CONSTRAINING LINEAR-CHAIN CRFS TO REGULAR LANGUAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

A major challenge in structured prediction is to represent the interdependencies within output structures. When outputs are structured as sequences, linear-chain conditional random fields (CRFs) are a widely used model class which can learn *local* dependencies in the output. However, the CRF's Markov assumption makes it impossible for CRFs to represent distributions with *nonlocal* dependencies, and standard CRFs are unable to respect nonlocal constraints of the data (such as global arity constraints on output labels). We present a generalization of CRFs that can enforce a broad class of constraints, including nonlocal ones, by specifying the space of possible output structures as a regular language $\mathcal{L}$. The resulting regular-constrained CRF (RegCCRF) has the same formal properties as a standard CRF, but assigns zero probability to all label sequences not in $\mathcal{L}$. Notably, RegCCRFs can incorporate their constraints during training, while related models only enforce constraints during decoding. We prove that constrained training is never worse than constrained decoding, and show empirically that it can be substantially better in practice. Finally, we demonstrate a practical benefit on downstream tasks by incorporating a RegCCRF into a deep neural model for semantic role labeling, exceeding state-of-the-art results on a standard dataset.

1 INTRODUCTION

Structured prediction is a field of machine learning where outputs are expected to obey some predefined discrete structure. Instances of structured prediction with various output structures occur in many applications, including computer vision (e.g., scene graph generation (Johnson et al., 2015) with graph-structured output), natural language processing (e.g., linguistic parsing (Niculae et al., 2018) with tree-structured output, relation extraction (Roth & Yih, 2004) with tuple-structured output) or modeling the spatial structure of physical entities and processes (Jiang, 2020).

A key difficulty faced by all models is to tractably model interdependencies between different parts of the output. As output spaces tend to be combinatorially large, special techniques, approximations, and independence assumptions must be used to work with the associated probability distributions. Considerable research has investigated specific structures for which such approaches make machine learning tractable. For instance, when outputs are trees over a fixed set of nodes, maximal arborescence algorithms allow for exact inference (Chu, 1965; Edmonds, 1967); when outputs are graph-structured, loopy belief propagation can provide approximate inference (Murphy et al., 1999).

If the output forms a linear sequence, a Markov assumption is often made: model outputs depend only on their immediate neighbors, but not (directly) on more distant ones. A common model that uses this assumption is the linear-chain conditional random field (CRF) (Lafferty et al., 2001), which has found ubiquitous use for sequence labeling tasks, including part-of-speech tagging (Gimpel et al., 2011) and named entity recognition (Lample et al., 2016). This model became popular with the use of hand-crafted feature vectors in the 2000s, and is nowadays commonly used as an output layer in neural networks to encourage the learning of structural properties of the output sequence. The Markov assumption makes training tractable, but also limits the CRF's expressive power, which can hamper performance, especially for long sequences (Scheible et al., 2016). Semi-Markov CRFs (Sarawagi & Cohen, 2004) and skip-chain CRFs (Sutton & McCallum, 2004) are techniques for relaxing the Markov assumption, but both come with drawbacks in performance and expressiveness.

In this work, we propose a new method to tractably relax the Markov assumption in CRFs. Specifically, we show how to constrain the output of a CRF to *a regular language*, such that the resulting *regular-constrained CRF (RegCCRF)* is guaranteed to output label sequences from that language. Since regular languages can encode long-distance dependencies between the symbols in their strings, RegCCRFs provide a simple model for structured prediction guaranteed to respect these constraints. The domain knowledge specifying these constraints is defined via regular expressions, a straightforward, well understood formalism. We show that our method is distinct from approaches that enforce constraints at decoding time, and that it better approximates the true data distribution. We evaluate our model empirically as the output layer of a neural network and attain state-of-the-art performance for semantic role labeling (Weischedel et al., 2011; Pradhan et al., 2012). Our PyTorch RegCCRF implementation can be used as a drop-in replacement for standard CRFs.

2 Related work

We identify three areas of structured prediction that are relevant to our work: constrained decoding, which can enforce output constraints at decoding time, techniques for weakening the Markov assumption of CRFs to learn long-distance dependencies, and weight-learning in finite-state transducers.

Constrained decoding. A common approach to enforcing constraints in model output is *constrained decoding*: Models are trained in a standard fashion, and decoding ensures that the model output satisfies the constraints. This almost always corresponds to finding or approximating a version of the model's distribution conditionalized on the output obeying the specified constraints. This approach is useful if constraints are not available at training time, such as in the interactive information extraction task of Kristjansson et al. (2004). They present *constrained conditional random fields*, which can enforce that particular tokens are or are not assigned particular labels (positive and negative constraints, respectively). Formally, our work is a strict generalization of this approach, as position-wise constraints can be formulated as a regular language, but regular languages go beyond position-wise constraints. Other studies treat decoding with constraints as a search problem, searching for the most probable decoding path which satisfies all constraints. For example, He et al. (2017) train a neural network to predict token-wise output probabilities for semantic role labeling following the BIO label-alphabet (Ramshaw & Marcus, 1999), and then use exact A* search to ensure that the output forms a valid BIO sequence and that particular task-specific constraints are satisfied. For autoregressive models, constrained beam search (Hokamp & Liu, 2017; Anderson et al., 2017; Hasler et al., 2018) can enforce regular-language constraints during search. We further discuss constrained decoding as it relates to RegCCRFs in Section 5.

Markov relaxations. While our approach can relax the Markov assumption of CRFs through nonlocal hard constraints, another strand of work has developed models which can directly *learn* nonlocal dependencies in CRFs: *Semi-Markov CRFs* (Sarawagi & Cohen, 2004) relax the Markov property to the semi-Markov property. In this setting, CRFs are tasked with segmentation, where individual segments may depend only on their immediate neighbors, but model behavior within a particular segment need not be Markovian. As such, semi-Markov CRFs are capable of capturing nonlocal dependencies between output variables, but only to a range of one segment and inside of a segment. *Skip-chain CRFs* (Sutton & McCallum, 2004) change the expressiveness of CRFs by relaxing the assumption that only the linear structure of the input matters: they add explicit dependencies between distant nodes in an otherwise linear-chain CRF. These dependencies are picked based on particular properties, e.g., input variables of the same value or which share other properties. In doing so, they add loops to the model's factor graph, which makes exact training and inference intractable, and leads to the use of approximation techniques such as loopy belief propagation and Gibbs sampling.

Weight learning for finite-state transducers. While our approach focuses on the task of constraining the CRF distribution to a known regular language, a related task is that of learning a weighted regular language from data. This task is usually formalized as learning the weights of a weighted finite-state transducer (FST), as in e.g. Eisner (2002) with directly parameterized weights and Rastogi et al. (2016) with weights parameterized by a neural network. Despite the difference in task-setting, this task is quite similar to ours in the formal sense, and in fact our proposal can be

viewed as a particularly well-behaved special case of FST weight learning for an appropriately chosen transducer architecture and parameterization. We discuss this connection further in Section 4.3.

3 PRELIMINARIES AND NOTATION

As our construction involves finite-state automata and conditional random fields, we define these here and specify the notation we will use in the remainder of this work.

Finite-state automata. All automata are taken to be nondeterministic finite-state automata (NFAs) without epsilon transitions. Let such an NFA be defined as a 5-tuple (Σ, Q, q_1, F, E), where $\Sigma = \{a_1, a_2, ..., a_{|\Sigma|}\}$ is a finite alphabet of symbols, $Q = \{q_1, q_2, ..., q_{|Q|}\}$ is a finite set of states, $q_1 \in Q$ is the sole starting state, $F \subseteq Q$ is a set of accepting states, and $E \subseteq Q \times \Sigma \times Q$ is a set of directed, symbol-labeled edges between states. The edges define the NFA's transition function $\Delta : Q \times \Sigma \to 2^Q$, with $r \in \Delta(q, a) \leftrightarrow (q, a, r) \in E$. An automaton is said to accept a string $\boldsymbol{s} \in \Sigma^*$ iff there exists a contiguous path of edges from q_1 to some accepting state whose edge labels are exactly the symbols of $\boldsymbol{s}$. The *language* defined by an automaton is the set of all such accepted strings. A language is *regular* if and only if it is the language of some NFA.

Linear-chain conditional random fields. Linear-chain conditional random fields (CRFs) Lafferty et al. (2001) are a sequence labeling model. Parameterized by θ, they use a global exponential model to represent the conditional distribution over label sequences $\boldsymbol{y} = \langle y_1, y_2, ..., y_t \rangle$ conditioned on input sequences $\boldsymbol{x} = \langle x_1, x_2, ..., x_t \rangle$: $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) \propto \exp \sum_j f_\theta^j(\boldsymbol{x}, \boldsymbol{y})$, with individual observations x_i coming from some observation space X, and outputs y_i coming from some finite alphabet Y. In this work, we use CRFs for sequence labeling problems, but the dataset labels do not correspond directly to the CRF's outputs y_i. In order to avoid ambiguity, and since the term "state" already has a meaning for NFAs, we call $\boldsymbol{y}$ the CRF's *tag sequence*, and each y_i a *tag*. The terms *label sequence* and *label* will thus unambiguously refer to the original dataset labels.

Each f_θ^j is a potential function of $\boldsymbol{x}$ and $\boldsymbol{y}$, parameterized by θ. Importantly, in a linear-chain CRF, these potential functions are limited to two kinds: The *transition function* $g_\theta(y_i, y_{i+1})$ assigns a potential to each pair (y_i, y_{i+1}) of adjacent tags in $\boldsymbol{y}$, and the *emission function* $h_\theta(y_i \mid \boldsymbol{x}, i)$ assigns a potential to each possible output tag y_i given the observation sequence $\boldsymbol{x}$ and its position i. With these, the distribution defined by a CRF is

$$P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) \propto \exp\left(\sum_{i=1}^{t-1} g_\theta(y_i, y_{i+1}) + \sum_{i=1}^{t} h_\theta(\boldsymbol{x}, y_i, i)\right). \tag{1}$$

Limiting our potential functions in this way imposes a Markov assumption on our model, as potential functions can only depend on a single tag or a single pair of adjacent tags. This makes learning and inference tractable: the forward algorithm (Jurafsky & Martin, 2009) can calculate negative log-likelihood (NLL) loss during training, and the Viterbi algorithm (Viterbi, 1967; Jurafsky & Martin, 2009) can be used for inference. Both are linear in t, and quadratic in $|Y|$ in both time and space.

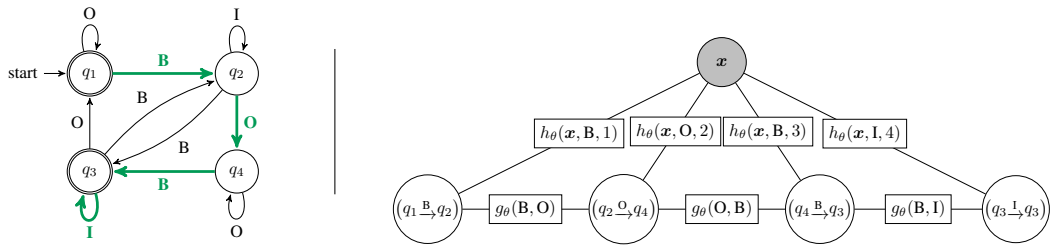
4 REGULAR-CONSTRAINED CRFS

Given a regular language $\mathcal{L}$, we would like to constrain a CRF to $\mathcal{L}$. We formalize this notion of constraint with conditional probabilities – a CRF constrained to $\mathcal{L}$ is described by a (further) conditionalized version of that CRF's distribution $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$, conditioned on the event that the tag sequence $\boldsymbol{y}$ is in $\mathcal{L}$. We write this distribution as

$$P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L}) = \begin{cases} \alpha \cdot P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) & \text{if } \boldsymbol{y} \in \mathcal{L} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

with $\alpha \geq 1$ defined as $\alpha^{-1} = \sum_{\boldsymbol{y} \in \mathcal{L}} P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$.

In order to utilize this distribution for machine learning, we need to be able to compute NLL losses and perform MAP inference. As discussed in Section 3, both of these are efficiently computable for CRFs. Thus, if we can construct a separate CRF whose output distribution can be interpreted as $P(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$, both of these operations will be available. We do this in the next section.



$$Y' = \left\{ \left(q_1 \xrightarrow{\mathrm{O}} q_1\right), \left(q_1 \xrightarrow{\mathrm{B}} q_2\right), \left(q_2 \xrightarrow{\mathrm{I}} q_2\right), \left(q_2 \xrightarrow{\mathrm{B}} q_3\right), \left(q_2 \xrightarrow{\mathrm{O}} q_4\right), \left(q_3 \xrightarrow{\mathrm{O}} q_1\right), \left(q_3 \xrightarrow{\mathrm{B}} q_2\right), \left(q_3 \xrightarrow{\mathrm{I}} q_3\right), \left(q_4 \xrightarrow{\mathrm{B}} q_3\right), \left(q_4 \xrightarrow{\mathrm{O}} q_4\right) \right\}$$

Figure 1: Example for a RegCCRF, showing NFA and unrolled factor graph. $\mathcal{L}$ describes the language $(\mathrm{O} \mid \mathrm{BI}^*\mathrm{O}^*\mathrm{BI}^*)^*$, the language of valid BIO sequences for an even number of spans. We would like to calculate $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ for $\boldsymbol{y} = \langle \mathrm{B}, \mathrm{O}, \mathrm{B}, \mathrm{I} \rangle$. We show an unambiguous automaton M for $\mathcal{L}$ (left), and a factor graph (right) for the auxiliary CRF computing $P_\theta(\boldsymbol{y'} \mid \boldsymbol{x})$, where $\boldsymbol{y'} \in Y'^*$ corresponds to the sole accepting path of $\boldsymbol{y}$ through M (marked).

4.1 Construction

Let $M := (\Sigma, Q, F, E)$ be an NFA that describes $\mathcal{L}$. We assume that M is *unambiguous* – i.e., every string in $\mathcal{L}$ is accepted by exactly one path through M. As every NFA can be transformed into an equivalent unambiguous NFA (Mohri, 2012), this assumption involves no loss of generality. Our plan is to represent $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ by constructing a separate CRF with a distinct tag set, whose output sequences can be interpreted directly as paths through M. As M is unambiguous, each label sequence in $\mathcal{L}$ corresponds to exactly one such path. We parameterize this auxiliary CRF identically to our original CRF – that is, with label-wise (not tag-wise) transition and emission functions. Thus, for all parameterizations θ, both distributions $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ and $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ are well defined.

There are many ways to construct such a CRF. As CRF training and inference are quadratic in the size of the tag set Y, we would prefer a construction which minimizes $|Y|$. However, for clarity, we will first present a conceptually simple construction, and discuss approaches to reduce $|Y|$ in Section 4.2. We start with our original CRF, parameterized by θ, with tag set $Y = \Sigma$, transition function g_θ, and emission function h_θ, describing the probability distribution $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$, $\boldsymbol{y} \in \Sigma^*$. From this, we construct a new CRF, also parameterized by the same θ, but with tag set Y', transition function g'_θ, and emission function h'_θ. This auxiliary CRF describes the distribution $P'_\theta(\boldsymbol{y'} \mid \boldsymbol{x})$ (with $\boldsymbol{y'} \in Y'^*$), which we will be able to interpret as $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$. The construction is as follows:

$$Y' = E \tag{3}$$

$$g'_\theta((q, a, r), (q', a', r')) = \begin{cases} g_\theta(a, a') & \text{if } r = q' \\ -\infty & \text{otherwise} \end{cases} \tag{4}$$

$$h'_\theta(\boldsymbol{x}, (q, a, r), i) = \begin{cases} -\infty & \text{if } i = 1, q \neq q_1 \\ -\infty & \text{if } i = t, r \notin F \\ h_\theta(\boldsymbol{x}, a, i) & \text{otherwise.} \end{cases} \tag{5}$$

This means that the tags of our new CRF are the edges of M, the transition function assigns zero probability to transitions between edges which do not pass through a shared NFA state, and the emission function assigns zero probability to tag sequences which do not begin at the starting state or end at an accepting state. Apart from these constraints, the transition and emission functions depend only on edge labels, and not on the edges themselves, and agree with the standard CRF's g_θ and h_θ when no constraints are violated.

As M is unambiguous, every tag sequence $\boldsymbol{y}$ corresponds to a single path through M, representable as an edge sequence $\boldsymbol{\pi} = \langle \pi_1, \pi_2, ..., \pi_t \rangle, \pi_i \in E$. Since this path is a tag sequence for our auxiliary CRF, we can directly calculate the auxiliary CRF's $P'_\theta(\boldsymbol{\pi} \mid \boldsymbol{x})$. From the construction of g'_θ and h'_θ, this must be equal to $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$, as it is proportional to $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ for $\boldsymbol{y} \in \mathcal{L}$, and zero (or, more correctly, undefined) otherwise. Figure 1 illustrates this construction with a concrete example.

4.2 TIME AND SPACE EFFICIENCY

As the Viterbi and forward algorithms are quadratic in $|Y|$, very large tag sets can lead to performance issues, possibly making training or inference intractible in extreme cases. Thus, we would like to characterize under which conditions a RegCCRF can be used tractibly, and identify techniques for improving performance. As Y corresponds to the edges of M, we would like to select our unambiguous automaton M to have as few edges as possible. For arbitrary languages, this problem is NP-complete (Jiang & Ravikumar, 1991), and, assuming P $\neq$ NP, is not even efficiently approximable (Gruber & Holzer, 2007). Nonetheless, for many common classes of languages, there exist approaches to obtain a tractably small automaton.

One straightforward method is to construct M directly from a short unambiguous regular expression. Brüggemann-Klein & Wood (1992) present a simple algorithm for constructing an unambiguous automaton from an unambiguous regular expression, with $|Q|$ linear in the length of the expression. Using this method to construct M, the time- and space-complexity of Viterbi are polynomial in the length of our regular expression, with a worst-case of quartic complexity when the connectivity graph of M is dense.

For many other tasks, a reasonable approach is to leverage domain knowledge about the constraints to manually construct a small unambiguous automaton. For example, if the constraints require that a particular label occurs exactly n times in the output sequence, an automaton could be constructed manually to count ocurrences of that label. Multiple constraints of this type can then be composed via automaton union and intersection.

Without making changes to M, we can also reduce the size of $|Y|$ by adjusting our construction. Instead of making each edge of M a tag, we can adopt equivalence classes of edges. Reminiscent of standard NFA minimization, we define $(q, a, r) \sim (q', a', r') \leftrightarrow (q, a) = (q', a')$. When constructing our CRF, whenever a transition would have been allowed between two edges, we allow a transition between their corresponding equivalence classes. We do the same to determine which classes are allowed as initial or final tags. As each equivalence class corresponds (non-uniquely) to a single symbol a, we can translate between tag sequences and strings of $\mathcal{L}$ just as before.

4.3 INTERPRETATION AS A WEIGHTED FINITE-STATE TRANSDUCER

While we present our model as a variation of a standard CRF which enforces regular-language constraints, an alternate characterization is as a weighted finite-state transducer with the transducer topology and weight parameterization chosen so as to yield the distribution $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$. In order to accommodate CRF transition weights, such an approach involves weight-learning in an auxiliary automaton whose edges correspond to edge-pairs in M – we give a full construction in Appendix C.

This interpretation enables direct comparison to studies on weight learning in finite-state transducers, such as Rastogi et al. (2016). While RegCCRFs can be viewed as special cases of neural-weighted FSTs, they inherit a number of useful properties from CRFs not possessed by neural-weighted automata in general. Firstly, as $|\boldsymbol{y}|$ is necessarily equal to $|\boldsymbol{x}|$, the partition function $\sum_{\boldsymbol{y} \in \mathcal{L}} P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ is guaranteed to be finite, and $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ is a well-defined probability distribution for all θ, which is not true for weighted transducers in general, which may admit paths with unbounded lengths and weights. Secondly, as M is assumed to be unambiguous, string probabilities correspond exactly to path probabilities, allowing for exact MAP inference with the Viterbi algorithm. In contrast, finding the most probable string in the highly ambiguous automata used when learning edge weights for an unknown language is NP-Hard (Casacuberta & de la Higuera, 1999), necessitating approximation methods such as crunching (May & Knight, 2006). Finally, as each RegCCRF can be expressed as a CRF with a particular parameterization, the convexity guarantees of standard CRFs carry over, in that the loss is convex with respect to emission and transition scores. In contrast, training losses in general weighted finite-state transducers are usually nonconvex (Rastogi et al., 2016).

5 CONSTRAINED TRAINING VS. CONSTRAINED DECODING

Our construction suggests two possible use cases for a RegCCRF: *constrained decoding*, where a CRF is trained unconstrained, and the learned weights are then used in a RegCCRF at decoding time, and *constrained training*, where a RegCCRF is both trained and decoded with constraints. In

this section, we compare between these two approaches and a standard, *unconstrained CRF*. We assume a machine learning setting where we have access to samples from some data distribution $\widetilde{P}(\boldsymbol{x}, \boldsymbol{y})$, with each $\boldsymbol{x} \in X^*$, and each $\boldsymbol{y}$ of matching length in some regular language $\mathcal{L} \subseteq \Sigma^*$. We wish to model the conditional distribution $\widetilde{P}(\boldsymbol{y} \mid \boldsymbol{x})$ with either a CRF or a RegCCRF, by way of maximizing the model's (log) likelihood given the data distribution.

The unconstrained CRF corresponds to a CRF that has been trained, without constraints, on data from $\widetilde{P}(\boldsymbol{x}, \boldsymbol{y})$, and is used directly for inference: It makes no use of the language $\mathcal{L}$. The model's output distribution is $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x})$, with parameter vector θ_u minimizing the NLL objective:

$$\theta_u = \arg\min_{\theta} E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}} \left[-\ln P_\theta(\boldsymbol{y} \mid \boldsymbol{x})\right] \tag{6}$$

In constrained decoding, a CRF is trained unconstrained, but its weights are used in a RegCCRF at decoding time. The output distribution of such a model is $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$. It is parameterized by the same parameter vector θ_u as the unconstrained CRF, as the training procedure is identical, but the output distribution is conditioned on $\boldsymbol{y} \in \mathcal{L}$.

Constrained training involves directly optimizing a RegCCRF's output distribution, avoiding any asymmetry between training and decoding time. In this case, the output distribution of the model is $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$, where

$$\theta_c = \arg\min_{\theta} E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}} \left[-\ln P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})\right] \tag{7}$$

is the parameter vector which minimizes the NLL of the RegCCRF's constrained distribution.

These three approaches form a hierarchy in terms of their ability to match the data distribution: $L_{\text{unconstrained}} \geq L_{\text{constrained decoding}} \geq L_{\text{constrained training}}$, with each L corresponding to the negative log-likelihood assigned by each model to the data; see Appendix B for a proof. This suggests we should prefer the constrained training regimen.

6 Synthetic Data Experiments

While constrained training cannot underperform constrained decoding, the conditions where it is strictly better depend on exactly how the transition and emission functions are parameterized, and are not easily stated in general terms. We now empirically show two simple experiments on synthetic data where the two are not equivalent.

The procedure is similar for both experiments. We specify a regular language $\mathcal{L}$, an observation alphabet X, and a joint data distribution $\widetilde{P}(\boldsymbol{x}, \boldsymbol{y})$ over observation sequences in X^* and label sequences in $\mathcal{L}$. We then train two models, one with a RegCCRF, parameterized by θ_c, and one with an unconstrained CRF, parameterized by θ_u. For each model, we initialize parameters randomly, then use stochastic gradient descent to minimize NLL with $\widetilde{P}$. We directly generate samples from $\widetilde{P}$ to use as training data. After optimizing θ_c and θ_u, construct a RegCCRF with θ_u for use as a constrained-decoding model, and we compare the constrained-training distribution $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ with the constrained-decoding distribution $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$.

We use a simple architecture for our models, with both the transition functions g_θ and emission functions h_θ represented as parameter matrices. We list training hyperparameters in Appendix A.

6.1 Arbitrarily large differences in likelihood

We would like to demonstrate that, when comparing constrained training to constrained decoding in terms of likelihood, constrained training can outperform constrained decoding by an arbitrary margin. We choose $\mathcal{L} = (\texttt{ac})^* \mid (\texttt{bc})^*$ to make conditional independence particularly relevant – as every even-indexed label is $\texttt{c}$, an unconstrained CRF must model odd-indexed labels independently, while a constrained CRF can use its constraints to account for nonlocal dependencies. For simplicity, we hold the input sequence constant, with $X = \{\texttt{o}\}$.

Our approach is to construct sequences of various lengths. For $k \in \mathbb{N}$, we let our data distribution be

$$\widetilde{P}(\texttt{o}^{2k}, (\texttt{ac})^k) = \frac{3}{4} \text{ and } \widetilde{P}(\texttt{o}^{2k}, (\texttt{bc})^k) = \frac{1}{4}. \tag{8}$$

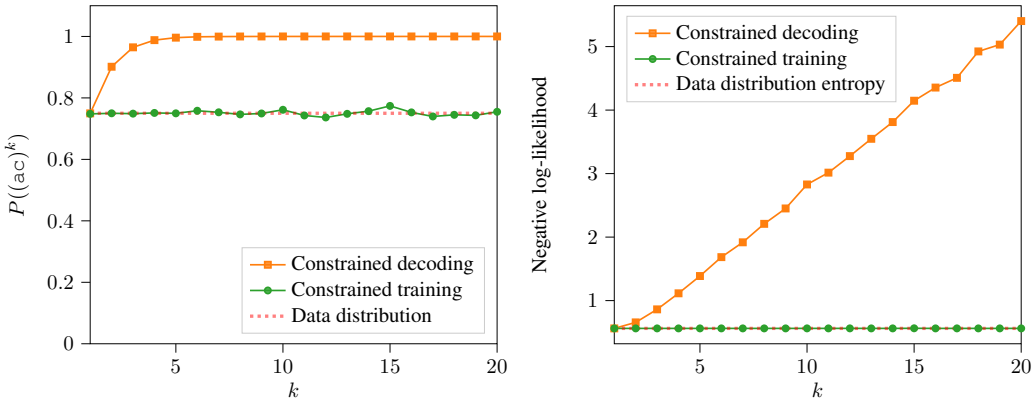


Figure 2: Model output probabilities, and NLL losses, plotted against sequence length k. As k increases, constrained decoding becomes a progressively worse approximation for the data distribution, while constrained training is consistently able to match the data distribution.

Table 1: Output distributions for constrained decoding ($P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$) and constrained training ($P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$), compared to the data distribution $\widetilde{P}(\boldsymbol{y} \mid \boldsymbol{x})$. Constrained decoding cannot learn the data distribution exactly, and yields a mode which disagrees with that of the data distribution.

$\boldsymbol{y}$	$\widetilde{P}(\boldsymbol{y} \mid \boldsymbol{x})$	$P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$	$P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$
acd	**0.4**	0.32	**0.40**
bcd	0.3	**0.48**	0.30
bce	0.3	0.20	0.30

As the marginal distributions for odd-indexed characters are not independent, an unconstrained CRF cannot exactly represent the distribution $\widetilde{P}$. We train and evaluate individual models for each sequence length k. Figure 2 plots model probabilities and NLL losses for various k. We see that, regardless of k, $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ is able to match $\widetilde{P}(\boldsymbol{y} \mid \boldsymbol{x})$ almost exactly, with only small deviations due to sampling noise in SGD. On the other hand, as sequence length increases, $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ becomes progressively "lop-sided", assigning almost all of its probability mass to the string $(\texttt{ac})^k$. This behavior is reflected in the models' likelihoods – constrained training stays at near-constant likelihood for all k, while the negative log-likelihood of constrained decoding grows linearly with k.

6.2 Differences in MAP inference

We show here that constrained training and constrained decoding can disagree about which label sequence they deem most likely. Furthermore, in this case, MAP inference agrees with the data distribution's mode for constrained training, but not for constrained decoding. To do this, we construct a fixed-length output language $\mathcal{L} = \texttt{acd} \mid \texttt{bcd} \mid \texttt{bce}$, where an unconstrained CRF is limited by the Markov property to predict $\boldsymbol{y}$'s prefix and suffix independently, and choose a data distribution which violates this independence assumption. We select our data distribution,

$$\widetilde{P}(\texttt{ooo}, \texttt{acd}) = 0.4 \text{ and } \widetilde{P}(\texttt{ooo}, \texttt{bcd}) = 0.3 \text{ and } \widetilde{P}(\texttt{ooo}, \texttt{bce}) = 0.3, \tag{9}$$

to be close to uniform, but with one label sequence holding the slight majority, and we ensure that the majority label sequence is *not* the label sequence with both the majority prefix and the majority suffix (i.e. bcd). As before, we hold the observation sequence as a constant (ooo). We train a constrained and an unconstrained CRF to convergence, and compare $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ to $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$.

Table 1 shows $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ and $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ as they compare to $\widetilde{P}(\boldsymbol{y} \mid \boldsymbol{x})$. We find that, while the mode of $\widetilde{P}(\boldsymbol{y} \mid \boldsymbol{x})$ is acd, with probability of 0.4, the mode of constrained decoding distribution $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ is bcd, the string with the majority prefix and the majority suffix, to which the model assigns a probability of 0.48. Conversely, the constrained training distribution $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ matches the data distribution almost exactly, and predicts the correct mode.

7 REAL-WORLD DATA EXPERIMENT: SEMANTIC ROLE LABELING

Task. As a final experiment, we apply our RegCCRF to the NLP task of semantic role labeling (SRL) in the popular PropBank framework (Palmer et al., 2005). In line with previous work, we adopt the *known-predicate setting*, where events are given and the task is to mark token spans as (types of) event participants. PropBank assumes 7 semantic *core roles* (ARG0 through ARG5 plus ARGA) plus 21 *non-core roles* for modifiers such as times or locations. For example, in [$_{\text{ARG0}}$ *Peter*] **saw** [$_{\text{ARG1}}$ *Paul*] [$_{\text{ARGM-TMP}}$ *yesterday*], the argument labels inform us who does the seeing (ARG0), who is seen (ARG1), and when the event took place (ARGM-TMP). In addition, role spans may be labeled as *continuations* of previous role spans, or as *references* to another role span in the sentence. SRL can be framed naturally as a sequence labeling task (He et al., 2017). However, the task comes with a number of hard constraints that are not automatically satisfied by standard CRFs, namely: (1) Each core role may occur at most once per event; (2) for continuations, the span type must occur previously in the sentence; (3) for references, the span type must occur elsewhere in the sentence.

Data. In line with previous work (Ouchi et al., 2018), we work with the OntoNotes corpus as used in the CoNLL 2012 shared task[1] (Weischedel et al., 2011; Pradhan et al., 2012), whose training set comprises 66 roles (7 core roles, 21 non-core roles, 19 continuation types, and 19 reference types).

RegCCRF Models. To encode the three constraints listed above in a RegCCRF, we define a regular language describing valid BIO sequences (Ramshaw & Marcus, 1999) over the 66 roles. A minimal unambiguous NFA for this language has more than $2^2 \cdot 3^{19}$ states, which is too large to run the Viterbi algorithm on our hardware. However, as many labels are very rare, we can shrink our automaton by discarding them at the cost of imperfect recall. We achieve further reductions in size by ignoring constraints on reference roles, treating them identically to non-core roles. Our final automaton recognizes 5 core role types (ARG0 through ARG4), 17 non-core / reference roles, and one continuation role type (for ARG1). This automaton has 672 states, yielding a RegCCRF with 2592 tags. A description of our procedure for constructing this automaton can be found in Appendix D.

Our model architecture is given by this RegCCRF, with emission scores provided by a linear projection of the output of a pretrained RoBERTa network Liu et al. (2019). In order to provide the model with event information, the given predicates are prefixed by a special reserved token in the input sequence. RoBERTa parameters are fine-tuned during the learning of transition scores and projection weights. We perform experiments with both constrained training and constrained decoding settings – we will refer to these as *ConstrTr* and *ConstrDec* respectively. A full description of the training procedure, including training times, is provided in Appendix A. As RegCCRF loss is only finite for label sequences in $\mathcal{L}$, we must ensure that our training data do not violate our constraints. We discard some roles, as described above, by simply removing the offending labels from the training data. In six cases, training instances directly conflict with the constraints specified — all cases involve continuation roles missing a valid preceding role. We discard these instances for *ConstrTr*.

CRF Baselines. As baseline models, we use the same architecture, but with a standard CRF replacing the RegCCRF. Since we are not limited by GPU memory for CRFs, we are optionally able to include all role types present in the training set, using the complete training set. We present two CRF baseline models: *CRF-full*, which is trained on all role-types from the training set, and *CRF-reduced*, which includes the same subset of roles as the RegCCRF models. For *CRF-reduced*, we use the same learned weights as for CD, but we decode without constraints.

Results and analysis. We evaluate our models on the evaluation partition, and measure performance using F_1 score for exact span matches. For comparability with prior work, we use the evaluation script[2] for the CoNLL-2005 shared task (Carreras & Màrquez, 2005). These results, averaged over five trials, are presented in Table 2. Excepting *CRF-reduced*, all of our models outperform the existing state-of-the-art ensemble model Ouchi et al. (2018). We ascribe this improvement over the existing literature to our use of RoBERTa – prior work in SRL relies on ELMo (Peters et al., 2018), which tends to underperform transformer-based models on downstream tasks (Devlin et al., 2019).

[1] As downloaded from `https://catalog.ldc.upenn.edu/LDC2013T19`, and preprocessed according to `https://cemantix.org/data/ontonotes.html`

[2] As available from `https://www.cs.upc.edu/~srlconll/soft.html`.

Table 2: Results from our experiments (averaged over 5 trials), along with selected reported results from recent literature. We rank of our models by precision, recall, and F_1 score – rankings differ if and only if the difference is significant at $p < 0.05$ (two-tailed), as measured by a permutation test.

	Model	Precision$^{(\text{rank})}$	Recall$^{(\text{rank})}$	$F_1{}^{(\text{rank})}$
Our experiments	RoBERTa + CRF (*CRF-full*)	86.82$^{(2)}$	87.73$^{(1)}$	87.27$^{(2)}$
	RoBERTa + CRF (*CRF-reduced*)	87.33$^{(1)}$	85.95$^{(3)}$	86.63$^{(3)}$
	RoBERTa + RegCCRF (*ConstrDec*)	87.28$^{(1)}$	87.13$^{(2)}$	87.20$^{(2)}$
	RoBERTa + RegCCRF (*ConstrTr*)	87.22$^{(1)}$	**87.79$^{(1)}$**	**87.51$^{(1)}$**
Results from literature	He et al. (2017)	—	—	85.5
	Ouchi et al. (2018)	87.1	85.3	86.2
	Ouchi et al. (2018) (ensemble)	**88.5**	85.5	87.0
	Li et al. (2019)	85.7	86.3	86.0

Of our models, *ConstrTr* significantly[3] outperforms the others in F_1 score and yields a new SOTA for SRL on OntoNotes, in line with expectations from theoretical analysis and on synthetic data. Our other three models show trade-offs between precision and recall, wherein *CRF-full* outperforms the other two in recall and underperforms them in precision. This is not surprising, as *CRF-full* is the only model capable of predicting rare role types. The other models, which use the reduced tag sets, have a theoretical maximum of 99% recall. Interestingly, when comparing between the three models that use the reduced tag sets, we find a statistically significant interaction between the constraint setting and model recall, but not between constraints and model precision: *ConstrTr* has significantly higher recall than *ConstrDec*, which in turn significantly beats *CRF-reduced* in recall, but there are no statistically significant differences between these models' precisions.

For our unconstrained models, *CRF-full* and *CRF-reduced*, the constraints specified in our automaton are violated in 0.81% and 0.84% of all output sequences respectively.[4] While this number is small, it should *not* be interpreted to mean that constraints are useless for almost all instances – as shown in Section 6.2, constraints during training can affect MAP inference even when none of the alternatives violate constraints.

8 Conclusion and Future Work

We have presented a method to constrain the output of CRFs to a regular language. Our construction permits constraints to be used at training or prediction time; both theoretically and empirically, training-time constraining better captures the data distribution. Conceptually, our approach constitutes a novel bridge between constrained CRFs and neural-weighted FSTs.

Future work could target enhancing the model's expressibility, either by allowing constraints to depend explicitly on the input as regular relations, or by investigating non-binary constraints, i.e., regular language-based constraints with learnable weights. Additionally, regular language induction (e.g. Dunay et al. (1994); Bartoli et al. (2016)) could be used to learn languages automatically, reducing manual specification and identifying non-obvious constraints. Another avenue for continuing research lies in identifying further applications for RegCCRFs. The NLP task of relation extraction could be a fruitful target – RegCCRFs offer a mechanism to make the proposal of a relation conditional on the presence of the right number and type of arguments. While our construction cannot be lifted directly to context-free languages due to the unbounded state space of the corresponding pushdown automata, context-free language can be approximated by regular languages (Mohri & Nederhof, 2001). On this basis, for example, a RegCCRF backed by a regular language describing trees of a limited depth could also be applied to tasks with context-free constraints.

To encourage the use of RegCCRFs, we provide an implementation as a Python library under the Apache 2.0 license which can be used as a drop-in replacement for standard CRFs in PyTorch.[5]

[3] All significance results are at the $p < 0.05$ level (two-tailed), as measured by a permutation test over the five trials of each model.

[4] For *CRF-full*, we only count violations of constraints for those roles that our automaton accounts for.

[5] Included anonymized in the supplementary materials.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we have released all code for the RegCCRF model as an open-source Python 3 library under the Apache 2.0 license, which is included anonymized in the supplementary materials. Additionally, we include Python scripts for reproducing all experiments presented in the paper, detailed descriptions of our datasets and preprocessing steps, and training logs. Furthermore, Appendix A lists all model hyperparameters, details the preprocessing steps taken for our experiments, and specifies the hardware used for our experiments along with average training and inference times for each experiment.

ETHICS STATEMENT

The research in this paper is fundamental in the sense that it enables machine learning models to better represent data and limit the search space at inference and learning time. It therefore does not in and of itself represent additional ethical risks on top of the previous work we build upon.

REFERENCES

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 936–945, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1098. URL https://www.aclweb.org/anthology/D17-1098.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230, 2016. doi: 10.1109/TKDE.2016.2515587.

Anne Brüggemann-Klein and Derick Wood. Deterministic regular languages. In Alain Finkel and Matthias Jantzen (eds.), *STACS 92*, pp. 173–184, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-46775-5.

Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 152–164, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0620.

F. Casacuberta and C. de la Higuera. Optimal linguistic decoding is a difficult computational problem. *Pattern Recognition Letters*, 20(8):813–821, January 1999. doi: 10.1016/S0167-8655(99)00045-8.

Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

B.D. Dunay, F.E. Petry, and B.P. Buckles. Regular language induction with genetic programming. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pp. 396–400 vol.1, 1994. doi: 10.1109/ICEC.1994.349918.

Jack Edmonds. Optimal branchings. *Journal of Research of the National Bureau of Standards*, 71B(4):233–240, 1967. URL https://nvlpubs.nist.gov/nistpubs/jres/71B/jresv71Bn4p233_A1b.pdf.

Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 1–8, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073085. URL https://aclanthology.org/P02-1001.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 42–47, 2011. URL https://aclanthology.org/P11-2008.pdf.

Hermann Gruber and Markus Holzer. Inapproximability of nondeterministic state and transition complexity assuming P$\neq$ NP. In *International Conference on Developments in Language Theory*, pp. 205–216. Springer, 2007.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 506–512, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2081. URL https://www.aclweb.org/anthology/N18-2081.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 473–483, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1044. URL https://aclanthology.org/P17-1044.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1141. URL https://www.aclweb.org/anthology/P17-1141.

Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. In *International Colloquium on Automata, Languages, and Programming*, pp. 629–640. Springer, 1991.

Zhe Jiang. Spatial structured prediction models: Applications, challenges, and techniques. *IEEE Access*, 8:38714–38727, 2020. doi: 10.1109/ACCESS.2020.2975584.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3668–3678, 2015. doi: 10.1109/CVPR.2015.7298990.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Trausti Kristjansson, Aron Culotta, Paul Viola, and Andrew McCallum. Interactive information extraction with constrained conditional random fields. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 412–418, 2004.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://www.aclweb.org/anthology/N16-1030.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. Dependency or span, end-to-end uniform semantic role labeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6730–6737, Jul. 2019. doi: 10.1609/aaai.v33i01.33016730. URL https://ojs.aaai.org/index.php/AAAI/article/view/4645.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692.

Jonathan May and Kevin Knight. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 351–358, New York City, USA, June 2006. Association for Computational Linguistics. URL https://aclanthology.org/N06-1045.

Mehryar Mohri. A disambiguation algorithm for finite automata and functional transducers. In *International Conference on Implementation and Application of Automata*, pp. 265–277. Springer, 2012.

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

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 467–475, Stockholm, Sweden, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606149.

Vlad Niculae, Andre Martins, Mathieu Blondel, and Claire Cardie. Sparsemap: Differentiable sparse structured inference. In *International Conference on Machine Learning*, pp. 3799–3808. PMLR, 2018.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1630–1642, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1191. URL https://www.aclweb.org/anthology/D18-1191.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005. doi: 10.1162/0891201053630264. URL https://www.aclweb.org/anthology/J05-1004.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pp. 1–40, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-4501.

Lance Ramshaw and Mitchell Marcus. Text chunking using transformation-based learning. In *Natural Language Processing Using Very Large Corpora*, pp. 157–176. Springer Netherlands, Dordrecht, 1999. ISBN 978-94-017-2390-9. doi: 10.1007/978-94-017-2390-9_10. URL https://doi.org/10.1007/978-94-017-2390-9_10.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 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*, pp. 623–633, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1076. URL https://aclanthology.org/N16-1076.

Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pp. 1–8, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-2401.

Sunita Sarawagi and William W Cohen. Semi-markov conditional random fields for information extraction. *Advances in neural information processing systems*, 17:1185–1192, 2004.

Christian Scheible, Roman Klinger, and Sebastian Padó. Model architectures for quotation detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1736–1745, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1164. URL https://aclanthology.org/P16-1164.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. In *Proceedings of the ICML 2004 Workshop on Statistical Relational Learning*, 2004.

Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. doi: 10.1109/TIT.1967.1054010.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. Ontonotes: A large training corpus for enhanced processing. In *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer, 2011.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

A Experimental Design

This appendix details the training procedures and hyperparameter choices for our experiments. These are summarized in Table 3. Full code for all experiments, along with training logs, are also included in the supplementary materials.

A.1 CRFs

For all CRFs and RegCCRFs, transition potentials were initialized randomly from a normal distribution with mean zero and standard deviation 0.1. No CRFs or RegCCRFs employed special start- or end-transitions – that is, we did not insert any additional beginning-of-sequence or end-of-sequence tags for the Viterbi or forward algorithms.

Table 3: Summary of hyperparameters for our models and experiments.

CRFs	Transition score initialization	$\mathcal{N}(0, 0.1)$
Synthetic data experiments	Emission score initialization	PyTorch default
	Optimizer	SGD
	Training iterations	5000
	Batch size	50
	Initial learning rate	1.0
	Learning rate decay	10% every 100 steps
SRL experiments	RoBERTa weights	`roberta-base`
	Projection weight and bias initialization	PyTorch default
	Optimizer	Adam
	Learning rate	10^{-5}
	Batch size	2
	Gradient accumulation	4 batches

A.2 Synthetic data experiments – training procedure

For both synthetic data experiments, the emission potentials were represented explicitly for each position as trainable parameters – since the observation sequence was constant in all experiments, these did not depend on $\boldsymbol{x}$.

Parameters were initialized randomly using PyTorch default initialization, and optimized using stochastic gradient descent. To ensure fast convergence to a stable distribution, we employed learning rate decay – learning rate was initially set to 1.0, and reduced by 10% every 100 training steps.

We trained all models for a total of 5000 steps with a batch size of 50. All models were trained on CPUs. For the experiment described in Section 6.1, we trained separate models for each k – the total training time for this experiment was approximately 35 minutes. The experiment described in Section 6.2 completed training in approximately 30 seconds.

A.3 Semantic role labeling – training procedure

In the semantic role labeling (SRL) experiments, we incorporated a pretrained RoBERTa network (Liu et al., 2019) – the implementation and weights for this model were obtained using the `roberta-base` model from the Hugging Face transformers library (Wolf et al., 2020). RoBERTa embeddings were projected down to transmission scores using a linear layer with a bias – projection weights and biases were initialized using the PyTorch default initialization.

Input tokens were sub-tokenized using RoBERTa's tokenizer. The marked predicate in each sentence was prefixed by a special `<unk>` token. During training, for efficiency reasons, we excluded all sentences with 120 or more subtokens – this amounted to 0.23% of all training instances. We nonetheless predicted on all instances, regardless of length.

We optimized models using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-5}. We fine-tune RoBERTa parameters and learn the projection and RegCCRF weights jointly. For performance reasons, batch size was set to 2, but we utilized gradient accumulation over groups of 4 batches to simulate a batch size of 8.

We utilized early stopping to avoid overfitting. Every 5000 training steps, we approximated our model's F_1 score against a subset of the provided development partition, using a simplified reimplementation of the official evaluation script. Each time we exceeded the previous best F_1 score for a model, we saved all model weights to disk. After 50 such evaluations with no improvement, we terminated training, and used the last saved copy of model weights for final evaluation.

We performed all SRL experiments on GeForce GTX 1080 Ti GPUs. Each experiment used a single GPU. Training took an average of 88 hours for RegCCRF models with constrained training, 23 hours for RegCCRF with constrained decoding, and 24 hours for CRF baseline models. Inference on the complete test set took an average of 18 minutes 55 seconds for CT and CD, and an average

of 55 seconds for CRF-full and CRF-reduced. All training logs with timestamps are included in the supplementary materials.

B PROOF OF CONSTRAINED TRAINING INEQUALITY

In this appendix, we prove the inequality presented in section 5: when compared by NLL against the data distribution, $L_{\text{unconstrained}} \geq L_{\text{constrained decoding}} \geq L_{\text{constrained training}}$, with each L corresponding to that model's negative log-likelihood. We first prove the left side of this inequality, comparing an unconstrained CRF to constrained decoding, and then prove the right side, comparing constrained decoding to constrained training. We use the notation introduced in Section 5.

Theorem 1. *For arbitrary θ:* $E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}}\left[-\ln P_\theta(\boldsymbol{y} \mid \boldsymbol{x})\right] \geq E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}}\left[-\ln P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})\right]$

Here we compare the distributions $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ and $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$. We wish to demonstrate that $P_\theta(\boldsymbol{y} \mid \boldsymbol{x})$ can never achieve lower NLL than $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$, and that the two distributions achieve identical NLL only when $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ i.e. when constraints have no effect. Of note, this proof is valid for *all* parameterizations θ, and not just for θ_u.

Proof. Since every $\boldsymbol{y}$ in $\widetilde{P}$ is in $\mathcal{L}$,

$$P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L}) = \alpha \cdot P_\theta(\boldsymbol{y} \mid \boldsymbol{x}), \tag{10}$$

with $\alpha \geq 1$. Thus, the NLL of the regular-constrained CRF is

$$E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}}\left[-\ln P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})\right] = E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}}\left[-\ln P_\theta(\boldsymbol{y} \mid \boldsymbol{x})\right] - \ln \alpha. \tag{11}$$

This differs from the NLL of the unconstrained CRF only by the term $-\ln \alpha$. As $\alpha \geq 1$, the regular-constrained CRF's NLL is less than or equal to that of the unconstrained CRF, with equality only when $\alpha = 1$ and therefore $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$. ■

Theorem 2. $E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}}\left[-\ln P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})\right] \geq E_{\boldsymbol{x},\boldsymbol{y}\sim\widetilde{P}}\left[-\ln P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})\right]$

In this case, we compare the distributions $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ and $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$. We will demonstrate that the former cannot achieve a lower NLL against the data distribution than the latter.

Proof. This follows directly from our definitions, as we define θ_c to minimize the NLL of $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ against the data distribution. Thus, $P_{\theta_u}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$ could never yield a lower NLL than $P_{\theta_c}(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$, as that would contradict our definition of θ_c. ■

C CONSTRUCTION AS WEIGHTED FST

In this appendix, we present a construction of the RegCCRF as a weighted finite-state transducer with weight sharing. We do this by first specifying the transducer topology used, and then specifying how edge weights are parameterized in terms of θ. The resulting transducer yields an identical distribution to that of the CRF-based construction, $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$.

C.1 TRANSDUCER TOPOLOGY

Starting from $\mathcal{L}$, we define $\ddot{\mathcal{L}}$ to be the regular language of bigram sequences for the strings in $\mathcal{L}$, i.e.,

$$\ddot{\mathcal{L}} = \left\{\left\langle (s_1, s_2), (s_2, s_3), ..., (s_{|\boldsymbol{s}-1|}, s_{|\boldsymbol{s}|}), (s_{|\boldsymbol{s}|}, \$)\right\rangle \mid \boldsymbol{s} \in \mathcal{L}\right\}, \tag{12}$$

with \$ acting as a end-of-string symbol. We let $\ddot{M}$ be an unambiguous FSA for the language $\ddot{\mathcal{L}}$, and choose to interpret this automaton as a finite-state transducer by stipulating that all edges should accept any symbol in the input language (but only one symbol per transition, and without allowing epsilon transitions). This unweighted transducer will be used as the topology for our weighted finite-state transducer.

C.2 EDGE WEIGHTS

In line with Rastogi et al. (2016), we would like to assign weights to the edges of our transducer $\ddot{M}$ with a neural network. In order to obtain the same distribution as from our CRF-based construction, these weights must be parameterized in terms of our transition function g_θ and emission function h_θ. For each edge in $\ddot{M}$, the weight depends only on the emitted bigram, the input sequence, and the index of the current input symbol – the weight does not depend on the FST states. For a symbol bigram (a, b), input sequence $\boldsymbol{x}$, and index i, the edge weight is equal to

$$W_{a,b} = \begin{cases} g_\theta(a, b) + h_\theta(\boldsymbol{x}, a, i) & b \neq \$ \\ h_\theta(\boldsymbol{x}, a, i) & \text{otherwise} \end{cases} . \tag{13}$$

Each string in $\mathcal{L}$ corresponds bijectively to exactly one bigram sequence in $\ddot{\mathcal{L}}$, which corresponds bijectively to exactly one accepting path in $\ddot{M}$ – this path's weight (in the Log semiring) is equal to the unscaled probability produced by our CRF construction, and so the weighted FST, interpreted as a probability distribution, yields the distribution $P_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{L})$.

D AUTOMATON CONSTRUCTION FOR SEMANTIC ROLE LABELING

In this appendix, we describe how we generate the automaton architecture for our semantic role labeling experiments. While our experiments used 5 core-roles, 17 non-core roles, and one continuation role, we discuss here a generalized setting with arbitrary sets of core, noncore, and continuations of core roles.

Algorithm 1 provides pseudocode for our construction. The core idea is to use subsets of core roles as NFA states, so that we can keep track of which core roles have already ocurred. Additional states are used in order to ensure all strings are valid BIO sequences.

Data: Sets $\mathcal{R}_{\text{core}}$, $\mathcal{R}_{\text{noncore}}$, and $\mathcal{R}_{\text{continuation}}$, of core, noncore, and continuation roles, respectively
Result: A finite-state automaton $M = (\Sigma, Q, q_1, F, E)$, parameterized as described in Section 3
$\Sigma \leftarrow \{\text{OUTSIDE}\} \cup (\{\text{BEGIN}, \text{INSIDE}\} \times (\mathcal{R}_{\text{core}} \cup \mathcal{R}_{\text{noncore}} \cup \mathcal{R}_{\text{continuation}}))$;
$Q \leftarrow \varnothing$;
$q_1 \leftarrow \varnothing$;
$F \leftarrow \varnothing$;
$E \leftarrow \varnothing$;
for $p \in 2^{\mathcal{R}_{core}}$ **do**
  $Q \leftarrow Q \cup \{p\}$;
  $F \leftarrow F \cup \{p\}$;
  $E \leftarrow E \cup \{(p, \text{OUTSIDE}, p)\}$;
  for $r \in \mathcal{R}_{noncore}$ **do**
    $s \leftarrow (r, p)$;
    $Q \leftarrow Q \cup \{s\}$;
    $E \leftarrow E \cup \{(p, (\text{BEGIN}, r), p), (p, (\text{BEGIN}, r), s)\}$;
    $E \leftarrow E \cup \{(r, (\text{INSIDE}, r), s), (r, (\text{INSIDE}, r), p)\}$;
  end
  for $r \in \mathcal{R}_{continuation}$ **do**
    if *The core role corresponding to* r *is in* p **then**
      $s \leftarrow (r, p)$;
      $Q \leftarrow Q \cup \{s\}$;
      $E \leftarrow E \cup \{(p, (\text{BEGIN}, r), p), (p, (\text{BEGIN}, r), s)\}$;
      $E \leftarrow E \cup \{(r, (\text{INSIDE}, r), s), (r, (\text{INSIDE}, r), p)\}$;
    end
  end
  for $r \in (\mathcal{R}_{core} \setminus p)$ **do**
    $s \leftarrow (r, p)$;
    $t \leftarrow p \cup \{r\}$;
    $Q \leftarrow Q \cup \{s\}$;
    $E \leftarrow E \cup \{(p, (\text{BEGIN}, r), s), (p, (\text{BEGIN}, r), t)\}$;
    $E \leftarrow E \cup \{(s, (\text{INSIDE}, r), s), (s, (\text{INSIDE}, r), t)\}$;
  end
end
return (Σ, Q, q_1, F, E)

Algorithm 1: Construction of an FSA from given sets of core, noncore, and continuation roles. To represent BIO labels, we use tuples of the form (BEGIN, `<roleType>`) for B labels, tuples of the form (INSIDE, `<roleType>`) for I labels, and the symbol OUTSIDE for the sole O label.