{\texttt{A}} q_3\rangle, \langle q_3 \xrightarrow{\texttt{B}} q_4\rangle, \langle q_4 \xrightarrow{\texttt{A}} q_2\rangle, \langle q_4 \xrightarrow{\texttt{X}} q_4\rangle, \langle q_4 \xrightarrow{\texttt{B}} q_5\rangle, \langle q_4 \xrightarrow{\texttt{A}} q_2\rangle, \langle q_4 \xrightarrow{\texttt{X}} q_3\rangle, \langle q_4 \xrightarrow{\texttt{B}} q_4\rangle\}$$

(b) A, the set of arcs in Π, which will be used as the label set for the auxiliary CRF.


$\boldsymbol{x}$
$\phi_\theta^{\nearrow}(\boldsymbol{x}, \text{B}, 1)$
$\phi_\theta^{\nearrow}(\boldsymbol{x}, \text{X}, 3)$
$\phi_\theta^{\nearrow}(\boldsymbol{x}, \text{A}, 5) \cdot \phi_\theta^{\mathsf{Q}}(L_1, 5)$
$\phi_\theta^{\nearrow}(\boldsymbol{x}, \text{A}, 2)$
$\phi_\theta^{\nearrow}(\boldsymbol{x}, \text{A}, 4) \cdot \phi_\theta^{\mathsf{Q}}(L_1, 4)$
$\langle q_1 \xrightarrow{\texttt{B}} q_4\rangle$
$\phi_\theta^{\leftrightarrow}(\text{B}, \text{A})$
$\langle q_4 \xrightarrow{\texttt{A}} q_2\rangle$
$\phi_\theta^{\leftrightarrow}(\text{A}, \text{X})$
$\langle q_2 \xrightarrow{\texttt{X}} q_2\rangle$
$\phi_\theta^{\leftrightarrow}(\text{X}, \text{A})$
$\langle q_2 \xrightarrow{\texttt{A}} q_3\rangle$
$\phi_\theta^{\leftrightarrow}(\text{A}, \text{A})$
$\langle q_3 \xrightarrow{\texttt{A}} q_3\rangle$

(c) The auxiliary CRF calculating the probability for the arc sequence corresponding to $\boldsymbol{y}$'s path through Π. Since q_3 corresponds to an accepting state for L_1, the emission function incorporates the pattern potential for L_1 at time steps which end on q_3. The resulting probability equals the RPCRF probability for the string $\boldsymbol{y}$.

Figure 2: A worked example for the label string $\boldsymbol{y}$ = BAXAA of an RPCRF with two patterns: L_1 = AX*A and L_2 = BX*B. (a) shows Π, the state-labeled automaton we obtain from these two languages, (b) shows the set of arcs in Π, which will be tags for our auxiliary CRF, and (c) demonstrates how we use our auxiliary CRF to calculate a probability for $\boldsymbol{y}$.

other label. For example, if we know that a particular label must occur exactly once in a sequence, assigning that label to any particular position affects the marginal distribution of every other position. These constraints may be soft, though – for example, in the classification of daily activities from a smartwatch data sequence, users typically go running once a day, but might run twice, or not at all (Kwon and Choi, 2018).

In order to investigate an RPCRF's ability to model such cardinality constraints, we construct a synthetic dataset of $(\boldsymbol{x}, \boldsymbol{y})$ pairs. For each pair, $\boldsymbol{x}$ consists of a single non-zero digit k, followed by nine zeros. The first label of $\boldsymbol{y}$ is always _, and, of the remaining nine labels, exactly k are A, with all others being _. We chose the value of k uniformly randomly, and then uniformly randomly select which k positions should be labeled as A.

Table 1: Example for Experiment 1 (cardinality patterns). The first token of each input specifies the number of As in the output.

$\boldsymbol{x}$	3000000000	9000000000	1000000000
$\boldsymbol{y}$	__A_AA____	_AAAAAAAAA	_____A____

As patterns, we use a set of nine regular lan-

Table 2: Results for Experiment 1 (EM acc. = Exact-match accuracy; Opt. str. = optimal strategy).

Model	EM acc. (%)	% Opt. str.
Optimal strategy	14.64	–
LSTM+CRF	11.27	76.98
LSTM+RPCRF	14.61	99.80

guages $\mathbb{L} = \{L_1, \cdots, L_9\}$:

$$L_k = \mathtt{\hat{}}(\_^*\mathtt{a})^k\_^*\mathtt{\$} \tag{7}$$

Each L_k matches label sequences with exactly k occurrences of A. As pattern can match only a complete label sequence, and as the languages are disjoint, only one pattern can match any given label sequence. An RPCRF should be able to learn from the first token of the input sequence which pattern should apply to the label sequence, and assign only that pattern a high weight with its pattern potential function, resulting in the model always predicting the correct number of As. Conversely, while a CRF can learn that the A label should be more or less likely depending on the value of k, it has no mechanism for enforcing a specific number of A labels (except in the case for $k = 9$, wherein the output is deterministic).

Table 1 gives examples of some datapoints for this experiment. Table 2 summarizes the performance of RPCRF and linear-chain CRFs on this task. We see that an RPCRF is able to achieve near-optimal accuracy. On the other hand, the linear-chain CRF, unable to directly enforce cardinality constraints, can only achieve approximately 77% of the optimal strategy's accuracy.

3.2 Experiment 2: Agreement patterns

Commonly for sequence labeling tasks, the presence of one type of label in a sequence might be highly informative about the presence or absence of other labels at distant positions in the sequence. For instance, when using sequence labeling to label named entities in text, an entity of type EVENT may be likely to occur in the same document as an entity of type DATE, while there may be no such affinity between entities of types LAW and WORK_OF_ART. In the extreme case, certain labels might be guaranteed to co-occur in a document, or alternatively forbidden from doing so.

To investigate an RPCRF's ability to learn such interactions, we construct a synthetic sequence-labeling dataset which exhibits strong agreement interactions between distant labels. In each $(\boldsymbol{x}, \boldsymbol{y})$ pair, $\boldsymbol{x}$ is a length-ten sequence containing eight zeros and exactly two ones, which represent entities to be labeled. The corresponding $\boldsymbol{y}$ assigns a _ label to all zeros, and a letter from A to F to the two ones. Importantly, these letter labels are selected such that A must co-occur with a B, C with a D, and E with an F. Table 3 provides some example $(\boldsymbol{x}, \boldsymbol{y})$-pairs for this experiment.

Table 3: Example for Experiment 2 (agreement patterns): model must learn which pairs of non-zero output labels correspond (A/B, C/D, E/F).

$\boldsymbol{x}$	0010000100	0011000000	0001000001
$\boldsymbol{y}$	__A____B__	__DC______	___F_____E

We assume a setting where model users know that *some* co-occurrence constraints exist, but do not know the particular letters which can or cannot co-occur. Thus, as patterns, we use a set of $\binom{6}{2} = 15$ languages, with each language matching a label sequence containing two distinct labels exactly once:

$$\mathbb{L} = \Big\{ \mathtt{\hat{}}\_^*\left(\alpha\_^*\beta \mid \beta\_^*\alpha\right)\_^*\mathtt{\$} : \{\alpha, \beta\} \subseteq \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}\}, \alpha \neq \beta \Big\} \tag{8}$$

Our model is thus responsible for learning which label pairs agree and disagree with one another.

Table 4 shows the results on this experiment for an RPCRF and for a linear-chain CRF baseline. As before, our RPCRF-based model achieves nearly optimal performance, while the linear-chain CRF, unable to learn the relationships between distant labels, lags significantly behind. Interestingly, the linear-chain CRF is able to model agreement in *some* cases – namely when the two entities happen to be directly adjacent Due to this, it performs better than the $\frac{1}{36}$ odds we would expect from having it label the two entities independently, but fails in cases where the entities are distant from one another.

3.3 Experiment 3: Battleship

While this paper has thus-far focused largely on CRFs with a linear-chain structure, CRFs are also commonly used for 2-dimensional data in tasks such as image segmentation (Chen et al., 2017). In such a setting, instead of labeling elements of a sequence, individual pixels or grid cells are labeled. Crucially, such a setting usually envisions

Table 4: Results for Experiment 2 on agreement patterns (EM acc. = Exact-match accuracy; Opt. str. = optimal strategy).

Model	EM acc. (%)	% Opt. str.
Optimal strategy	16.67	–
LSTM+CRF	6.97	41.81
LSTM+RPCRF	16.60	99.58

each pixel as directly adjacent to all four of its orthogonal neighbors, leading to a highly cyclic graph structure not amenable to tractable exact inference (Murphy et al., 1999).

With appropriate encoding and patterns, RPCRFs can also be used for labeling such 2-dimensional data. Any 2-dimensional grid can be serialized row-by-row into a linear sequence. Cells which neighbored horizontally in the original grid are still neighbors in the sequence, while vertical neighbors are now separated by from one another by a constant distance equal to the grid width. By writing patterns that are specifically sensitive to labels separated by exactly this distance, we can enable an RPCRF to model interactions between vertically adjacent cells in our original grid.

We demonstrate this concretely with a synthetic task on a 5×5 grid. Somewhere on this grid, a 4×1 battleship is hiding, positioned and oriented randomly. The input sequence $\boldsymbol{x}$ comprises all zeros, except for a single one, at some randomly-chosen cell of the battleship. In the label sequence $\boldsymbol{y}$, each cell occupied by the battleship is labeled A, while all other cells are labeled _. The model's task is thus to guess the position and location of the battleship, given only a single "hit."

Table 5 illustrates some input-output pairs. We use a single pattern, sensitive to two As separated by four _s (i.e., vertically adjacent in the grid):

$$\mathbb{L} = \{\texttt{A\_\_\_\_A}\} \tag{9}$$

This allows RPCRF to be sensitive to vertically adjacent pairs of As in the label sequence (at least when all intervening labels are instances of _).

Table 6 reports the performance of our two models. In this case, the RPCRF-based model does not achieve the performance of the optimal strategy here. This is due to a limitation in the pattern used: while the model can use its pattern to ensure the predicted $\boldsymbol{A}$s are adjacent, it has no mechanism for ensuring that it predicts the correct *number* of $\boldsymbol{A}$s. Nonetheless, even though the provided pattern set cannot capture all structural properties of the label sequences, we still see significant improvements over a linear-chain CRF.

4 Related Work

Our proposed approach is one of many ways for extending a linear-chain CRF in a manner that selectively circumvents the Markov assumption of default CRFs. Here we will briefly discuss some alternate formalisms for defining and working with such 'higher-order' CRFs.

Pattern-based CRFs. A conceptually similar approach to our current proposal are pattern-based CRFs (Ye et al., 2009; Takhanov and Kolmogorov, 2013). As with our regular-pattern-sensitive CRFs, pattern-based CRFs allow practitioners to specify a set of label patterns, allowing the CRF to learn long-distance dependencies by either encouraging or discouraging the presence of these patterns at particular locations of the label sequence. However, the patterns in pattern-based CRFs are limited to exact string matches, while our RPCRFs allow for arbitrary regular-expression patterns. Critically, a pattern-based CRF can only model dependencies as distant as its longest search pattern, while RPCRFs can easily be designed to learn dependencies over arbitrary distances, as our Experiment 1 demonstrated.

Semi-Markov CRFs. Another approach commonly used for allowing CRFs to learn non-local label interactions are semi-Markov CRFs (Sarawagi and Cohen, 2004). Under this formalism, rather than labeling each individual token, a semi-Markov CRF outputs a segmentation of the input, labeling each segment. While segment labels must follow the Markov assumption (each segment's label depends directly only on its neighboring segments), the model's behavior *within* each segment may be non-Markovian. Such models offer an efficient approach to modeling certain types of nonlocal interactions, but these interactions are limited to occurring within the same segment, again in contrast to our model.

Skip-chain CRFs. A skip-chain CRFs (Sutton and McCallum, 2007) is an otherwise linear-chain CRF augmented with *skip-connections*, a number of connections directly connecting otherwise distant labels in the sequence. The exact structure

Table 5: Example for Experiment 3 (battleship). Each input marks a single cell of the battleship, while the output marks all of its cells. Inputs/outputs are shown as 5×5 grids here but are treated as length-25 sequences by models.

$\boldsymbol{x}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\boldsymbol{y}$	_	_	_	A	_	_	_	_	_	_	_	_	_	_	_
	_	_	_	A	_	_	_	_	_	_	A	_	_	_	_
	_	_	_	A	_	A	A	A	A	_	A	_	_	_	_
	_	_	_	A	_	_	_	_	_	_	A	_	_	_	_
	_	_	_	_	_	_	_	_	_	_	A	_	_	_	_

Table 6: Results for Experiment 3, Battleship (EM acc. = Exact-match accuracy; Opt. str. = optimal strategy).

Model	EM acc. (%)	% Opt. str.
Optimal strategy	31.25	–
LSTM+CRF	2.50	8.00
LSTM+RPCRF	12.49	39.98

of these skip connections can be specified according to the task, and may even be specified conditioned on the input sequence. This provides a conceptually straightforward way to enable linear-chain CRFs to model long-distance dependencies. While skip connections can be selected to account for many possible types of long-distance interactions, the resulting graphs are highly cyclic, and often require approximate techniques for parameter estimation and inference. Nonetheless, with certain connection structures, tricks are possible to allow for exact training and inference on skip-chain CRFs (Galley, 2006).

Regular-constrained CRFs. Regular-constrained CRFs (Papay et al., 2022) enforce that a model's output sequence *must* match some user-specified regular expression. While this enables linear-chain CRFs to respect non-local label interactions, our proposal allows a CRF to learn the likelihood of regular expressions matching at different positions in the label sequence. Thus, a regular-constrained CRF can be understood as a special case of a RPCRF with a single pattern (the complement of the user-specified language) given a constant potential of zero. While regular-constrained CRFs are limited to enforcing constraints known a priori, our regular-pattern-sensitive CRFs can *learn* when different label patterns are likely or unlikely.

5 Conclusions

This paper introduced regular-pattern-sensitive CRFs, a method for enriching linear-chain CRFs with the ability to learn long-distance interactions which occur within user-specified regular-expression patterns. By representing all patterns in a single state-labeled DFA, and using an auxiliary CRF to represent a distribution over paths through this DFA, we can selectively extend CRFs with non-local features while preserving efficient parameter learning and inference.

Regular patterns are often sufficient to model the relevant structures in the domain, as Experiment 2 illustrates. More complex structures can often be rewritten with regular patterns by assuming a maximum input length (cf. (Mohri and Nederhof, 2001) and Experiment 1). Even when regular-language patterns cannot fully capture the dependency structure of the labels, and imperfect approximation can still yield a substantial improvement, as we found in Experiment 3.

Regular patterns offer a flexible and powerful tool for incorporating domain knowledge into sequence classification models that combine the knowledge-based and data-driven paradigms in a promising fashion. Sequence labeling models can be made to account for specific tasks' output structures by simply specifying regular-expression patterns, without the need to explicitly construct an FST or otherwise adapt the model architecture.

A promising direction for future work lies in the combination of RPCRFs with LLM encoders. The strengths of these two paradigms could prove complementary, and LLMs with RPCRF output layers may make good models for structured prediction tasks such as relation extraction or semantic role labeling, where it is necessary to model both linguistic interactions in the input as well as structural interactions in the output.

Limitations

While training and inference time for RPCRFs are quadratic in the number of arcs in the underlying automaton, this number is worst-case exponential in the number of patterns, limiting our model's use with some large sets of patterns. While some combinations of patterns synergize and yield small automata, we do not have a formal characterization of which combinations of patterns lead to tractable models.

References

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Grammatical Inference: Algorithms and Applications*, pages 15–24, Berlin, Heidelberg. Springer Berlin Heidelberg.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Alebachew Chiche and Betselot Yitagesu. 2022. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9(1):10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Asmita, Ryan Tsang, Najmeh Nazari, Han Wang, and Houman Homayoun. 2024. Large language models for code analysis: do llms really do their job? In *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, USA. USENIX Association.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 364–372, Sydney, Australia. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA.

Min-Cheol Kwon and Sunwoong Choi. 2018. Recognition of daily human activity using an artificial neural network and smartwatch. *Wireless Communications and Mobile Computing*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Jinxin Liu, Shulin Cao, Jiaxin Shi, Tingjian Zhang, Lunyiu Nie, Linmei Hu, Lei Hou, and Juanzi Li. 2024. How proficient are large language models in formal languages? an in-depth insight for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 792–815, Bangkok, Thailand. Association for Computational Linguistics.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Mehryar Mohri and Mark-Jan Nederhof. 2001. Regular approximation of context-free grammars through transformation. In *Robustness in language and speech technology*, pages 153–163. Springer.

Zhalgas Mukanov and Rustem Takhanov. 2022. Learning the pattern-based CRF for prediction of a protein local structure. *Informatica*, 46(6).

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, page 467–475, Stockholm, Sweden. Morgan Kaufmann Publishers Inc.

Sean Papay, Roman Klinger, and Sebastian Pado. 2022. Constraining linear-chain CRFs to regular languages. In *International Conference on Learning Representations*.

Deepashree Raje and P. P. Mujumdar. 2009. A conditional random field–based downscaling method for assessment of climate change impact on multisite daily precipitation in the mahanadi basin. *Water Resources Research*, 45(10).

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, pages 623–633, San Diego, California. Association for Computational Linguistics.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Christian Scheible, Roman Klinger, and Sebastian Padó. 2016. Model architectures for quotation detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745, Berlin, Germany. Association for Computational Linguistics.

Helmut Schmid. 1994. Part-of-speech tagging with neural networks. In *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Charles Sutton and Andrew McCallum. 2007. An introduction to conditional random fields for relational learning.

Rustem Takhanov and Vladimir Kolmogorov. 2013. Inference algorithms for pattern-based crfs on sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 145–153.

Sheng Wang, Jian Peng, Jianzhu Ma, and Jinbo Xu. 2016. Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6(1):1–11.

Nan Ye, Wee Lee, Hai Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In *Proceedings of Advances in Neural Information Processing Systems*. Curran Associates, Inc.