Bregman Conditional Random Fields: Sequence Labeling with Parallelizable Inference Algorithms

Caio Corro[1] **Mathieu Lacroix**[2] **Joseph Le Roux**[2]
[1]INSA Rennes, IRISA, Inria, CNRS, Université de Rennes, France
[2]Université Sorbonne Paris Nord, CNRS, LIPN, France
caio.corro@irisa.fr {lacroix,leroux}@lipn.fr

Abstract

We propose a novel discriminative model for sequence labeling called Bregman conditional random fields (BCRF). Contrary to standard linear-chain conditional random fields, BCRF allows fast parallelizable inference algorithms based on iterative Bregman projections. We show how such models can be learned using Fenchel-Young losses, including extension for learning from partial labels. Experimentally, our approach delivers comparable results to CRF while being faster, and achieves better results in highly constrained settings compared to mean field, another parallelizable alternative.

1 Introduction

Sequence labeling is a core natural language processing task: it consists in assigning one tag per token of an input sentence. Although simple, it is powerful enough to encompass part-of-speech (POS) tagging (Church, 1988), named-entity recognition (Ramshaw and Marcus, 1995) including complex variants (Corro, 2024), word segmentation (Xue, 2003), and even syntactic and semantic parsing (Xipeng, 2009; Marcheggiani et al., 2017; Gómez-Rodríguez and Vilares, 2018).

The main structured models for sequence labeling are hidden Markov models (Jelinek, 1997) and conditional random fields (CRF, Lafferty et al., 2001). In both settings, the most probable sequence of tags is computed via the Viterbi algorithm (Viterbi, 1967; Forney, 1973). Learning parameters via maximum likelihood requires to compute the log-partition function of the exponential family distribution over all possible tag sequences (Wainwright and Jordan, 2008), which can be done using the Forward algorithm (Rabiner, 1989).

Both algorithms have a linear-time complexity in the input length. They are naturally expressed as dynamic programming algorithms that recursively solve subproblems. Unfortunately, these algorithms are inherently sequential and therefore cannot benefit from parallelization of modern GPUs.

We propose a radically different approach named Bregman conditional random fields (BCRF), inspired by entropic optimal transport (Cuturi, 2013; Benamou et al., 2015) and SPARSEMAP (Niculae et al., 2018). BCRF relies on mean regularization (Blondel et al., 2020) to define probability distributions over sequence labelings. In this setting, we develop approximate inference algorithms based on iterative Bregman projections (Bregman, 1967), where each step of the algorithm solves parallelizable subproblems, taking full advantage of modern hardware (Censor, 1998). Empirically we show that this method is accurate and compares favorably to alternatives like mean field (MF, Wang et al., 2020), especially in the presence of forbidden tag transitions, a typical requirement in segmentation and named entity recognition.

Our contributions can be summarized as follows: (1) we introduce BCRF, a novel approach to define distributions over sequence labelings; (2) we develop a strongly parallelizable inference algorithm that can be used as a drop-in replacement for both Viterbi and Forward; (3) we show how Fenchel-Young losses (Blondel et al., 2020) can be leveraged to learn the parameters of a BCRF model, including the case of learning from partial labels; (4) we experiment on POS tagging, token segmentation and named entity recognition and show that our approach delivers the same performance as standard CRF while being faster.

Our code is publicly available.[1]

2 Background

In this section, we set notations and review CRF to stress what the computational limitations are and how our approach differs from standard CRF.

[1]https://github.com/FilippoC/lightspeed-crf

Notations. We write $[\![a, b]\!]$ the set of integers from a to b. Given a discrete set S, we write $\mathbb{R}^S$ the vector of reals indexed by elements of S, and similarly for higher dimension tensors. Given set S and function h, we denote $\boldsymbol{v} = [h(s)]_{s\in S}$ the vector indexed by elements of S s.t. $v_s = h(s)$. Given matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we denote $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \sum_{i,j} A_{ij}B_{ij}$ the sum of entries of the Hadamard product (*i.e.* dot product for vectors). The $d-1$ dimension simplex is denoted $\triangle(d) = \{\boldsymbol{x} \in \mathbb{R}^d_+ : \|\boldsymbol{x}\|_1 = 1\}$, and we write $\triangle$ when dimension can be inferred from context and $\triangle(S)$ when dimensions are indexed by elements of S. Given vectors $\boldsymbol{v} \in \mathbb{R}^d_+$ and $\boldsymbol{t} \in \mathbb{R}^d_{++}$, the Shannon entropy and the Kullback-Leibler divergence[2] are respectively defined as:

$$H(\boldsymbol{v}) = -\langle \boldsymbol{v}, \log \boldsymbol{v}\rangle,$$
$$D_{\mathrm{KL}}[\boldsymbol{v}|\boldsymbol{t}] = \left\langle \boldsymbol{v}, \log \frac{\boldsymbol{v}}{\boldsymbol{t}} \right\rangle - \langle \boldsymbol{v}, \boldsymbol{1}\rangle.$$

Given a convex set S, we denote δ_S the indicator function of S:

$$\delta_S(\boldsymbol{s}) = \begin{cases} 0 & \text{if } \boldsymbol{s} \in S, \\ \infty & \text{otherwise.} \end{cases}$$

Given function h, we note h^* its Fenchel conjugate:

$$h^*(\boldsymbol{v}) = \sup_{\boldsymbol{\mu}\in \operatorname{dom} h} \langle \boldsymbol{v}, \boldsymbol{\mu}\rangle - h(\boldsymbol{\mu}),$$

and use $\max$ instead of $\sup$ when maximum exists.

2.1 Conditional Random Fields

Without loss of generality, we assume all inputs of length $n \geq 3$. Let $\boldsymbol{s} = s_1 \dots s_n$ be an input sentence and T the set of tags. We denote by X the set of sentence labelings (or tag sequences) for $\boldsymbol{s}$, *i.e.* $\boldsymbol{x} = x_1 \dots x_n \in X$ where $x_i \in T$ is the tag associated with word s_i.

A Markov random field (MRF, Hammersley and Clifford, 1971) defines a distribution over elements of X using a collection of sufficient statistics F:

$$\phi(\boldsymbol{x}) \triangleq (\phi_f(\boldsymbol{x}))_{f\in F}.$$

For sequence tagging, it is usual to rely on a first-order linear-chain, a MRF composed of unary and sequential binary sufficient statistics. Unary sufficient statistics are one-hot vectors representing each tag x_i, while binary sufficient statistics indicate transitions between consecutive tags:

$$\mathsf{bi}_i(\boldsymbol{x}) = \underbrace{[0 \dots 1 \dots 0]}_{|T|\times|T| \text{ values}},$$

i.e. $\mathsf{bi}_i(\boldsymbol{x})_{t,t'} = 1$ if $x_i = t$ and $x_{i+1} = t'$, and 0 otherwise. Note that in the case of linear chains, unary sufficient statistics are redundant. Therefore, in the following we will use only binary sufficient statistics and denote sufficient statistics as vectors:

$$\phi(\boldsymbol{x}) \in \{0, 1\}^W$$

where $W = [\![1, n-1]\!] \times T \times T$, and $\phi(\boldsymbol{x})_{i,t,t'} = 1 \Leftrightarrow x_i = t \wedge x_{i+1} = t'$. We denote the set of sufficient statistics as $Y = \{\phi(\boldsymbol{x})|\boldsymbol{x} \in X\}$.

The exponential family associated with ϕ is:

$$p_{\boldsymbol{w}}(\boldsymbol{x}) = \exp\left(\, \langle \boldsymbol{w}, \phi(\boldsymbol{x})\rangle - A_Y(\boldsymbol{w}) \,\right),$$

where $\boldsymbol{w} \in \mathbb{R}^W$ is the vector of canonical parameters weighting statistics. More precisely $w_{i,t,t'}$ is the transition score associated with tagging s_i with t and s_{i+1} with t', reflected by the sufficient statistics $\mathsf{bi}_i(\boldsymbol{x})_{t,t'}$. To forbid tag assignments, or transitions, one can set the corresponding values in $\boldsymbol{w}$ to $-\infty$. A_Y is the log-partition function ensuring that the distribution is correctly normalized:

$$A_Y(\boldsymbol{w}) = \log \sum_{\boldsymbol{y}\in Y} \exp \langle \boldsymbol{w}, \boldsymbol{y}\rangle. \tag{1}$$

Computing the most probable tag sequence $\widehat{\boldsymbol{x}}(\boldsymbol{w})$ can be reduced to solving:

$$\widehat{\boldsymbol{x}}(\boldsymbol{w}) \in \operatorname*{argmax}_{\boldsymbol{x}\in X} \langle \boldsymbol{w}, \phi(\boldsymbol{x})\rangle. \tag{2}$$

A conditional random field (CRF, Lafferty et al., 2001) is a MRF for which the canonical parameters $\boldsymbol{w}$ are conditioned on an external observation, the input sentence $\boldsymbol{s}$ in our case. To this end, their values are computed by a θ-parameterized scoring function f_θ, *e.g.* a neural net. Then, the conditional probability of $\boldsymbol{x}$ given $\boldsymbol{s}$ can be written as:

$$p_\theta(\boldsymbol{x}|\boldsymbol{s}) = \exp\left(\, \langle f_\theta(\boldsymbol{s}), \phi(\boldsymbol{x})\rangle - A_Y(f_\theta(\boldsymbol{s})) \,\right).$$

Parameters θ are usually learned from a labeled dataset $(\boldsymbol{s}, \boldsymbol{x}) \in D$ by minimizing the expected negative log-likelihood (NLL) of the data:

$$\min_\theta \mathbb{E}_{(\boldsymbol{s},\boldsymbol{x})\sim D}\left[\, \ell_{\mathrm{NLL}}(f_\theta(\boldsymbol{s}); \phi(\boldsymbol{x}))\,\right],$$

where the NLL loss function is defined as:

$$\ell_{\mathrm{NLL}}(\boldsymbol{w}; \boldsymbol{y}) = -\langle \boldsymbol{w}, \boldsymbol{y}\rangle + A_Y(\boldsymbol{w}).$$

[2] Although the definition of D_{KL} may seem surprising, it is the specific form used for iterative Bregman projections as it corresponds to a Bregman divergence (Benamou et al., 2015).

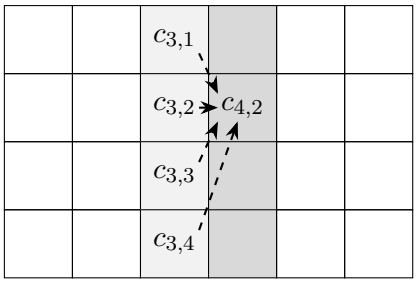

Figure 1: Illustration of the dependency between stages in the dynamic programming recursion.

2.2 The Viterbi and Forward Algorithms

We now give a brief presentation of the Viterbi and forward algorithms to compute (2) and (1), respectively, using the framework proposed by Mensch and Blondel (2018). For any vector $\boldsymbol{v} \in \mathbb{R}^d$, we define the Ω-regularized maximum function (Nesterov, 2004; Niculae and Blondel, 2017):

$$\max_\Omega(\boldsymbol{v}) = \max_{\boldsymbol{\mu} \in \triangle(d)} \langle \boldsymbol{v}, \boldsymbol{\mu} \rangle - \Omega(\boldsymbol{\mu}).$$

In particular, we can recover two known functions. First, the *hard* maximum via null regularization:

$$\Omega(\cdot) = 0 \;\Leftrightarrow\; \max_\Omega(\boldsymbol{v}) = \max_i \; v_i.$$

Second, we can express the *soft* maximum (also called *logsumexp*) via entropic regularization:

$$\Omega = -H \;\Leftrightarrow\; \max_\Omega(\boldsymbol{v}) = \log \textstyle\sum_i \exp v_i \,, \quad (3)$$

see Appendix B.

Let $c_\Omega(\boldsymbol{w})$ be the regularized maximum score over tag sequences X:

$$c_\Omega(\boldsymbol{w}) \triangleq \max_\Omega \left(\left[\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle \right]_{\boldsymbol{x} \in X} \right) .$$

We can recast computing equations (2) and (1) as:

$$c_0(\boldsymbol{w}) = \max_{\boldsymbol{x} \in X} \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle \text{ and } c_{-H}(\boldsymbol{w}) = A_Y(\boldsymbol{w}),$$

highlighting the fact that MAP and marginal inference only differ in the choice of regularization.

Although $|X|$ is exponential in the length of the input sentence, we can decompose the computation of $c_\Omega(\boldsymbol{w})$ into a sequence of subproblems, *i.e.* we can recursively compute intermediate chart values $c_{\Omega,i,t}$ via dynamic programming (Bellman, 1954) :

$$c_{\Omega,1,t}(\boldsymbol{w}) \triangleq 0,$$
$$c_{\Omega,i,t}(\boldsymbol{w}) \triangleq \max_\Omega \left(\left[c_{\Omega,i-1,t'}(\boldsymbol{w}) + w_{i,t',t} \right]_{t' \in T} \right),$$
$$c_\Omega(\boldsymbol{w}) \triangleq \max_\Omega \left(\left[c_{\Omega,n,t}(\boldsymbol{w}) \right]_{t \in T} \right).$$

Proof is given in Appendix A. Space and time complexities are $\mathcal{O}(n|T|)$ and $\mathcal{O}(n|T|^2)$, respectively.

Wavefront Parallelization. For a position i, chart values $c_{\Omega,i,t}$ do not depend on each other. We can group chart elements in stages corresponding to each position i. Elements in stage $i+1$ directly depend on elements in stage i, see Figure 1. Most implementations of the Viterbi and forward parallelize computations in a given stage, a technique called *wavefront parallelization* (Muraoka, 1971). Unfortunately, due to the dependency between stages, the n parallel computations must be performed sequentially, which prevent leveraging parallelization capabilities of GPUs.

3 Bregman Conditional Random Fields

We now introduce BCRF and link maximum a posteriori (MAP) inference with marginal inference. Then, we show how marginal inference for BCRF can be reduced to a Kullback-Liebler projection into the intersection of two convex sets. This paves the way for inference via iterative Bregman projections.

3.1 Mean Regularization

Let $\boldsymbol{M} = \{0,1\}^{W \times Y}$ be the matrix whose columns are the vectors of Y, *i.e.* $\left[\boldsymbol{M}^\top\right]_{\boldsymbol{y}} = \boldsymbol{y}$. As such, $\boldsymbol{M}^\top \boldsymbol{w}$ is a vector containing the score of each valid tag sequence. From Equation (3), we rewrite the log-partition function as:

$$A_Y(\boldsymbol{w}) = \max_{\boldsymbol{p} \in \triangle(Y)} \left\langle \boldsymbol{p}, \boldsymbol{M}^\top \boldsymbol{w} \right\rangle + \underbrace{H(\boldsymbol{p})}_{\text{distrib. reg.}} , \quad (4)$$

where the entropy regularizes the distribution over all outputs. Setting $\boldsymbol{q} = \boldsymbol{M}\boldsymbol{p}$ and optimizing over $\boldsymbol{q} \in \{\boldsymbol{M}\boldsymbol{p} | \boldsymbol{p} \in \triangle(Y)\} = \operatorname{conv}(Y)$, we obtain:

$$A_Y(\boldsymbol{w}) = \max_{\boldsymbol{q} \in \operatorname{conv}(Y)} \langle \boldsymbol{q}, \boldsymbol{w} \rangle - R(\boldsymbol{q}) \,, \quad (5)$$

where R is a structured regularization term, defined so that equality holds.[3] The set $\operatorname{conv}(Y)$ is called the marginal polytope, and computing $A_Y(\boldsymbol{w})$ is therefore often referred to as marginal inference. The gradient of A_Y is defined as (Appendix C):

$$\nabla A_Y(\boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{q} \in \operatorname{conv}(Y)} \langle \boldsymbol{q}, \boldsymbol{w} \rangle - R(\boldsymbol{q}) \,, \quad (6)$$

i.e. $\nabla A_Y(\boldsymbol{w})$ is a vector of marginal probabilities, and there exists a unique distribution over sequences $\boldsymbol{p}$ such that $\nabla A_Y(\boldsymbol{w}) = \boldsymbol{M}\boldsymbol{p}$. As such,

[3] See (Blondel et al., 2020, Eq. 25) for the exact definition.

function $A_Y(\boldsymbol{w})$ identifies an exponential family distribution over sequences X.

It is important to note that the dimension of $\boldsymbol{q} \in \text{conv}(Y)$ is polynomial in the input length, while the dimension of $\boldsymbol{p} \in \triangle(Y)$ is exponential in the input length. Key to our approach is to use another family of distributions over sequences where regularization is applied on marginals, called mean regularization.

Definition 1. Given cannonical parameters $\boldsymbol{w} \in \mathbb{R}^W$, a BCRF defines a distribution over sequence labeling Y whose marginal transition probabilities are given by $\nabla B_Y(\boldsymbol{w})$ where:

$$B_Y(\boldsymbol{w}) = \max_{\boldsymbol{p} \in \triangle(Y)} \langle \boldsymbol{p}, \boldsymbol{M}^\top \boldsymbol{w} \rangle + \underbrace{H(\boldsymbol{M}\boldsymbol{p})}_{\text{mean reg.}}.$$

Using the same change of variable, we obtain:

$$B_Y(\boldsymbol{w}) = \max_{\boldsymbol{q} \in \text{conv}(Y)} \langle \boldsymbol{q}, \boldsymbol{w} \rangle + H(\boldsymbol{q}).$$

As in Equation (5), the optimization is done over vector $\text{conv}(Y)$ but the regularization has now a simple analytical form, paving the way for efficient solvers.

MAP inference. Computing the most probable $\boldsymbol{y} \in Y$ reduces to solving an unregularized problem for both distributions defined by A_Y and B_Y:

$$\widehat{\boldsymbol{y}}(\boldsymbol{w}) \triangleq \operatorname*{argmax}_{\boldsymbol{y} \in \text{conv}(Y)} \langle \boldsymbol{w}, \boldsymbol{y} \rangle,$$

and the best tag assignation is then:

$$\widehat{\boldsymbol{x}}(\boldsymbol{w})_i = \operatorname*{argmax}_{t \in T} \max_{t' \in T} \widehat{\boldsymbol{y}}(\boldsymbol{w})_{i,t,t'}$$
$$\widehat{\boldsymbol{x}}(\boldsymbol{w})_n = \operatorname*{argmax}_{t \in T} \max_{t' \in T} \widehat{\boldsymbol{y}}(\boldsymbol{w})_{n-1,t',t}$$

for $i \in [\![1, n-1]\!]$. Although seemingly different from marginal inference, the next proposition shows that this problem can be approximated by:

$$\nabla B_Y(\tau^{-1}\boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{q} \in \text{conv}(Y)} \langle \tau^{-1}\boldsymbol{w}, \boldsymbol{q} \rangle + H(\boldsymbol{y}),$$

for a sufficiently small temperature $\tau > 0$.

Proposition 1.

$$\lim_{\tau \to 0} B_Y(\tau^{-1}\boldsymbol{w}) = \max_{\boldsymbol{y} \in \text{conv}(Y)} \langle \boldsymbol{w}, \boldsymbol{y} \rangle.$$

The proof is given in Appendix D. Although Proposition 1 tells that mean regularization can be used as approximate MAP inference, setting τ close to zero raises several issues: (1) the algorithm we develop in Section 3.3 cannot be applied when $\tau = 0$, and (2) too small values for τ may lead to computational instabilities such as over- and underflow. Therefore, we fix τ to a small value and then approximate MAP by searching for the most probable tag after marginalizing over transition probabilities:

$$\widehat{\boldsymbol{x}}(\boldsymbol{w})_i \simeq \operatorname*{argmax}_{t \in T} \sum_{t' \in T} \left[\nabla B_Y(\tau^{-1}\boldsymbol{w})\right]_{1,t,t'}$$
$$\text{and } \widehat{\boldsymbol{x}}(\boldsymbol{w})_n \simeq \operatorname*{argmax}_{t \in T} \sum_{t' \in T} \left[\nabla B_Y(\tau^{-1}\boldsymbol{w})\right]_{n-1,t',t}$$

for $i \in [\![1, n-1]\!]$. This can be understood as minimum Bayes risk decoding (MBR, Bickel and Doksum, 1977; Goodman, 1996), with risk defined as the number of incorrect predictions.[4]

3.2 Marginal Polytope

The marginal polytope $\text{conv}(Y)$ plays an important role in BCRF as inference requires to optimize a concave objective over it. In this section, we give a novel tight characterization of this polytope, that will be used to develop our algorithms. Key is our focus on transitions instead of tags.

Reduction. We formalize interpreting each vector in Y as a set of arc selection variables in a graph. Let $G = (V, E)$ be a directed graph. The node set V is partitioned into clusters $V_i = \{\text{v}_i^t \mid t \in T\}$ associated with each word s_i, for $i \in [\![1, n]\!]$. The arc set E consists of arcs connecting nodes of a cluster to nodes of the consecutive one:

$$E = \{(\text{v}_i^t, \text{v}_{i+1}^{t'}) \mid i \in [\![1, n-1]\!], t \in T, t' \in T\}.$$

By construction, any path from a node of V_1 to one of V_n covers a node in each cluster, and covering node v_i^t corresponds to assigning tag t to word s_i. Such a path is called *valid*. It is is easy to see that there is one-to-one mapping between valid paths and elements of Y: each vector $\boldsymbol{y} \in Y$ corresponds to the valid path than contains arc $(\text{v}_i^t, \text{v}_{i+1}^{t'})$ if and only if $\boldsymbol{y}_{i,t,t'} = 1$.

[4] As such, MBR decoding can return an invalid sequence of tags. In the seminal work of Goodman (1996) for syntactic parsing, this was motivated by the fact that evaluation metric did not take into account the well-formedness of predictions.

Linear Programming. We now give a description of the convex hull of Y. For a node subset $U \subseteq V$, let $\delta^+(U)$ and $\delta^-(U)$ denote the set of arcs leaving and entering U, respectively. For simplicity, we write $\delta^+(\mathrm{v})$ and $\delta^-(\mathrm{v})$ instead of $\delta^+(\{\mathrm{v}\})$ and $\delta^-(\{\mathrm{v}\})$.

Any valid path enters each cluster but V_1 once, which can be expressed as:[5]

$$\forall i \in [\![2, n-1]\!] : \sum_{a \in \delta^-(V_i)} y_a = 1. \qquad (7)$$

Next, for any node not in V_1 nor V_n, the number of incoming arcs must be egal to the number of outgoing arcs (which will be either 0 or 1 in an integral solution):

$$\begin{matrix} \forall i \in [\![2, n-1]\!], \\ \forall \mathrm{v} \in V_i \end{matrix} : \sum_{a \in \delta^+(\mathrm{v})} y_a = \sum_{a \in \delta^-(\mathrm{v})} y_a. \quad (8)$$

It is easy to see that any valid path in G satisfies these constraints. Since G is acyclic, any vector $\boldsymbol{y} \in \{0,1\}^E$ that satisfies equalities (7) and (8) describes a valid path. The next proposition shows that they define an integral polytope, *i.e.* they characterize the convex hull of feasible solutions.

Proposition 2. The tagging polytope $\mathrm{conv}(Y)$ is described by:

$$\mathrm{conv}(Y) = \{\boldsymbol{y} \in \mathbb{R}_+^E \mid \boldsymbol{y} \text{ satisfies (7), (8).}\}$$

The proof is given in Appendix E.

3.3 Iterative Bregman Projections

As explained in the previous section, both MAP and marginal inference can be reduced to computing $\nabla B_Y(\cdot)$. Next, we introduce our efficient method based on iterative Bregman projections (IBP).

First, notice that our objective can be reduced to a Kullback-Leibler (KL) projection:

$$\begin{aligned} &\operatorname*{argmax}_{\boldsymbol{q} \in \mathrm{conv}(Y)} \langle \boldsymbol{q}, \boldsymbol{w} \rangle + \tau H[\boldsymbol{q}] \\ &= \operatorname*{argmin}_{\boldsymbol{q} \in \mathrm{conv}(Y)} \langle \boldsymbol{q}, \log \boldsymbol{q} \rangle - \langle \boldsymbol{q}, \mathbf{1} \rangle - \langle \boldsymbol{q}, \log \exp \tau^{-1} \boldsymbol{w} \rangle \\ &= \operatorname*{argmin}_{\boldsymbol{q} \in \mathrm{conv}(Y)} D_{\mathrm{KL}}[\boldsymbol{q} | \exp \tau^{-1} \boldsymbol{w}], \end{aligned}$$

where the equality in the second line holds because $\langle \boldsymbol{q}, \mathbf{1} \rangle = n-1$ is constant by constraints (7) and (8).

[5] Since G is acyclic and arcs connect consecutive clusters, only one of these constraints is necessary, the others becoming redundant. However, they are kept in order to strengthen each subproblem in our decomposition.

This optimization problem aims to compute the projection of $\exp \tau^{-1} \boldsymbol{w}$ into the set $\mathrm{conv}(Y)$ where the distance is measured using the KL divergence. In order to use on IBP, we need to rewrite $\mathrm{conv}(Y)$ as an intersection of convex sets for which we can derive efficient KL projections. We describe $\mathrm{conv}(Y)$ as the intersection of two polytopes $\mathcal{C}_{\text{even}}$ and $\mathcal{C}_{\text{odd}}$ by partitioning the set of constraints (7) and (8) into those associated with clusters V_i with i odd, and those associated with clusters V_i with i even.[6]

Let $\mathcal{C}_i$ be the set of vectors $\boldsymbol{q}$ for which constraints on the arcs that are incident to vertices in cluster V_i are satisfied, that is:

$$\mathcal{C}_i = \left\{ \boldsymbol{q} \in \mathbb{R}_+^E \;\middle|\; \begin{array}{l} \sum_{a \in \delta^-(V_i)} q_a = 1, \\ \forall \mathrm{v} \in V_i : \sum_{a \in \delta^+(\mathrm{v})} q_a \\ \qquad\qquad = \sum_{a \in \delta^-(\mathrm{v})} q_a \end{array} \right\}.$$

We define $\mathcal{C}_{\text{even}}$ and $\mathcal{C}_{\text{odd}}$ as the intersection of polytopes $\mathcal{C}_i$ as follows:

$$\mathcal{C}_{\text{even}} = \bigcap_{j=1}^{\lceil \frac{n}{2} \rceil - 1} \mathcal{C}_{2j} \;\text{ and }\; \mathcal{C}_{\text{odd}} = \bigcap_{j=1}^{\lfloor \frac{n}{2} \rfloor - 1} \mathcal{C}_{2j+1}.$$

Therefore, we have $\mathrm{conv}(Y) = \mathcal{C}_{\text{even}} \cap \mathcal{C}_{\text{odd}}$.

We now describe how to compute the KL projection onto $\mathcal{C}_{\text{even}}$. Since the constraints of each $\mathcal{C}_i$ contains only variables associated with arcs $\delta(V_i)$ and the objective function of the projection decomposes per arc, one can decompose computing $\min_{\boldsymbol{q} \in \mathcal{C}_{\text{even}}} D_{\mathrm{KL}}(\boldsymbol{q}, \boldsymbol{w})$ as:

$$\sum_{j=1}^{\lceil \frac{n}{2} \rceil - 1} \min_{\boldsymbol{q}|_{\delta(V_{2j})} \in \mathcal{C}_{2j}} D_{\mathrm{KL}} \left(\boldsymbol{q}|_{\delta(V_{2j})}, \boldsymbol{w}|_{\delta(V_{2j})} \right),$$

where for a vector $\boldsymbol{v} \in \mathbb{R}^S$ and $S' \subseteq S$, $\boldsymbol{v}|_{S'}$ denotes the the restriction of $\boldsymbol{v}$ to the indices associated with the elements of S', and when used in set inclusions it also restricts the right-hand side elements. Similarly, $\min_{\boldsymbol{q} \in \mathcal{C}_{\text{odd}}} D_{\mathrm{KL}}(\boldsymbol{q}, \boldsymbol{w})$ can be rewritten as:

$$\sum_{j=1}^{\lfloor \frac{n}{2} \rfloor - 1} \min_{\boldsymbol{q}|_{\delta(V_{2j+1})} \in \mathcal{C}_{2j+1}} D_{\mathrm{KL}} \left(\boldsymbol{q}|_{\delta(V_{2j+1})}, \boldsymbol{w}|_{\delta(V_{2j+1})} \right).$$

Hence, each KL projection reduces to a sum of KL projections $\min_{\boldsymbol{q}|_{\delta(V_i)} \in \mathcal{C}_i} D_{\mathrm{KL}} \left(\boldsymbol{q}|_{\delta(V_i)}, \boldsymbol{w}|_{\delta(V_i)} \right)$ with closed-form solutions, see Appendix F.

[6] Note that clusters V_1 and V_n are not considered for defining $\mathcal{C}_{\text{even}}$ and $\mathcal{C}_{\text{odd}}$ since all constraints are associated with a cluster V_i with $i \in [\![2, n-1]\!]$.

Algorithm 1 The BCRF inference algorithm based on iterative Bregman projections, where $\boldsymbol{w} \in \mathbb{R}^W$ are canonical parameters, n the input length and k the number of iterations. When using the $|$ notation on the left-hand side of an assignation, we assume only the designated coordinates are updated.

function BCRFINFERENCE($\boldsymbol{w}, n, k$)
  $\boldsymbol{q}^{(0)} \leftarrow \boldsymbol{w}$
  for $i \leftarrow 1 \dots k$ **do**
    $\boldsymbol{q}^{(i)} \in \mathbb{R}^W$ ▷ Randomly initialized vector
    if $i \equiv 0 \pmod 2$ **then**
      for $j \leftarrow 1 \dots \lceil \frac{n}{2} \rceil - 1$ **do** ▷ Can be computed in parallel
        $\boldsymbol{q}^{(i)}|_{\delta(V_{2j})} \leftarrow \underset{\boldsymbol{\mu} \in \mathbb{R}^W \text{ s.t. } \boldsymbol{\mu}|_{\delta(V_{2j})} \in \mathcal{C}_{2j}}{\operatorname{argmin}} D_{\mathrm{KL}}\left(\boldsymbol{\mu}|_{\delta(V_{2j})}, \boldsymbol{q}^{(i-1)}|_{\delta(V_{2j})}\right)$
    else
      for $j = 1 \dots \lfloor \frac{n}{2} \rfloor - 1$ **do** ▷ Can be computed in parallel
        $\boldsymbol{q}^{(i)}|_{\delta(V_{2j+1})} \leftarrow \underset{\boldsymbol{\mu} \in \mathbb{R}^W \text{ s.t. } \boldsymbol{\mu}|_{\delta(V_{2j+1})} \in \mathcal{C}_{2p+1}}{\operatorname{argmin}} D_{\mathrm{KL}}\left(\boldsymbol{\mu}|_{\delta(V_{2j+1})}, \boldsymbol{q}^{(i-1)}|_{\delta(V_{2j+1})}\right)$
  return $\boldsymbol{q}^{(k)}$ ▷ Returns an approximation of $\nabla B_Y(\boldsymbol{w})$

The full algorithm simply alternates between projections in $\mathcal{C}_{\text{odd}}$ and $\mathcal{C}_{\text{even}}$. Pseudocode is given in Algorithm 1. A more detailed description of ITB algorithm can be found in (Benamou et al., 2015, Sec. 2.1)

4 Loss Functions

In Section 3, we defined a mean regularized distribution over sequences, and show that both marginal inference and MAP in this setting reduce to computing $\nabla B_Y(\cdot)$. We now define loss functions for supervised and weakly-supervised learning also based on computing $\nabla B_Y(\cdot)$, and thus enjoying the same computational properties.

4.1 Supervised Learning

For supervised learning, we rely on the Fenchel-Young (FY) loss framework (Blondel et al., 2020).

Definition 2. Given a regularization function Ω, the FY loss is defined as:

$$\ell_\Omega(\boldsymbol{w}; \boldsymbol{y}) \triangleq -\langle \boldsymbol{w}, \boldsymbol{y} \rangle + \Omega(\boldsymbol{y}) + \left(\Omega + \delta_{\mathrm{conv}(Y)}\right)^*(\boldsymbol{w}),$$

where $\boldsymbol{y} \in Y$ is the gold output and $\boldsymbol{w}$ are weights.

Setting $\Omega = R$ as defined in Eq. 5, we obtain the NLL loss for CRFs since the last term is then $A_Y(\boldsymbol{w})$. This framework naturally suggests using mean regularization $\Omega = -H$ to define a loss function, which gives the following loss and gradient:

$$\ell_{-H}(\boldsymbol{w}; \boldsymbol{y}) = -\langle \boldsymbol{w}, \boldsymbol{y} \rangle - H(\boldsymbol{y}) + B_Y(\boldsymbol{w})$$
$$\text{and } \nabla \ell_{-H}(\boldsymbol{w}; \boldsymbol{y}) = -\boldsymbol{y} + \nabla B_Y(\boldsymbol{w}),$$

i.e. computing the gradient can be done using our inference algorithm.

4.2 Learning with Partial Labels

Learning with partial labels is a weakly supervised learning scenario where we do not have access to the gold annotation, but instead, for each input, we have access to a subset of labels $\widetilde{Y} \subseteq Y$ containing the gold label.

Definition 3. Given a regularization function Ω, the partial FY loss is defined as:

$$\widetilde{\ell}_\Omega(\boldsymbol{w}; \widetilde{Y}) \triangleq \left(\Omega + \delta_{\mathrm{conv}(Y)}\right)^*(\boldsymbol{w}) - \left(\Omega + \delta_{\mathrm{conv}(\widetilde{Y})}\right)^*(\boldsymbol{w}),$$

where $\widetilde{Y} \subset Y$ is a non-empty set containing the expected gold output.

Proposition 3. Partial FY losses satisfy the following properties:

1. *Generalization of FY losses.* If $\widetilde{Y} = \{\boldsymbol{y}\}$ is a singleton containing only a gold label, then $\widetilde{\ell}_\Omega(\boldsymbol{w}, \{\boldsymbol{y}\}) = \ell_\Omega(\boldsymbol{w}, \boldsymbol{y})$.
2. *Non-negativity.* The loss is bounded below by 0.
3. *Smaller partial labeling set $\implies$ bigger loss.* Let $\widetilde{Y}' \subseteq \widetilde{Y}$, then:

$$\widetilde{\ell}(\boldsymbol{w}; \widetilde{Y}') \geq \widetilde{\ell}(\boldsymbol{w}; \widetilde{Y}).$$

Proofs are given in App. H. This family of losses was introduced by Stewart et al. (2023). They analyze properties of the loss function when $\Omega = 0$,

that we (trivially) extend to more general regularization. Importantly, property (5) shows that the more information we have, the better is the loss

Setting $\Omega = R$, we obtain the NLL loss after marginalizing over $\widetilde{Y}$, a standard approach in learning from partial labels (Jin and Ghahramani, 2002) that can be understood as a special instance of the EM algorithm, see (Corro, 2024, Sec. 4.1). Yang et al. (2018) and Huang et al. (2019) learned sequence labeling models using this loss. For the BCRF case, we rely on mean regularization and define the following loss function and gradient:

$$\widetilde{\ell}_{-H}(\boldsymbol{w}; \widetilde{Y}) = B_Y(\boldsymbol{w}) - B_{\widetilde{Y}}(\boldsymbol{w})$$
$$\text{and } \nabla\widetilde{\ell}_{-H}(\boldsymbol{w}; \widetilde{Y}) = \nabla B_Y(\boldsymbol{w}) - \nabla B_{\widetilde{Y}}(\boldsymbol{w})$$

which requires two calls to our IBP algorithm.

5 Related Work

The main motivation of our approach is to develop a sequence labeling model suitable for parallel processors, e.g. GPUs. Developing such models is a shared preoccupation in deep learning, starting with the ubiquitous transformers (Vaswani et al., 2017) or more recently state-space models (Gu et al., 2022) and following works on linear transformers (Yang et al., 2024, Table 2). Designing NLP algorithms for GPUs has been explored in syntactic parsing (Johnson, 2011; Hall et al., 2014) and machine translation (He et al., 2015; Argueta and Chiang, 2018). More generally, efficient parallelization of dynamic algorithms is widely studied (Valiant et al., 1983; Maleki et al., 2014).

With motivations similar to ours, Wang et al. (2020) proposed to rely on mean field (MF, Weiss, 1907; Parisi, 1979) to approximate a CRF distribution using a parallel algorithm. However, MF has the following issues, all of which are solved by our approach: (1) the MF objective (see Appendix G for details) is non-convex (Wainwright and Jordan, 2008, Sec. 5.4), and therefore its approximation strongly depends on initialization; (2) the parallel decoding algorithm used for MF by Wang et al. (2020) is not even guaranteed to converge to a local minima (Kraehenbuehl and Koltun, 2013); (3) MF cannot handle highly structured problems, *i.e.* problems where there are forbidden transitions between tags to ensure well-formedness of prediction (see our experiments and Appendix G).

Our method also has connections with frameworks based on mean regularization, such as SPARSEMAP (Niculae et al., 2018). The main difference is that we use an entropic regularization in order to develop a fast alternative to the dynamic programming (DP) inference algorithms, while SPARSEMAP solver relies on iterative calls to these DP algorithms, and hence is slower.

6 Experiments

We present a series of experiments on part-of-speech (POS) tagging, joint word segmentation and POS tagging, and named entity recognition (NER). We compare CRF (*i.e.* Viterbi/forward), MF and BCRF in terms of accuracy and time. For reference, we also include an unstructured model, that predict each tag independently.[7] For MF and BCRF, we evaluate the algorithms with both 5 and 10 iterations of the respective inference algorithms. For decoding with BCRF, we set $\tau^{-1} = 10$ in all experiments.

We use datasets from the Universal Dependency Treebank 2.15 (UD, Nivre et al., 2020) for the first two tasks and the CONLL 2003 shared task data (Tjong Kim Sang and De Meulder, 2003) for the last one. We use standard splits in all cases. We average results over 5 runs initialized randomly.

Neural network. The weight function f_θ is implemented by a self-attentive encoder network (Vaswani et al., 2017) for all our experiments. Tag and transition scores are computed by a simple multi-linear perceptron. Note that our model is non-homogeneous, that is transition scores are computed independently for each adjacent word. The neural network is trained from scratch, except for NER where we use BERT (Devlin et al., 2019). Hyperparameters are given in Appendix I.

6.1 Part-of-Speech Tagging

We evaluate on four corpora from UD: LassySmall (Dutch), EWT (English) and GSD (French and German). We use coarse POS tags for all languages except French as it contains only UD tags. Results are given in Table 1a. We can see that all methods are close, but our algorithm outperforms MF for approximate decoding after training with the standard CRF loss.

6.2 Word Segmentation and POS Tagging

We cast joint word segmentation and POS tagging as a character sequence labeling problem in the

[7] Unstructured models cannot guarantee well-formedness of predictions for highly constrained settings. In these cases, we rely on simple heuristics to construct valid outputs.

	No self-attentive encoder				2 layers			
	Dutch	English	French	German	Dutch	English	French	German
Baseline								
Unstructured	90.5	85.8	92.7	89.0	93.4	90.8	96.0	94.0
CRF training								
Viterbi	94.3	91.3	95.9	94.1	94.7	91.9	96.2	94.3
MF 5 it.	93.8	90.6	95.5	94.0	94.4	91.0	95.8	94.2
MF 10 it.	93.9	90.7	95.5	94.1	94.5	91.0	95.8	94.2
BCRF 5 it.	94.4	91.3	95.9	94.1	94.7	91.9	96.2	94.3
BCRF 10 it.	94.3	91.3	95.9	94.1	94.7	91.9	96.2	94.3
MF training								
Viterbi	93.7	89.1	95.8	93.8	94.6	91.6	96.3	94.0
MF 5 it.	94.3	91.4	95.9	94.0	94.7	91.7	96.3	94.1
MF 10 it.	94.3	91.4	95.9	94.0	94.7	91.7	96.3	94.1
BCRF training								
Viterbi	94.1	91.1	95.8	94.0	94.6	91.8	96.5	94.4
BCRF 5 it.	94.1	91.1	95.8	94.0	94.6	91.8	96.5	94.4
BCRF 10 it.	94.1	91.1	95.8	94.0	94.6	91.8	96.5	94.4

(a) Accuracy results for POS tagging.

	BERT	
	Base	Large
Baseline		
Unstructured	91.6	92.0
CRF training		
Viterbi	91.9	92.3
MF 5 it.	91.7	91.9
MF 10 it.	91.6	91.9
BCRF 5 it.	91.9	92.3
BCRF 10 it.	91.9	92.3
MF training		
Viterbi	91.7	92.2
MF 5 it.	91.6	92.1
MF 10 it.	91.6	92.1
BCRF training		
Viterbi	91.7	92.1
BCRF 5	91.6	92.1
BCRF 10	91.7	92.1

(b) NER F-scores.

	No self-att. encoder		2 layers		4 layers	
	Chinese	Japanese	Chinese	Japanese	Chinese	Japanese
Baseline						
Unstructured	47.3	52.9	62.0	86.3	64.9	89.0
CRF training						
Viterbi	84.2	90.6	84.3	90.8	84.3	90.7
MF 5	72.9	76.5	71.4	75.4	71.9	75.1
MF 10	74.6	77.6	72.2	76.1	72.5	76.0
BCRF 5	81.9	88.7	81.7	88.7	82.0	88.5
BCRF 10	83.7	90.1	83.7	90.4	83.8	90.2
MF training						
Viterbi	77.6	79.6	79.9	82.3	81.1	83.5
MF 5	81.0	88.7	80.7	88.8	80.8	89.0
MF 10	82.0	89.1	81.2	89.1	81.2	89.3
BCRF training						
Viterbi	83.6	89.8	84.5	90.5	84.0	90.7
BCRF 5	82.2	88.8	83.3	89.8	82.8	90.1
BCRF 10	83.2	89.6	84.2	90.4	83.8	90.6

(c) F-score for joint segmentation and POS tagging.

	BERT	
	Base	Large
CRF training		
Viterbi	90.5	91.2
MF 5 it.	90.2	90.8
MF 10 it.	90.3	90.8
BCRF 5 it.	90.4	91.2
BCRF 10 it.	90.5	91.2
BCRF training		
Viterbi	90.6	91.2
MF 5 it.	90.3	90.9
MF 10 it.	90.3	90.9
BCRF 5 it.	90.6	91.2
BCRF 10 it.	90.6	91.2

(d) Partially supervised NER F-scores.

Table 1: Experimental results. The unstructured model does not have transition scores neither constraints between adjacent pair of tags. For each training approach, we evaluate different decoding strategies.

	Dutch		English		French		German	
L.	MF	BCRF	MF	BCRF	MF	BCRF	MF	BCRF
0	×6.2	×7.3	×5.0	×6.1	×6.0	×6.5	×5.4	×6.6
2	×3.8	×4.1	×3.9	×4.1	×4.3	×4.7	×3.8	×4.0

(a) POS tagging.

	Chinese		Japanese	
L.	MF	BCRF	MF	BCRF
0	×8.8	×7.8	×12.1	×12.3
2	×5.5	×5.4	×5.6	×5.9
4	×4.1	×4.0	×4.4	×4.5

(b) Joint word segmentation and POS tagging.

	Sup.	Partial
	BCRF	BCRF
Base	×1.9	×1.7
Large	×1.2	×1.2

(c) NER.

Table 2: Relative training time speed-up compared to using the forward algorithm to compute the loss. We set the number of iterations to 10. Column *L.* is the number of encoder layers, whereas *Base*/*Large* refer to BERT sizes.

BIES format (Ratinov and Roth, 2009). For example, tag `B-VB` indicates the first character of a multi-character verb, whereas `S-VB` indicates a single character verb. Note that this problem is highly structured: for example, we can only assign tag `I-VB` if the previous tag was `B-VB` or `I-VB`. This is implemented using $-\infty$ weights on forbidden tag transitions. We report the F-score on reconstructed tagged words. We present experiments on Chinese and Japanese UD GSD datasets.

Results are given in Table 1c. We see that on this task, with more tags than simple POS tagging and, more importantly, with constraints on forbidden adjacent tag pairs, both unstructured approach and MF struggle to recover the exact CRF performance. In contrast, decoding with the BCRF approach performs better than MF, and when training is performed with BCRF too, the F-scores are almost similar to the CRF model.

6.3 Named-Entity Recognition

For NER we report F-scores of reconstructed labeled mentions on CoNLL 2003, using BERT (Devlin et al., 2019) in the base and large versions. In this series of experiments we test supervised and partially supervised training regimes.

For supervised training, we report results in Table 1b. As in POS tagging, we see that BCRF is closer to CRF than MF, and in the case of the large BERT model, BCRF manages to (slightly) improve over CRF decoding.

For partially supervised learning, we follow the same setting as Huang et al. (2019) and split the training set into 4 subsets, each annotated with only one entity type. For a given sentence $\boldsymbol{s}$, the partial labeling $\widetilde{Y}$ is all sequence labelings that contains at least the gold annotated tags. We report results in Table 1d. We see that our method recovers the same accuracy as CRF when trained with CRF loss, while MF does not. When trained with BCRF, there is a small drop in performance, but CRF and BCRF give equivalent results while MF is lower.

6.4 Speed Improvement

We now analyze the time speed-up. All timing include the forward (and backward if training) pass in the neural network.

We report relative training time in Table 2. For POS tagging our model trains up to 7.3 times faster than standard CRF training with simple word embeddings and up to 4.7 times when using 2 encoder layers. Moreover, our method is faster than MF training. For joint segmentation and POS tagging, MF is slightly faster, but at the expense of a drop in performance (see Table 1c). Importantly, for NER, MF cannot be used in the partial supervision scenario. Our BCRF approach is significantly faster than CRF training even when using a full BERT model.

	Dutch		English		French		German	
L.	MF	BCRF	MF	BCRF	MF	BCRF	MF	BCRF
0	×9.1	×5.4	×8.0	×4.9	×10.4	×7.0	×8.5	×5.2
2	×6.6	×4.7	×6.2	×4.4	×8.1	×5.9	×6.4	×4.5

(a) POS tagging

	Chinese		Japanese	
L.	MF	BCRF	MF	BCRF
5 iterations				
0	×14.2	×7.7	×24.6	×14.6
2	×9.0	×6.2	×12.0	×9.3
4	×6.9	×5.0	×9.3	×7.8
10 iterations				
0	×9.5	×5.2	×19.8	×11.0
2	×7.1	×4.6	×10.5	×7.7
4	×5.8	×3.8	×8.7	×6.8

(b) Joint word segmentation and POS tagging.

	MF	BCRF
5 iterations		
Base	×2.9	×2.6
Large	×1.6	×1.5
10 iterations		
Base	×2.9	×2.4
Large	×1.6	×1.5

(c) NER.

Table 3: Relative decoding time speed-up compared to the Viterbi algorithm. Column *L.* is the number of encoder layers, whereas *Base*/*Large* refer to BERT sizes.

We report relative decoding time in Table 3. Although MF is faster that BCRF, the difference diminishes as the network becomes larger. Moreover, as shown in downstream task results, BCRF can handle structual constraint, while still being faster than using the Viterbi algorithm for decoding.

7 Conclusion

In this work, we introduce BCRF, a novel sequence labeling model based on entropic mean regularization, and a novel inference algorithm based on iterative Bregman projections. This method is designed to take full advantage of parallel processors.

While we designed BCRF for sequence labeling, we believe that the proposed methodology paves the way for novel inference algorithms in other structured prediction settings, such as parsing but also automatic speech recognition for which Viterbi and forward are known bottlenecks (Ondel et al., 2022).

Acknowledgements

This work was granted access to the HPC resources of IDRIS under the allocation 2024-AD011013727R1 made by GENCI. This work was supported by the LABEX EFL (Empirical Foundations of Linguistics, ANR-10-LABX-0083), operated by the French National Research Agency (ANR). This work is supported by the SEMIAMOR (CE23-2023-0005) and InExtenso (ANR-23-IAS1-0004) project grants given by the French National Research Agency (ANR).

Limitations

While our model demonstrates that a theoretical treatment of a NLP task deemed simple and well understood can lead to a novel method with practical effectiveness, it suffers some limitations.

First, our model is designed with parallel computing architectures in mind, *i.e.* GPUs. On a purely sequential architecture. However, we observe that parallel architectures have become ubiquitous and so it seems a reasonable limitation.

Second, our method relies on the fact that the continuous relaxation of the optimization problem underlying the labeling task is integral. While our approach can be easily adapted to second-order linear chains using the standard trick (He, 1988), it cannot directly rely on the *factorized* second-order CRF trick of Wang et al. (2020). More generally, our approach cannot be applied on CRF more general than linear chain ones as our formulation would not ensure local consistency without extra variables, but adding these variables results in problems for which we may not find a closed-form expression of the solution to the KL projection. However, our method could be used to solve linear-chain CRF arising from dual decomposition based methods like in the row and column decomposition of a grid CRF (Komodakis et al., 2011).

Finally, while our approach is generic in the sense that it could be applied to any dynamic programming algorithm that operates on graphs, it cannot be trivially adapted to dynamic programming algorithms that operate on hypergraphs (Martin et al., 1990).

References

Arturo Argueta and David Chiang. 2018. Composing finite state transducers on GPUs. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2697–2705, Melbourne, Australia. Association for Computational Linguistics.

Amir Beck. 2017. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Richard Bellman. 1954. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. 2015. Iterative bregman projections for regularized transportation problems. *SIAM J. Sci. Comput.*, 37(2).

Dimitri P Bertsekas. 1999. *Nonlinear programming*. Athena Scientific Belmont.

Peter J. Bickel and Kjell A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected Topics*. Prentice Hall.

Mathieu Blondel, André F.T. Martins, and Vlad Niculae. 2020. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69.

Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Lev M. Bregman. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217.

Yair Censor. 1998. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press-New York, NY.

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA. Association for Computational Linguistics.

Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. 2013. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143.

Caio Filippo Corro. 2024. A fast and sound tagging method for discontinuous named-entity recognition. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19506–19518, Miami, Florida, USA. Association for Computational Linguistics.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

John M. Danskin. 1966. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664.

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.

S. C. Fang. 1992. An unconstrained convex programming view of linear programming. *ZOR Zeitschrift für Operations Research Methods and Models of Operations Research*, 36(2):149–161.

S.-C. Fang, J. R. Rajasekera, and H.-S. J. Tsao. 1997. *Entropy Optimization and Mathematical Programming*. Springer US.

G.D. Forney. 1973. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, California, USA. Association for Computational Linguistics.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–606.

Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

David Hall, Taylor Berg-Kirkpatrick, and Dan Klein. 2014. Sparser, better, faster GPU parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 208–217, Baltimore, Maryland. Association for Computational Linguistics.

JM Hammersley and P Clifford. 1971. Markov fields on finite graphs and lattices.

Hua He, Jimmy Lin, and Adam Lopez. 2015. Gappy pattern matching on GPUs for on-demand extraction of hierarchical translation grammars. *Transactions of the Association for Computational Linguistics*, 3:87–100.

Yang He. 1988. Extended viterbi algorithm for second order hidden markov process. In *[1988 Proceedings] 9th International Conference on Pattern Recognition*, pages 718–720 vol.2.

Xiao Huang, Li Dong, Elizabeth Boschee, and Nanyun Peng. 2019. Learning a unified named entity tagger from multiple partially annotated corpora for efficient adaptation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 515–527, Hong Kong, China. Association for Computational Linguistics.

Frederik Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Mark Johnson. 2011. Parsing in parallel on multiple cores and GPUs. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 29–37, Canberra, Australia.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2011. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552.

Philipp Kraehenbuehl and Vladlen Koltun. 2013. Parameter learning and convergent inference for dense random fields. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 513–521, Atlanta, Georgia, USA. PMLR.

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 (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. 2014. Parallelizing dynamic programming through rank convergence. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '14, Orlando, FL, USA, February 15-19, 2014*, pages 219–232. ACM.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

R Kipp Martin, Ronald L Rardin, and Brian A Campbell. 1990. Polyhedral characterization of discrete dynamic programming. *Operations research*, 38(1):127–138.

Arthur Mensch and Mathieu Blondel. 2018. Differentiable dynamic programming for structured prediction and attention. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3462–3471. PMLR.

Yoichi Muraoka. 1971. *Parallelism exposure and exploitation in programs*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Yu. Nesterov. 2004. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vlad Niculae, Andre Martins, Mathieu Blondel, and Claire Cardie. 2018. SparseMAP: Differentiable sparse structured inference. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3799–3808. PMLR.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Lucas Ondel, Léa-Marie Lam-Yee-Mui, Martin Kocour, Caio Filippo Corro, and Lukás Burget. 2022. Gpu-accelerated forward-backward algorithm with application to lattice-free mmi. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8417–8421.

Giorgio Parisi. 1979. Toward a mean field theory for spin glasses. *Physics Letters A*, 73(3):203–205.

Gabriel Peyré and Marco Cuturi. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Alexander Schrijver. 1986. *Theory of linear and integer programming*. Publication Title: Wiley-Interscience series in discrete mathematics and optimization.

Lawrence Stewart, Francis Bach, Felipe Llinares-Lopez, and Quentin Berthet. 2023. Differentiable clustering with perturbed spanning forests. In *Advances in Neural Information Processing Systems*, volume 36, pages 31158–31176. Curran Associates, Inc.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. 1983. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Martin J Wainwright and Michael I Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2020. AIN: Fast and accurate sequence labeling with approximate inference network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6019–6026, Online. Association for Computational Linguistics.

Pierre Weiss. 1907. L'hypothèse du champ moléculaire et la propriété ferromagnétique. *J. Phys. Theor. Appl.*, 6(1):661–690.

J. F. Q. Xipeng. 2009. A new chinese dependency analysis method based on sequence labeling model. In *Computer Applications and Software*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, pages 29–48.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, pages 115491–115522. Curran Associates, Inc.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2159–2169, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. 2015. Conditional random fields as recurrent neural networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1529–1537. IEEE Computer Society.

A Dynamic Programming Recursion

This section contains proof of correctness for both Viterbi and forward algorithms in a single analysis of the dynamic programming recursion. A popular technique to this end is based on the semiring parsing framework (Goodman, 1999). We give an alternate analysis by following the analysis of Mensch and Blondel (2018).[8]

We first prove two properties of the $\max_\Omega$ operator for $\Omega \in \{0, -H\}$: associativity, and distributivity over addition.

A.1 Associativity of $\max_\Omega$

We show that:

$$\max_\Omega \begin{bmatrix} \max_\Omega(\boldsymbol{v}) \\ \max_\Omega(\boldsymbol{v}') \end{bmatrix} = \max_\Omega(\boldsymbol{u})$$

for all $\boldsymbol{v} \in \mathbb{R}^d, \boldsymbol{v}' \in \mathbb{R}^{d'}$ and $\boldsymbol{u} \in \mathbb{R}^{d+d'}$ where:

$$u_i \triangleq \begin{cases} v_i & \text{if } i \in [\![1, d]\!] \,, \\ v'_{i-d} & \text{otherwise,} \end{cases}$$

i.e. $\boldsymbol{u}$ is the concatenation of $\boldsymbol{v}$ and $\boldsymbol{v}'$.

In the case $\Omega = 0$, we have:

$$\begin{aligned} \max_0 \begin{bmatrix} \max_0 \boldsymbol{v}, \\ \max_0 \boldsymbol{v}' \end{bmatrix} &= \max_0 \begin{bmatrix} \max_i(v_i) \\ \max_i(v'_i) \end{bmatrix} \\ &= \max_i(u_i) \\ &= \max_0(\boldsymbol{u}) \end{aligned}$$

which proves the property.

In the case $\Omega = -H$, we have:

$$\begin{aligned} \max_{-H} \begin{bmatrix} \max_{-H} \boldsymbol{v}, \\ \max_{-H} \boldsymbol{v}' \end{bmatrix} &= \max_{-H} \begin{bmatrix} \log \sum_i \exp(v_i) \\ \log \sum_i \exp(v'_i) \end{bmatrix} \\ &= \log \begin{pmatrix} \cancel{\exp \log} \sum_i \exp(v_i) \\ + \cancel{\exp \log} \sum_i \exp(v'_i) \end{pmatrix} \\ &= \log \sum_i \exp(u_i) \\ &= \max_{-H}(\boldsymbol{u}) \end{aligned}$$

which proves the property.

A.2 Distributivity over addition of $\max_\Omega$

We show that:

$$\max_\Omega(\boldsymbol{v} + c\mathbf{1}) = c + \max_\Omega(\boldsymbol{v}) \,,$$

for all vectors $\boldsymbol{v} \in \mathbb{R}^d$ and scalars $c \in \mathbb{R}$.

In the case $\Omega = 0$, we have:

$$\max_0(\boldsymbol{v} + c\mathbf{1}) = \max_i(v_i + c) = c + \max_0(\boldsymbol{v}) \,,$$

which proves the property.

In the case $\Omega = -H$, we have:

$$\begin{aligned} \max_{-H}(\boldsymbol{v} + c\mathbf{1}) &= \log \sum_i \exp(v_i + c) \\ &= \log \sum_i \exp(v_i) \exp(c) \\ &= \log \left(\exp(c) \sum_i \exp(v_i) \right) \\ &= c + \log \sum_i \exp(v_i) \\ &= c + \max_{-H}(\boldsymbol{v}) \,, \end{aligned}$$

which proves the property.

A.3 Correctness Proof

We now turn to the proof of the recursion.[9] By definition, we have:

$$c_{i,t}(\boldsymbol{w}) = \max_\Omega \left[\textstyle\sum_{j=1}^{i-1} \langle \boldsymbol{w}_j, \phi(\boldsymbol{x})_j \rangle \right]_{\boldsymbol{x} \in X_i | x_i = t} \,,$$

where the regularized maximum is taken on all tag sequences from the beginning until position i, where position i is constrained to be tagged with $t \in T$. We can split the sum by extracting the term that weights the last transition:

$$= \max_\Omega \left[\begin{array}{l} \displaystyle\sum_{j=1}^{i-2} \langle \boldsymbol{w}_j, \phi(\boldsymbol{x})_j \rangle \\ + \langle \boldsymbol{w}_{i-1}, \phi(\boldsymbol{x})_{i-1} \rangle \end{array} \right]_{\boldsymbol{x} \in X_i | x_i = t} \,,$$

[8] More precisely, what we describe here is exactly their proof, but under our notation to make our paper self-contained for newcomers.

[9] We stress out that this proof is only true for $\Omega \in \{0, -H\}$, see (Mensch and Blondel, 2018, Proposition 2)

which can be equivalently written as a regularized maximum over sequences in X_{i-1} as follows:

$$= \max_\Omega \left[\begin{array}{l} \sum_{j=1}^{i-2} \langle \boldsymbol{w}_j, \phi(\boldsymbol{x})_j \rangle \\ + \boldsymbol{w}_{i-1,x_{i-1},t} \end{array} \right]_{\boldsymbol{x} \in X_{i-1}}.$$

Thank to associativity property of $\max_\Omega$, we can "split" the operation over $|T|$ regularized maximums plus one that aggregates the results:

$$= \max_\Omega \left[\max_\Omega \left[\begin{array}{l} \sum_{j=1}^{i-2} \langle \boldsymbol{w}_j, \phi(\boldsymbol{x})_j \rangle \\ + \boldsymbol{w}_{i-1,t',t} \end{array} \right]_{\substack{\boldsymbol{x} \in X_{i-1} \\ |x_{i-1}=t'}} \right]_{t' \in T}.$$

Finally, using distributivity over addition we extract the term $\boldsymbol{w}_{i-1,t',t}$ and obtain the recursive formulation used by the dynamic programming algorithm:

$$= \max_\Omega \left[\begin{array}{l} \boldsymbol{w}_{i-1,t',t} \\ + \max_\Omega \left[\sum_{j=1}^{i-2} \langle \boldsymbol{w}_j, \phi(\boldsymbol{x})_j \rangle \right]_{\substack{\boldsymbol{x} \in X_{i-1} \\ |x_{i-1}=t'}} \end{array} \right]_{t' \in T}$$

$$= \max_\Omega \left[\boldsymbol{w}_{i-1,t',t} + c_{i-1,y'}(\boldsymbol{w}) \right]_{t' \in T}.$$

B Variational Formulation of the Log-Partition Function

Let $\boldsymbol{v} \in \mathbb{R}^d$ be a vector. In this appendix, we prove the variational formulation of the log-partition function, that is:

$$\begin{aligned} \log \sum_i \exp v_i &= \max_{\boldsymbol{\mu} \in \triangle(d)} \langle \boldsymbol{v}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{\mu}, \log \boldsymbol{\mu} \rangle \\ &= \max\nolimits_{-H}(\boldsymbol{v}), \end{aligned}$$

or, in other words, that the Shannon entropy function restricted to the simplex is the Fenchel conjugate of the *logsumexp* function. This results is well-known, see (Boyd and Vandenberghe, 2004, Example 3.25) and (Beck, 2017, Section 4.4.10), among others.

We start by explicitly writing the mathematical program:

$$\begin{aligned} \max_{\boldsymbol{\mu} \in \mathbb{R}^d} \ & \langle \boldsymbol{v}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{\mu}, \log \boldsymbol{\mu} \rangle \\ \text{s.t.} \ & \langle \boldsymbol{\mu}, \mathbf{1} \rangle = 1 \\ & \boldsymbol{\mu} \geq \mathbf{0} \end{aligned}$$

Dualizing the constraints gives the following Lagrangian function:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}, \lambda, \boldsymbol{\nu}) = & \langle \boldsymbol{v}, \boldsymbol{\mu} \rangle - \langle \boldsymbol{\mu}, \log \boldsymbol{\mu} \rangle \\ & + \lambda(1 - \langle \boldsymbol{\mu}, \mathbf{1} \rangle) + \langle \boldsymbol{\nu}, \boldsymbol{\mu} \rangle, \end{aligned}$$

where $\lambda \in \mathbb{R}$ and $\boldsymbol{\nu} \in \mathbb{R}^d_{\geq 0}$ are dual variables associated with the equality and inequalities, respectively.

As the objective is concave and all constraints are linear, $\widehat{\boldsymbol{\mu}}$, $\widehat{\lambda}$ and $\widehat{\boldsymbol{\nu}}$ are optimal primal and dual variables if and only if they satisfy the KKT conditions (Boyd and Vandenberghe, 2004, Sec. 5.5.3). By stationarity, we have:

$$\begin{aligned} \frac{\partial}{\partial \widehat{\mu}_i} \mathcal{L}(\widehat{\boldsymbol{\mu}}, \widehat{\lambda}, \widehat{\boldsymbol{\nu}}) &= 0 \\ \log \widehat{\mu}_i &= v_i - \widehat{\lambda} + \widehat{\mu}_i \\ \widehat{\mu}_i &= \exp(v_i - \widehat{\lambda} + \widehat{\mu}_i). \end{aligned}$$

Note that $\widehat{\mu}_i > 0$ and by complementary slackness we must have $\widehat{\nu}_i \widehat{\mu}_i = 0$, therefore $\widehat{\nu}_i = 0$, and we can write:

$$\widehat{\mu}_i = \frac{\exp(v_i)}{\exp(\widehat{\lambda})}.$$

By primal feasilibty, we have:

$$\begin{aligned} \langle \widehat{\boldsymbol{\mu}}, \mathbf{1} \rangle &= 1 \\ \sum_i \frac{\exp(v_i)}{\exp(\widehat{\lambda})} &= 1 \\ \exp(\widehat{\lambda}) &= \sum_i \exp(v_i), \end{aligned}$$

and therefore:

$$\widehat{\mu}_i = \frac{\exp(v_i)}{\sum_j \exp(v_j)}.$$

Plugging the optimal primal variables in the objective, we obtain:

$$\langle \boldsymbol{v}, \widehat{\boldsymbol{\mu}} \rangle - \langle \widehat{\boldsymbol{\mu}}, \log \widehat{\boldsymbol{\mu}} \rangle = \log \sum_i \exp v_i.$$

C Subgradient of the Fenchel Conjugate

First, note that Eq. (6) can be rewritten as the gradient of a Fenchel conjugate. Indeed, the log-partition function can be rewritten as:

$$\begin{aligned} A_Y(\boldsymbol{w}) &= \max_{\boldsymbol{q} \in \operatorname{conv}(Y)} \langle \boldsymbol{q}, \boldsymbol{w} \rangle - R(\boldsymbol{q}) \\ &= \left(-R + \delta_{\operatorname{conv}(Y)} \right)^* (\boldsymbol{w}). \end{aligned}$$

Therefore, we give a simple proof of the (sub)gradient of the Fenchel conjugate, which is used to derive the formula of marginal probabilities for CRF and BCRF. Although this can be proved via Danskin's theorem (Danskin, 1966; Bertsekas, 1999), we give here a simple alternate proof.

Let $h : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be a function. We first note that the Fenchel conjugate of h is convex as it is the maximum of a set of affine functions, no matter if f is convex or not.

Proposition 4. Let $h : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ be a function and $\boldsymbol{v} \in \operatorname{dom} h^*$. Then, the following formula can be used to build a subgradient of h^* at $\boldsymbol{v}$:

$$\partial h^*(\boldsymbol{v}) \supseteq \operatorname*{argmax}_{\boldsymbol{t} \in \operatorname{dom} h} \langle \boldsymbol{t}, \boldsymbol{v} \rangle - h(\boldsymbol{t}).$$

Moreover, if h^* is differentiable at $\boldsymbol{v}$, ∂h^* is a singleton (Beck, 2017, Th. 3.33).

Proof. Let $\widehat{\boldsymbol{t}}$ be defined as follows:

$$\widehat{\boldsymbol{t}} \in \operatorname*{argmax}_{\boldsymbol{t} \in \operatorname{dom} h} \langle \boldsymbol{t}, \boldsymbol{v} \rangle - h(\boldsymbol{t}).$$

We have $\widehat{\boldsymbol{t}} \in \partial h^*(\boldsymbol{v})$ if and only the subgradient inequality holds (Beck, 2017, Def. 3.1), that is:

$$\forall \boldsymbol{v}' \in \operatorname{dom} h^* : h^*(\boldsymbol{v}') \geq h^*(\boldsymbol{v}) + \langle \widehat{\boldsymbol{t}}, \boldsymbol{v}' - \boldsymbol{v} \rangle.$$

Starting from the right-hand side, for all $\boldsymbol{v}' \in \operatorname{dom} h^*$, we can write:

$$\begin{aligned} & h^*(\boldsymbol{v}) + \langle \widehat{\boldsymbol{t}}, \boldsymbol{v}' - \boldsymbol{v} \rangle \\ &= \cancel{\langle \widehat{\boldsymbol{t}}, \boldsymbol{v} \rangle} - h(\widehat{\boldsymbol{t}}) + \langle \widehat{\boldsymbol{t}}, \boldsymbol{v}' \rangle - \cancel{\langle \widehat{\boldsymbol{t}}, \boldsymbol{v} \rangle} \end{aligned}$$

where we simply replaced $h^*(\boldsymbol{v})$ by its definition using the fact that $\widehat{\boldsymbol{t}}$ is a solution of the maximization problem. We derive an upper bound on this formula by maximizing over possible values for $\boldsymbol{t}$:

$$\begin{aligned} &= \langle \widehat{\boldsymbol{t}}, \boldsymbol{v}' \rangle - h(\widehat{\boldsymbol{t}}) \\ &\leq \max_{\boldsymbol{t} \in \operatorname{dom} h} \langle \boldsymbol{t}, \boldsymbol{v}' \rangle - h(\boldsymbol{t}) \\ &= h^*(\boldsymbol{v}') \end{aligned}$$

Hence the subgradient inequality holds, and $\widehat{\boldsymbol{t}}$ is a subgradient of h^* at $\boldsymbol{v}$. □

D Proof of Proposition 1

Linear programming with entropic regularization is a well-studied setting (Fang, 1992; Fang et al., 1997). Nonetheless, we adapt the proof of (Peyré and Cuturi, 2019, Prop. 4.1) to our problem for completness.

Proof. In this proof, we will write:

$$\begin{aligned} \widehat{\boldsymbol{y}} &\in \operatorname*{argmax}_{\boldsymbol{y} \in \operatorname{conv}(Y)} \langle \boldsymbol{w}, \boldsymbol{y} \rangle \\ \text{and} \quad \widehat{\boldsymbol{\mu}}^{(\tau)} &= \operatorname*{argmax}_{\boldsymbol{y} \in \operatorname{conv}(Y)} \langle \boldsymbol{w}, \boldsymbol{y} \rangle + \tau H(\boldsymbol{y}) \\ &= \nabla B_Y(\tau^{-1} \boldsymbol{w}). \end{aligned}$$

By optimality of $\widehat{\boldsymbol{y}}$, we have:

$$\begin{aligned} & \langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle \geq \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle \\ \iff \quad & 0 \leq \langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle - \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle. \end{aligned}$$

Similarly, by optimality of $\widehat{\boldsymbol{\mu}}^{(\tau)}$:

$$\begin{aligned} & \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle + \tau H(\widehat{\boldsymbol{\mu}}^{(\tau)}) \geq \langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle + \tau H(\widehat{\boldsymbol{y}}) \\ \iff \quad & \langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle - \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle \leq \tau (H(\widehat{\boldsymbol{\mu}}^{(\tau)}) - H(\widehat{\boldsymbol{y}})) \end{aligned}$$

Combining the two inequalities, we have:

$$0 \leq \langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle - \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle \leq \tau \underbrace{(H(\widehat{\boldsymbol{\mu}}^{(\tau)}) - H(\widehat{\boldsymbol{y}}))}_{\geq 0}$$

Notice that for any $\tau > 0$, we have $H(\widehat{\boldsymbol{\mu}}^{(\tau)}) \leq c$ where $c > 0$ is a constant. Therefore:

$$\lim_{\tau \to 0} \tau (H(\widehat{\boldsymbol{\mu}}^{(\tau)}) - H(\widehat{\boldsymbol{y}})) = 0.$$

We can therefore apply the squeeze theorem:

$$\begin{aligned} & \lim_{\tau \to 0} \left(\langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle - \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle \right) = 0 \\ \iff \quad & \lim_{\tau \to 0} \langle \boldsymbol{w}, \widehat{\boldsymbol{\mu}}^{(\tau)} \rangle = \langle \boldsymbol{w}, \widehat{\boldsymbol{y}} \rangle, \end{aligned}$$

which ends the proof. □

E Proof of Proposition 2

Proof. Denote by $\mathcal{P}$ the polytope on the right-hand side of the theorem statement. Since there is a one-to-one correspondence between valid paths and binary points of $\mathcal{P}$, it remains to prove that $\mathcal{P}$ is integral.

Consider the directed graph $G' = (V', E')$ obtained from G by adding two nodes s and t, an arc (s, v) for all $\mathrm{v} \in V_1$, an arc (v, t) for $\mathrm{v} \in V_n$, and the arc (t, s). By construction, every point $\overline{\boldsymbol{y}}$ of $\mathcal{P}$ can be extended to a point of the following polytope:

$$\mathcal{Q} = \left\{ y \in \mathbb{R}_+^{E'} \;\middle|\; \begin{array}{ll} y_{ts} = 1, & \\ \forall v \in V' : & \sum_{a \in \delta^+(\mathrm{v})} y_a \\ & = \sum_{a \in \delta^-(\mathrm{v})} y_a \end{array} \right\}$$

by setting $\overline{y}_{(t,s)} = 1$, $\overline{y}_{(s,\mathrm{v})} = \sum_{a \in \delta^+(\mathrm{v})} \overline{y}_a$ for all $\mathrm{v} \in V_1$, and $\overline{y}_{(\mathrm{v},t)} = \sum_{a \in \delta^-(\mathrm{v})} \overline{y}_a$ for all $\mathrm{v} \in V_n$. Hence, $\mathcal{Q}$ is an extended formulation of $\mathcal{P}$ (Conforti et al., 2013). Since $\mathcal{Q}$ is integral (Schrijver, 1986, p. 274), so is $\mathcal{P}$. □

F Iterative Bregman Projections

Recall that $\mathrm{conv}(Y) = \mathcal{C}_{\text{even}} \cap \mathcal{C}_{\text{odd}}$ and solving the projections on $\mathcal{C}_{\text{even}}$ and $\mathcal{C}_{\text{odd}}$ decompose into solving several problems of the form:

$$\min_{\boldsymbol{q}|_{\delta(V_i)} \in \mathcal{C}_i} D_{\mathrm{KL}}\left(\boldsymbol{q}|_{\delta(V_i)}, \boldsymbol{w}|_{\delta(V_i)}\right),$$

for some $i \in [\![2, n-1]\!]$. This latter problem can be reformulated as:

$$\max_{\boldsymbol{q}} \sum_{a \in \delta(V_i)} \left((w_a + 1)q_a - q_a \log q_a\right) \tag{9}$$

$$\text{s.t.} \sum_{a \in \delta^-(\mathrm{v})} q_a = \sum_{a \in \delta^+(\mathrm{v})} q_a, \quad \forall \mathrm{v} \in V_i, \tag{10}$$

$$\sum_{a \in \delta^-(V_i)} q_a = 1, \tag{11}$$

$$\boldsymbol{q} \geq \mathbf{0} \tag{12}$$

Therefore, in the following we show that this problem has a closed-form expression.

Dualizing Constraints (10) and (11) gives the following Lagrangian function:[10]

$$\begin{aligned}\mathcal{L}(\boldsymbol{q}, \boldsymbol{\lambda}, \nu) = &\sum_{a \in \delta(V_i)} \left((w_a + 1)q_a - q_a \log q_a\right) \\ &+ \sum_{\mathrm{v} \in V_i} \lambda_v \left(\sum_{a \in \delta^-(\mathrm{v})} q_a - \sum_{a \in \delta^+(\mathrm{v})} q_a\right) \\ &+ \nu \left(\sum_{a \in \delta^-(V_i)} q_a - 1\right).\end{aligned}$$

For fixed $\boldsymbol{\lambda}$ and ν, the associated Lagrangian relaxed problem $\mathcal{L}(\boldsymbol{\lambda}, \nu)$ is:

$$\mathcal{L}(\boldsymbol{\lambda}, \nu) = \max_{\boldsymbol{q}} \mathcal{L}(\boldsymbol{q}, \boldsymbol{\lambda}, \nu), \tag{13}$$

and the Lagrangian dual problem $\mathcal{L}$ is:

$$\mathcal{L} = \min_{\boldsymbol{\lambda}, \nu} \mathcal{L}(\boldsymbol{\lambda}, \nu). \tag{14}$$

Since (9) is concave, strong duality holds, that is, the optimum of (9)-(12) equals the one of $\mathcal{L}$. Moreover, an optimal solution $\widehat{\boldsymbol{q}}$ of (9)-(12) and an optimal $\widehat{\boldsymbol{\lambda}}, \widehat{\nu}$ of $\mathcal{L}$ satisfy the KKT conditions (Boyd and Vandenberghe, 2004, Sec. 5.5.3).

The stationarity condition for $a \in \delta^-(\mathrm{v})$ implies that:

$$\frac{\partial}{\partial q_a} \mathcal{L}(\widehat{\boldsymbol{q}}, \widehat{\boldsymbol{\lambda}}, \widehat{\nu}) = 0$$

$$\log \widehat{q}_a = w_a + \widehat{\lambda}_{\mathrm{v}} + \widehat{\nu}$$

$$\widehat{q}_a = \exp(w_a) \exp(\widehat{\lambda}_{\mathrm{v}}) \exp(\widehat{\nu}). \tag{15}$$

Similarly, the stationarity condition for $a \in \delta^+(\mathrm{v})$ implies that:

$$\frac{\partial}{\partial q_a} \mathcal{L}(\widehat{\boldsymbol{q}}, \widehat{\boldsymbol{\lambda}}, \widehat{\nu}) = 0$$

$$\log \widehat{q}_a = w_a - \widehat{\lambda}_{\mathrm{v}}$$

$$\widehat{q}_a = \frac{\exp(w_a)}{\exp(\widehat{\lambda}_{\mathrm{v}})} \tag{16}$$

At optimality, $\widehat{\boldsymbol{q}}$ is primal feasible. Hence, it satisfies (10) so we have for all $\mathrm{v} \in V_i$:

$$\sum_{a \in \delta^-(\mathrm{v})} \exp(w_a) \exp(\widehat{\lambda}_{\mathrm{v}}) \exp(\widehat{\nu}) = \sum_{a \in \delta^+(\mathrm{v})} \frac{\exp(w_a)}{\exp(\widehat{\lambda}_{\mathrm{v}})},$$

which gives:

$$\widehat{\lambda}_{\mathrm{v}} = \frac{1}{2} \log w^+(\mathrm{v}) - \frac{1}{2} \log w^-(\mathrm{v}) - \frac{1}{2} \widehat{\nu}, \tag{17}$$

where:

$$w^+(\mathrm{v}) = \sum_{a \in \delta^+(\mathrm{v})} \exp(w_a)$$

$$\text{and } w^-(\mathrm{v}) = \sum_{a \in \delta^-(\mathrm{v})} \exp(w_a).$$

Since $\widehat{\boldsymbol{q}}$ satisfies (11), we have

$$\sum_{\mathrm{v} \in V_i} \sum_{a \in \delta^-(\mathrm{v})} \exp(w_a) \exp(\widehat{\lambda}_{\mathrm{v}}) \exp(\widehat{\nu}) = 1.$$

By (17), we get:

$$\exp\left(-\frac{1}{2}\widehat{\nu}\right) = \sum_{\mathrm{v} \in V_i} w^-(\mathrm{v}) \exp\left(\sigma(\mathrm{v})\right)$$

where:

$$\sigma(\mathrm{v}) = \frac{1}{2} \log w^+(\mathrm{v}) - \frac{1}{2} \log w^-(\mathrm{v}).$$

for all $\mathrm{v} \in V_i$. This gives:

$$\widehat{\nu} = -2 \log \left(\sum_{\mathrm{v} \in V_i} w^-(\mathrm{v}) \exp(\sigma(\mathrm{v}))\right). \tag{18}$$

Finally, replacing $\widehat{\boldsymbol{\lambda}}$ and $\widehat{\nu}$ by their value given by (17) and (18) in the formulas (15) and (16) gives:

$$\widehat{q}_{(u,u')} = \begin{cases} \dfrac{\exp(w_{(u,u')})}{\exp(\sigma(u))\left(\sum_{\mathrm{v} \in V_i} w^-(\mathrm{v}) \exp(\sigma(\mathrm{v}))\right)} & \text{if } u \in V_i, \\ \dfrac{\exp(w_{(u,u')}) \exp(\sigma(u'))}{\sum_{\mathrm{v} \in V_i} w^-(\mathrm{v}) \exp(\sigma(\mathrm{v}))} & \text{if } u' \in V_i, \end{cases}$$

for all $(u, u') \in \delta(V_i)$.

[10] For a sake of simplicity, we ignore inequalities $\boldsymbol{q} \geq \mathbf{0}$ as by (15) and (16), $\boldsymbol{q}$ is nonnegative for all $\boldsymbol{\lambda}, \nu$, following similar argument to the one in Appendix B.

G Mean Field Theory

Mean field (MF) approximation for CRF (Wainwright and Jordan, 2008; Zheng et al., 2015; Wang et al., 2020) is a general method which aims at approximating a MRF with a tractable distribution, in the case of tagging approximate a linear-chain CRF with a factorized distribution (the so-called *naive* MF) of the form:

$$r(\boldsymbol{x}|\boldsymbol{s}) = \prod_i r_i(x_i|\boldsymbol{s}).$$

The main idea is to search for the factorized distribution that is the closest to the CRF distribution $p(\cdot|\boldsymbol{s})$ in terms of their Kullback-Leibler divergence (Wainwright and Jordan, 2008, Chap. 5.2.2):

$$\operatorname*{argmin}_r D_{KL}(r(\cdot|\boldsymbol{s}), p(\cdot|\boldsymbol{s})).$$

Even if it's not necessarily obvious at first glance, the key advantage of this approach is that the log-partition term A_Y appears as a constant in the objective, hence there is no need to rely on the forward algorithm to search for the optimal MF distribution.

MF inference is often implemented using iterative parallel updates. Starting from a any distribution r^0, each iteration computes an updated distribution as follows:

$$r_i^k(t) \leftarrow \frac{\exp\big(m(i,t,k)\big)}{\sum_{t'} \exp\big(m(i,t',k)\big)},$$

where variables m are "messages" incoming from adjacent tag distributions defined as follows::

$$\begin{aligned} m_{\rightarrow}(i,t,k) &= \mathbb{E}_{t' \sim r_{i-1}^{k-1}(\cdot)}[f_{\boldsymbol{\theta}}(\boldsymbol{s})_{i-1,t',t}] \\ m_{\leftarrow}(i,t,k) &= \mathbb{E}_{t' \sim r_{i+1}^{k-1}(\cdot)}[f_{\boldsymbol{\theta}}(\boldsymbol{s})_{i,t,t'}] \\ m(i,t,k) &= m_{\rightarrow}(i,t,k) + m_{\leftarrow}(i,t,k). \end{aligned}$$

Since the MF inference objective is non-convex, the quality of this method relies on the initial distribution. Moreover, the parallel update procedure is not guaranteed to converge (Kraehenbuehl and Koltun, 2013).

Note that given the update formulas, MF cannot take into account well-formedness constraints, *i.e.* we cannot forbid adjacent pair of tags by removing transitions or setting transition weights to $-\infty$.

H Partial Fenchel-Young Losses

In this appendix, we prove several properties of partial FY losses that motivate their use for learning from partial labels.

Generalization of FY losses. If $\widetilde{Y} = \{\boldsymbol{y}\}$ is a singleton containing only a gold label, then $\widetilde{\ell}_\Omega(\boldsymbol{w}, \{\boldsymbol{y}\}) = \ell_\Omega(\boldsymbol{w}, \boldsymbol{y})$.

Proof. When $\widetilde{Y} = \{\boldsymbol{y}\}$ is a singleton, we have $\operatorname{conv}(\widetilde{Y}) = \{\boldsymbol{y}\}$ and therefore:

$$\begin{aligned} &\widetilde{\ell}_\Omega(\boldsymbol{w}, \{\boldsymbol{y}\}) \\ &= \big(\Omega + \delta_{\operatorname{conv}(Y)}\big)^*(\boldsymbol{w}) - \big(\Omega + \delta_{\{\boldsymbol{y}\}}\big)^*(\boldsymbol{w}) \\ &= \big(\Omega + \delta_{\operatorname{conv}(Y)}\big)^*(\boldsymbol{w}) \\ &\quad - \left(\sup_{\boldsymbol{\mu} \in \operatorname{dom}\Omega} \langle \boldsymbol{\mu}, \boldsymbol{w} \rangle - \Omega(\boldsymbol{\mu}) - \delta_{\{\boldsymbol{y}\}}(\boldsymbol{\mu}) \right). \end{aligned}$$

Note that the indicator function restrict the search space of the maximization to a single element $\boldsymbol{y}$, therefore:

$$\begin{aligned} &= \Omega^*(\boldsymbol{w}) - \langle \boldsymbol{\mu}, \boldsymbol{y} \rangle + \Omega(\boldsymbol{y}) \\ &= \ell_\Omega(\boldsymbol{w}, y), \end{aligned}$$

which ends the proof. □

Non-negativity. The loss is bounded below by 0.

Proof. Note that we have:

$$\begin{aligned} &\Big(\Omega + \delta_{\operatorname{conv}(\widetilde{Y})}\Big)^*(\boldsymbol{w}) \\ &= \sup_{\boldsymbol{\mu} \in \operatorname{dom}\Omega} \langle \boldsymbol{\mu}, \boldsymbol{w} \rangle - \Omega(\boldsymbol{\mu}) - \delta_{\operatorname{conv}(\widetilde{Y})}(\boldsymbol{\mu}). \end{aligned}$$

The indicator function act as a constraint on $\boldsymbol{\mu}$. By definition we have $\widetilde{Y} \subset Y$, and therefore by increasing the search space we derive an upper bound:

$$\begin{aligned} &\leq \sup_{\boldsymbol{\mu} \in \operatorname{dom}\Omega} \langle \boldsymbol{\mu}, \boldsymbol{w} \rangle - \Omega(\boldsymbol{\mu}) - \delta_{\operatorname{conv}(Y)}(\boldsymbol{\mu}) \\ &= \big(\Omega + \delta_{\operatorname{conv}(Y)}\big)^*(\boldsymbol{w}). \end{aligned}$$

Therefore, the loss is non-negative. □

Smaller partial labeling set $\implies$ bigger loss. Let $\widetilde{Y}' \subseteq \widetilde{Y}$, then:

$$\widetilde{\ell}(\boldsymbol{w}; \widetilde{Y}') \geq \widetilde{\ell}(\boldsymbol{w}; \widetilde{Y}).$$

Proof. Note that if $\widetilde{Y}' \subseteq \widetilde{Y}$, then $\operatorname{conv}(\widetilde{Y}') \subseteq \operatorname{conv}(\widetilde{Y})$. By definition, we have:

$$\begin{aligned} &\widetilde{\ell}_\Omega(\boldsymbol{w}; \widetilde{Y}') \\ &= \big(\Omega + \delta_{\operatorname{conv}(Y)}\big)^*(\boldsymbol{w}) - \Big(\Omega + \delta_{\operatorname{conv}(\widetilde{Y}')}\Big)^*(\boldsymbol{w}) \\ &= \big(\Omega + \delta_{\operatorname{conv}(Y)}\big)^*(\boldsymbol{w}) \end{aligned}$$

$$- \left(\sup_{\boldsymbol{\mu} \in \operatorname{dom} \Omega} \langle \boldsymbol{\mu}, \boldsymbol{w} \rangle - \Omega(\boldsymbol{\mu}) - \delta_{\operatorname{conv}(\widetilde{Y}')}(\boldsymbol{\mu}) \right).$$

If we maximize over a larger set, the second term will be larger therefore:

$$\begin{aligned}
&\geq \left(\Omega + \delta_{\operatorname{conv}(Y)}\right)^* (\boldsymbol{w}) \\
&\quad - \left(\sup_{\boldsymbol{\mu} \in \operatorname{dom} \Omega} \langle \boldsymbol{\mu}, \boldsymbol{w} \rangle - \Omega(\boldsymbol{\mu}) - \delta_{\operatorname{conv}(\widetilde{Y})}(\boldsymbol{\mu}) \right) \\
&= \widetilde{\ell}_\Omega(\boldsymbol{w}; \widetilde{Y}),
\end{aligned}$$

which ends the proof. □

I Neural Network Hyperparameters

We use standard self-attentive networks with embedings of size 768, 8 heads per layers and hidden dimension projection of 2048. The model are training using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 3×10^{-4} when the transformer is learned from scratch, and 3×10^{-5} when fine-tuning BERT. We use a linear rate scheduler with warmup on 10% of updates.

For part-of-speech tagging, we sum character-level embedding to word-level embedding at the input of the transformer, where the character level embeddings are obtained using a simple 1D convolution layer.