A Fast Variational Approach for Learning Markov Random Field Language Models

Yacine Jernite JERNITE@CS.NYU.EDU
CIMS, New York University, 251 Mercer Street, New York, NY 10012, USA

Alexander M. Rush SRUSH@SEAS.HARVARD.EDU
Facebook AI Research, 770 Broadway, New York, NY 10003, USA

David Sontag DSONTAG@CS.NYU.EDU
CIMS, New York University, 251 Mercer Street, New York, NY 10012, USA

Abstract

Language modelling is a fundamental building block of natural language processing. However, in practice the size of the vocabulary limits the distributions applicable for this task: specifically, one has to either resort to local optimization methods, such as those used in neural language models, or work with heavily constrained distributions. In this work, we take a step towards overcoming these difficulties. We present a method for global-likelihood optimization of a Markov random field language model exploiting long-range contexts in time independent of the corpus size. We take a variational approach to optimizing the likelihood and exploit underlying symmetries to greatly simplify learning. We demonstrate the efficiency of this method both for language modelling and for part-of-speech tagging.

1. Introduction

The aim of language modelling is to estimate a distribution over words that best represents the text of a corpus. Language models are central to tasks such as speech recognition, machine translation, and text generation, and the parameters of these models are commonly used as features or as initialization for other algorithms. Examples include the word distributions learned by topic models, or the word embeddings learned through neural language models.

Central to the language modelling problem is the challenge of scale. It is typical for languages to have vocabularies of hundreds of thousands of word types, and language models themselves are often estimated on corpora with billions of tokens (Graff et al., 2003). The scale of the problem inherently limits the types of distributions that can be effectively applied.

In practice, the most commonly used class of language models are n-gram models. These represent the probability of the next word as a multinomial distribution conditioned on the previous context. The parameters of this class of models can be very efficiently estimated by simply collecting sufficient statistics and tuning a small set of parameters.

More recently neural language models (NLMs) have gained popularity (Bengio et al., 2006; Mnih & Hinton, 2007). These models estimate the same distribution as n-gram models, but utilize a non-linear neural network parameterization. NLMs have been shown to produce competitive results with n-gram models using many fewer parameters. Additionally the parameters themselves have proven to be useful for other language tasks (Collobert et al., 2011). Unfortunately training NLMs can be much slower than n-gram models, often requiring expensive gradient computations for each token; techniques have been developed to speed up training in practice (Mnih & Hinton, 2009; Mnih & Teh, 2012).

In this work, we consider a different class of language models. Instead of estimating the local probability of the next word given its context, we globally model the entire corpus as a Markov random field (MRF) language model. Undirected graphical models like MRFs have been widely applied in natural language processing as a way to flexibly model statistical dependencies; however, MRFs are rarely used for language modelling since estimating their parameters requires computing a costly partition function.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

Our contribution is to provide a simple to implement algorithm for very efficiently estimating this class of models. We take a variational approach to the optimization problem, and devise a lower bound on the log-likelihood using *lifted inference*. By exploiting the problem's symmetry, we derive an efficient approximation of the partition function.

Crucially, each step of the final algorithm has time complexity of $O(KC^2)$ where K is the size of the n-gram context and C is the size of the vocabulary. Note that besides collecting statistics, this algorithm has no time dependence on the number of tokens, potentially allowing its estimation speed to scale similarly to n-gram models.

Experimentally, we demonstrate the quality of the models learned by our algorithm by applying it to a language modelling task. Additionally we show that this same estimation algorithm can be effectively applied to other common sequence modelling tasks such as part-of-speech tagging.

2. Background

Notation We denote sequences by bold variables: $\mathbf{t} = (t_1, \ldots, t_n)$. A sub-sequence will be defined as either $\mathbf{t}_i^j = (t_i, t_{i+1}, \ldots, t_j)$ or $\mathbf{t}^{-i} = (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$.

Contextual language models Let us first define the class of contextual language models as the set of distributions over words conditioned on a fixed-length left context. Formally this is an estimate of $p(t_i|\mathbf{t}_{i-K}^{i-1})$ where t_i is the current word and K is the size of the context window. For a basic n-gram language model, this is simply a multinomial distribution, and the maximum-likelihood estimate can be computed in closed-form from the statistics of the corpus (although in practice some smoothing is often employed).

A neural (probabilistic) language model (NLM) is a contextual language model where the word probability is a non-linear function of the context estimated from a neural network. In this work, we will focus specifically on the class of NLMs with potentials that are bilinear in the context and predicted word, such as the log-bilinear language model of Mnih & Hinton (2007). This model is parametrized as:

$$p(t_i|\mathbf{t}_{i-K}^{i-1}) = \frac{\exp\left(\left(\sum_{l=1}^{K} U_{t_{i-l}} R^l\right) W_{t_i}^\top\right)}{Z(\mathbf{t}_{i-K}^{i-1})} \quad (1)$$

where $U \in \mathbb{R}^{C \times D}$, $W \in \mathbb{R}^{C \times D}$, and $R \in \mathbb{R}^{K \times D \times D}$ are the parameters of the model, and $Z(\mathbf{t}_{i-K}^{i-1})$ is a local normalization function (dependent on the context). Specifically, U_{t_i} and W_{t_i} are the left and right embeddings respectively of token t_i, R^l is a distance-dependent transition matrix, C is the size of the vocabulary and $D \ll C$ is the size of the low-rank word embeddings. Another form of bilinear model is the variant used in Word2Vec (Mikolov et al., 2013):

$$p(t_i|\mathbf{t}_{i-K}^{i-1}, \mathbf{t}_{i+1}^{i+K}) = \frac{\exp\left(\sum_{\substack{l=-K \\ l \neq 0}}^{K} U_{t_{i+l}} W_{t_i}^\top\right)}{Z(\mathbf{t}_{i-K}^{i-1}, \mathbf{t}_{i+1}^{i+K})} \quad (2)$$

These models have been shown to give similar results to n-gram models while providing useful word representations. However, the local normalization means that most optimization methods need to look at one token at a time, and scale at least linearly with the size of the corpus.

Markov random fields To avoid this issue of local normalization, we model the entire corpus as a sequence of random variables $T_1 \ldots T_N$, for which we give a joint, globally normalized distribution. We specify this distribution with a Markov random field.

A Markov random field is defined by a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a set of potentials $(\theta^c)_{c \in \mathcal{C}}$, where $\mathcal{C}$ is defined as the set of cliques in graph $\mathcal{G}$.

Let $\mathbf{t}$ denote a specific assignment of $T_1 \ldots T_N$, and let $\mathbf{t}_c = (t_i)_{i \in c}$. The log-probability of a sequence $\mathbf{t}$ is then:

$$\log(p(\mathbf{t}; \theta)) = \sum_{c \in \mathcal{C}} \theta^c_{\mathbf{t}_c} - A(\theta)$$

where $A(\theta)$ is called the log-partition function, and can in general be computed exactly with a complexity exponential in the size of the tree-width of $\mathcal{G}$.

3. MRF Language Models

Sentence model Building on this formalism, let us now define a family of MRF distributions over text. We start by considering a sequence $\mathbf{x} = (x_1, \ldots, x_n)$ of n variables with state space $\mathcal{X}$. We define an order K Markov sequence model as a Markov Random field where each element of the sequence is connected to its K left and right neighbours.

For simplicity of exposition, we restrict our description in the rest of this paper to pairwise Markov sequence models, for which only cliques of size 2 (edges) have potentials: $\theta = \{\theta^{(i,j)} | (i,j) \in \mathcal{E}\}$. The lifted inference method, however, can be easily extended to higher-order potentials.

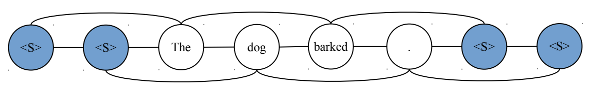

Figure 1. The sentence distribution model for $M = 4$.

Following the notation introduced in Section 2, this gives us the following distribution $p^n_{\text{seq}_K}$ over $\mathcal{X}^n$:

$$\forall \mathbf{x} \in \mathcal{X}^n, \quad \log(p^n_{\text{seq}_K}(\mathbf{x})) = \sum_{i=1}^{n-K} \sum_{l=1}^{K} \theta^{(i,i+l)}_{x_i, x_{i+l}} - A(\theta)$$

Let $\mathcal{T}$ denote the vocabulary of our text corpus. In this work, we define the context of a word as its K left and right neighbouring tokens. By adding K "padding" or "separator" tokens $\langle S \rangle \notin \mathcal{T}$ to the left and right of the sentence, this notion of context also allows us to bias the distribution of tokens at the beginning and end of the sentence.

Let $\mathcal{X} = \mathcal{T} \cup \{\langle S \rangle\}$. A sentence $\mathbf{t}$ of length M is then implicitly mapped to a sequence $\mathbf{x}(\mathbf{t}) \in \mathcal{X}^{M+2K}$ by adding the start and end $\langle S \rangle$ tokens. Now, letting $\mathcal{S} = \{\mathbf{x}(\mathbf{t}) | \mathbf{t} \in \mathcal{T}^M\}$, the order K Markov sequence model allows us to define the following distribution over sentences of length M, as illustrated in Figure 1:

$$p^M(\mathbf{t}) = p^{M+2K}_{\text{seq}_K}(\mathbf{x} = \mathbf{x}(\mathbf{t}) | \mathbf{x} \in \mathcal{S}) \tag{3}$$

This gives rise to the following generative model for a sentence:

1. Sample the sentence length $M \sim \tau(M)$,
2. Sample M tokens: $(t_1, \ldots, t_M) \sim p^M(\mathbf{t})$,

where τ is any distribution over integers and can easily be fit to the data. We focus in this work on learning the parameters of p^M.

These graphical models define a large family of log-linear distributions, depending on the value of K and the parametrization of the edge log-potentials θ. In the applications that follow θ will either be defined (and optimized over) explicitly, or represented as a product of low-rank matrices. We will show how to optimize the likelihood of the corpus for both these settings.

Low rank Markov random fields We now consider different low-rank realizations of the log-potentials θ. Suppose that $\theta^{i,j}$ only depends on $|j-i|$, that is to say, parameters are shared across edges of the same length (in which case we shall simply write $\theta^{i,j} = \theta^{|j-i|}$), we can have for example:

$$\theta^{|j-i|}_{t_i,t_j} = U_{t_i} R^{|j-i|} W_{t_j} \tag{4}$$

$$\theta^{|j-i|}_{t_i,t_j} = U_{t_i} W^{|j-i|}_{t_j} \tag{5}$$

One interesting property of these models is that since the Markov blanket of a word consists only of its immediate neighbours, its conditional likelihood can be expressed as:

$$p(t_i | \mathbf{t}^{-\mathbf{i}}) = p(t_i | \mathbf{t}^{i-1}_{i-K}, \mathbf{t}^{i+K}_{i+1}) \propto \exp(\sum_{l=1}^{K} \theta^l_{t_{i-l}, t_i} + \theta^l_{t_i, t_{i+l}})$$

This class of probability functions corresponds to those defined by a bi-directional log-bilinear neural language model. The model in Equation 4 ($\theta = URW$) can easily be rewritten in terms of the bi-directional version of Mnih's LBL (Mnih & Teh, 2012) and the model in Equation 5, which we use in the rest of this work, is a slightly more general (distance-dependent) version of the Word2Vec CBOW model from (Mikolov et al., 2013).

Conversely, log-bilinear NLMs can be seen as optimizing the pseudo-likelihood (defined as $\prod_{i=1}^{n} p(t_i | \mathbf{t}^{-\mathbf{i}})$) of an order K Markov sequence model as defined above. Since the pseudo-likelihood is a consistent estimator of the likelihood (Besag, 1975), we expect our factorization to have properties similar to those of the embeddings learned by log-bilinear neural language models.

4. Efficient Learning Using Lifted Variational Inference

We now outline our method for optimizing the likelihood of a corpus under our class of models. Learning undirected graphical models is challenging because of the global normalization constant, or partition function. We derive a tractable algorithm by using a variational approximation: we define a lower bound on the data likelihood (Wainwright et al., 2005; Yanover et al., 2008), and alternate between finding the tightest version of that bound and taking a gradient ascent step in the parameters of the model.

The novelty of our method comes from the fact that for the bound we define, both the tightening and gradient step only require us to consider K pairwise moments, i.e. the running time of learning will be independent of the size of the corpus. We achieve this by showing how to reduce the learning task to *lifted* variational inference, allowing us to build upon recent work by Bui et al. (2014). We then derive an algorithm to efficiently perform lifted variational inference using belief propagation and dual decomposition. The overall learning algorithm is simple to implement and runs very fast.

4.1. Creating Symmetry using a Cyclic Model

Given a corpus $\mathbf{t^c} = (\mathbf{t}^1, \ldots, \mathbf{t}^{n_c})$ of n_c sentences drawn independently from our model, we wish to maximize its likelihood $p(\mathbf{t^c}) = \prod_{i=1}^{n_c} p(\mathbf{t}^i) = \prod_{i=1}^{n_c} \tau(M_i) p^{M_i}(\mathbf{t}^i)$. We first show how to obtain a symmetric lower bound on the likelihood $\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t}^i)$.

Consider the sequence $\mathbf{x}(\mathbf{t^c}) \in \mathcal{X}^N$ obtained by adding K $\langle S \rangle$ tokens before the first and between any two adjacent sentences in $\mathbf{t^c}$, where $N = \sum_{i=1}^{n_c}(M_i + K)$. Let p_{cycl_K} be the wrapped around version of p_{seq_K}, as illustrated in Figure 2. We have the following result (the proof can be found in the supplementary materials):

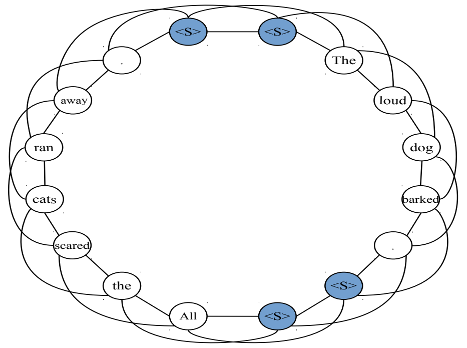

Figure 2. The cyclic model with $N = 16$ and a setting $s \in \mathcal{S}$ corresponding to a corpus of two sentences.

Lemma 1. *Let* $\mathcal{S} = \{\mathbf{x}(\mathbf{t^c}) | \mathbf{t^c} \in \mathcal{T}^{M_1} \times \ldots \times \mathcal{T}^{M_{n_c}}\}$. *Then,*

$$\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t}_i) = \frac{p^N_{cycl_K}(\mathbf{x} = \mathbf{x}(\mathbf{t}))}{p^N_{cycl_K}(\mathbf{x} \in \mathcal{S})}.$$

Hence, the cyclic model $p^N_{\text{cycl}_K}(\mathbf{x}(\mathbf{t^c}))$ provides a lower bound on $p(\mathbf{t^c})$, which happens to be invariant to rotations; the rest of the paper makes use of the symmetry to maximize this lower bound. Let θ^0 denote single node unary potentials, and θ^l the edge potentials as defined in Section 3. The objective we want to optimize is:

$$\log(p^N_{\text{cycl}_K}(\mathbf{x}(\mathbf{t^c}))) = \sum_{i=1}^{N} \left(\theta^0_{x_i} + \sum_{l=1}^{K} \theta^l_{x_i, x_{i+l}} \right) - A(\theta),$$

where

$$A(\theta) = \log(\sum_{\mathbf{y} \in \mathcal{X}^N} \exp(\sum_{i=1}^{N} \left(\theta^0_{y_i} + \sum_{l=1}^{K} \theta^l_{y_i, y_{i+l}} \right)))$$

and $\forall l \in \{1, \ldots, K\}$, $x_{N+l} = x_l$ and $y_{N+l} = y_l$.

4.2. Variational lower bound

Unfortunately, the partition function A is extremely costly to compute for any reasonable vocabulary size, as dynamic programming would have running time $O(NC^{2K+1})$. However, it is easy to formulate upper bounds on A, which give rise to a family of lower bounds on the log-likelihood. We start by using an equivalent variational formulation of the partition function as an optimization problem:

$$A(\theta) = \max_{\mu \in \mathcal{M}} \sum_{i=1}^{N} \left(\langle \mu^i, \theta^0 \rangle + \sum_{l=1}^{K} \langle \mu^{i,i+l}, \theta^l \rangle \right) + H(\mu),$$

where $\mathcal{M}$ denotes the marginal polytope (Wainwright & Jordan, 2008). We then make two approximations to make solving this optimization problem easier. First, we replace $\mathcal{M}$ with the local consistency polytope $\mathcal{LC}$. Since $\mathcal{LC} \supseteq \mathcal{M}$, this gives us an upper bound on the original solution. Second, we replace the entropy $H(\mu)$ with the tree-reweighted (TRW) upper bound (Wainwright et al., 2005):

$$H(\mu) \leq \bar{H}^\rho(\mu) = \sum_{i=1}^{N} \left(H(\mu^i) - \sum_{l=1}^{K} \rho_{i,i+l} I(\mu^{i,i+l}) \right)$$

where $\rho_{i,j}$ denotes the probability of edge (i, j) appearing in a covering set of forests for the MRF. Let:

$$\bar{A}(\theta; \rho) = \max_{\mu \in \mathcal{LC}} \sum_{i=1}^{N} \left(\langle \mu^i, \theta^0 \rangle + \sum_{l=1}^{K} \langle \mu^{i,i+l}, \theta^l \rangle \right) + \bar{H}^\rho(\mu)$$

Using this variational approximation, we now have an upper bound on the log-partition function which can be computed by solving a convex optimization problem. Altogether this then gives us the following tractable lower bound on the log-likelihood:

$$\begin{aligned} \log(p^N_{\text{cycl}_K}(\mathbf{x})) &\geq \sum_{i=1}^{N} \left(\theta^0_{x_i} + \sum_{l=1}^{K} \theta^l_{x_i, t_{x+l}} \right) - \bar{A}(\theta; \rho) \\ &= \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho). \qquad (6) \end{aligned}$$

Learning using gradient ascent then requires that we compute the derivative of $\bar{A}(\theta; \rho)$, which we will show is the μ that maximizes the variational optimization problem (we return to this process in more detail in the next section). We can therefore reduce the learning task to that of repeatedly performing approximate inference using TRW. Fast combinatorial solvers for TRW exist, including tree-reweighted belief propagation (Wainwright et al., 2005), convergent message-passing based on geometric programming (Globerson & Jaakkola, 2007), and dual decomposition (Jancsary & Matz, 2011), which all have complexity linear in the size of the corpus.

However, we next show that by taking advantage of the symmetries present in the optimization problem, it is possible to solve it in time which is independent of N, the number of word tokens in the corpus.

4.3. Lifting the objective

Our key insight is that because of the parameter sharing in our model, each of the random variables in the cyclic model are indistinguishable. More precisely, there is an automorphism group of rotation which can be applied to the sufficient statistic vector and to the model parameters which does not change the joint distribution (Bui et al., 2013).

When such symmetry exists, Bui et al. (2014) show that without loss of generality one can choose the edge appearance probabilities to be symmetric, which in our setting corresponds to choosing a ρ such that $\forall i, j$, $\rho_{i,j} = \rho_{|j-i|}$ (i.e., the tightest TRW upper bound on $A(\theta)$ can be obtained by a symmetric ρ). When the edge appearance probabilities are chosen accordingly, since the objective is strictly concave and the variables are rotationally symmetric, it follows (Bui et al., 2014, Theorem 3) that the optimum must satisfy the following property:

$$\forall 1 \le i, j \le N, 1 \le l \le K, \mu^i = \mu^j \text{ and } \mu^{i,i+l} = \mu^{j,j+l}. \tag{7}$$

We will take advantage of this structural property to dramatically simplify the variational optimization problem. In particular, using the notation μ_V^0 to refer to the single-node marginal (there is only one) and $\mu_E^l(x_1, x_2)$ to refer to the edge marginal corresponding to the potential θ^l, we have:

$$\bar{A}(\theta) = \max_{\mu} N \Big[\langle \mu_V^0, \theta^0 \rangle + H(\mu_V^0) \tag{8}$$
$$+ \sum_{l=1}^{K} \Big(\langle \mu_E^l, \theta^l \rangle - \rho_l I(\mu_E^l) \Big) \Big],$$

where the maximization is subject to the non-negativity constraints $\mu_V^0, \mu_E^1, \ldots, \mu_E^K \ge 0$, sum-to-one constraints $\sum_{x_v} \mu_V^0(x_v) = 1$ and $\forall l$, $\sum_{x_1,x_2} \mu_E^l(x_1, x_2) = 1$, and pairwise consistency constraints:

$$\sum_{x_1} \mu_E^l(x_1, x_2) = \mu_V^0(x_2) \qquad \forall l, x_2, \tag{9}$$
$$\sum_{x_2} \mu_E^l(x_1, x_2) = \mu_V^0(x_1) \qquad \forall l, x_1. \tag{10}$$

The optimal μ_E^l is guaranteed to be symmetric, and so we could have used a slightly more compact form of the optimization problem (c.f. Bui et al., 2014). However, we prefer this form both because it is easier to describe and because it is more amenable to solving efficiently.

The lifted problem, Eq. 8, has only $C + KC^2$ optimization variables, instead of the $N(C + KC^2)$ of the original objective. However, it remains to figure out how to solve this optimization problem. Bui et al. (2014) solve the lifted TRW problem using Frank-Wolfe, which has to repeatedly solve a linear program over the same feasible space (i.e., Eqs. 9 and 10). These linear programs would be huge in our setting, where C can be as large as $10,000$, leading to prohibitive running times.

4.4. Dual Decomposition

We now derive an efficient algorithm based on dual decomposition to optimize our lifted TRW objective. We will have an upper bound on the log-partition function, and thus a lower bound on the likelihood, for *any* valid edge appearance probabilities. However, our algorithm requires a specific choice for all edges: $\forall l, \rho_l = \frac{1}{K+1}$.

We assume that the corpus length N is a multiple of $K+1$, which can always be achieved by adding "filler" $\langle S \rangle$ tokens. To prove that our choice of ρ defines valid edge appearance probabilities, we demonstrate a set of $K+1$ forests T such that $\rho_{ij} = \sum_{T:ij \in T} \rho_T$, where for all T, $\rho_T = \frac{1}{K+1}$. In particular, we take forests which are made up of disconnected stars, rotated so that each edge is covered exactly once. Figure 3 illustrates this choice of forests.

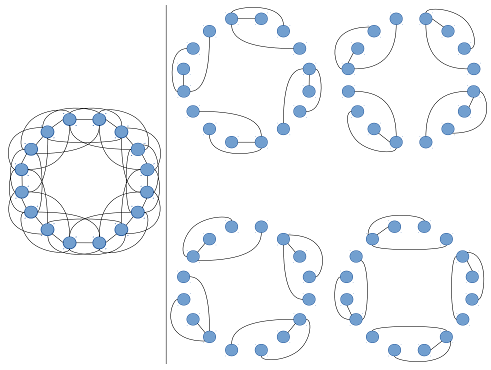
Figure 3. The set of $K+1$ covering forests used for $K = 3$, $N = 16$. Each edge is represented in exactly one forest.

Using this, we can rewrite the objective in Eq. 8 as:

$$\bar{A}(\theta) = \frac{N}{K+1} \max_{\mu} (K+1) \Big[\langle \mu_V^0, \theta^0 \rangle + H(\mu_V^0) \Big] \tag{11}$$
$$+ \sum_{l=1}^{K} \Big(\langle \mu_E^l, (K+1)\theta^l \rangle - I(\mu_E^l) \Big).$$

Finally, rather than optimizing over Eq. 11 explicitly, we re-write it in a form in which we can use a belief propagation algorithm to perform part of the maximization. To do so, we introduce redundant variables μ_V^l for $l \in [1, K]$, enforce that they are equal to μ_V^0 and use them instead of μ_V^0 for each pairwise consistency constraint. The resulting equivalent form of the optimization problem is:

$$\bar{A}(\theta) = \frac{N}{K+1} \max_{\mu} \sum_{l=0}^{K} \langle \mu_V^l, \theta^0 \rangle + \sum_{l=1}^{K} \langle \mu_E^l, (K+1)\theta^l \rangle$$
$$+ \sum_{l=0}^{K} H(\mu_V^l) - \sum_{l=1}^{K} I(\mu_E^l), \tag{12}$$

subject to non-negativity and sum-to-1 constraints, and:

$$\forall l \in [1, K], x_2, \qquad \sum_{x_1} \mu_E^l(x_1, x_2) = \mu_V^l(x_2)$$
$$\forall l \in [1, K], x_1, \qquad \sum_{x_2} \mu_E^l(x_1, x_2) = \mu_V^l(x_1)$$
$$\forall l \in [1, K], x_1, \qquad \mu_V^l(x_1) = \mu_V^0(x_1). \tag{13}$$

If one ignores the equality constraints (13), we see that the constrained optimization problem in (12) exactly corresponds to a Bethe variational problem for the tree-structured MRF shown in Figure 4. As a result, it could be

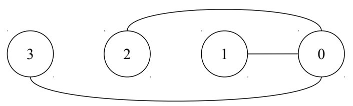

Figure 4. The tree corresponding to the maximization subproblem in the lifted inference, for $K = 3$.

maximized in linear time using belief propagation (Wainwright & Jordan, 2008, Theorem 4.2b, pg. 83–84).

Our next step is to introduce these constraints in a way that still allows for efficient optimization. This can be achieved through the use of Lagrangian duality: by formulating the right dual problem, we obtain a tight bound on our objective which can still be maximized through message passing.

We introduce Lagrange multipliers $\delta_V^l(x_l)$ for each constraint in (13) and form the Lagrangian by adding $\sum_{l=1}^K \sum_{x_l=1}^C \delta_V^l(x_l)(\mu_V^l(x_l) - \mu_V^0(x_l))$ to the objective (12). Re-arranging terms and omitting the constant, we obtain that the dual objective is:

$$O^\delta(\theta,\mu) = \langle \mu_V^0, \theta^0 - \sum_{l=1}^K \delta_V^l \rangle + \sum_{l=1}^K \langle \mu_V^l, \theta^0 + \delta_V^l \rangle \quad (14)$$
$$+ \sum_{l=1}^K \langle \mu_E^l, (K+1)\theta^l \rangle + \sum_{l=0}^K H(\mu_V^l) - \sum_{l=1}^K I(\mu_E^l).$$

Since the primal problem is concave and strictly feasible (it is feasible with no inequality constraints), Slater's conditions are met and we have strong duality. Thus,

$$A(\theta) \le \bar{A}(\theta) = \frac{N}{K+1} \min_\delta \max_{\mu \in \mathcal{LC}} O^\delta(\theta, \mu). \quad (15)$$

One useful property of the above is that we have a valid upper bound on $A(\theta)$, the log-partition function of the circular model, for *any* choice of the dual variables δ. For a fixed δ, computing the upper bound simply requires one pass of belief propagation in the tree MRF shown in Figure 4, for a running time of $O(KC^2)$.

5. Learning Algorithm

Recall that our goal is to estimate parameters θ to maximize $\bar{\mathcal{L}}(\theta, \mathbf{x}; \rho)$ given in Eq. 6. Letting $\hat{\mu}_\mathbf{x}$ denote the observed moments of the corpus $\mathbf{x}(\mathbf{t^c})$, we have for $\rho = \frac{1}{K+1}$ that:

$$\begin{aligned}\bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) &= N \times \langle \hat{\mu}_\mathbf{x}, \theta \rangle - \bar{A}(\theta) \\ &= N \times \left(\langle \hat{\mu}_\mathbf{x}, \theta \rangle - \frac{\min_\delta \max_{\mu \in \mathcal{LC}} O^\delta(\theta, \mu)}{K+1} \right) \\ &= N \times \max_\delta \left(\langle \hat{\mu}_\mathbf{x}, \theta \rangle - \frac{\max_{\mu \in \mathcal{LC}} O^\delta(\theta, \mu)}{K+1} \right) \\ &= N \times \max_\delta \mathcal{L}(\theta, \mathbf{x}; \delta),\end{aligned}$$

Algorithm 1 Tightening the bound

```
  input: model parameters $\theta$
  repeat
    compute $\bar{\theta}(\theta, \delta)$ for the lifted MRF { Eq. (17) }
    compute $\mu(\bar{\theta})$ {BP on Fig. 4 MRF}
    compute $\nabla\delta(\bar{\theta})$ { Eq. (16) }
    Take sub-gradient step: $\delta^{new} = \delta - \alpha\nabla\delta$
  until $\mu$ satisfies primal constraints { Eq. (13) }
  output: $\bar{\mathcal{L}}(\theta, \mathbf{x}; \rho)$, pseudo-marginals $\mu$
```

Algorithm 2 Gradient ascent

```
  input: data $\mathbf{x}(\mathbf{t}) = (x_i)_{i=1}^N$, precision $\epsilon$, initial $U$, $V$
  collect pairwise moments $(\hat{\mu}_{u,v})_{(u,v)\in\mathcal{X}^2}$ from the data
  repeat
    compute $\theta(U, W)$
    compute bound $\bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) = \max_\delta \mathcal{L}(\theta, \mathbf{x}; \delta)$ {Alg.1}
    compute $\nabla_\theta \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho)$ {Eq. (18)}
    compute $\nabla_U \bar{\mathcal{L}}$ and $\nabla_W \bar{\mathcal{L}}$ {Eq. (19)}
    take gradient step $U^{new} \leftarrow U + \nabla_U \bar{\mathcal{L}}$
    take gradient step $W^{new} \leftarrow W + \nabla_W \bar{\mathcal{L}}$
  until convergence: $|\bar{\mathcal{L}}^{new} - \bar{\mathcal{L}}^{old}| < \epsilon$
  output: estimated parameters $U^{new}$, $W^{new}$
```

where $\mathcal{L}(\theta, \mathbf{x}; \delta) = \langle \hat{\mu}_\mathbf{x}, \theta \rangle - \frac{\max_{\mu \in \mathcal{LC}} O^\delta(\theta,\mu)}{K+1}$. Hence, for any δ, $N \times \mathcal{L}(\theta, \mathbf{x}; \delta)$ defines a lower bound over the log-likelihood of $\mathbf{x}(\mathbf{t^c})$, which can be made tighter by optimizing over δ. Moreover, $\mathcal{L}(\theta, \mathbf{x}; \delta)$ is jointly concave in δ and θ. The learning algorithm consists of alternating between tightening this bound (Algorithm 1), and taking gradient steps in θ (Algorithm 2), in an approach similar to that of Meshi et al. (2010) and Hazan & Urtasun (2010).

Tightening the bound: For a fixed value of the parameters θ, the tightest bound is obtained for $\delta^* = \arg\min_\delta \max_{\mu \in \mathcal{LC}} O^\delta(\theta, \mu)$. We can find this minimizer through a sub-gradient descent algorithm. In particular, letting μ^* be a maximizer of $O^\delta(\theta, \mu)$, the following is a sub-gradient of $O^\delta(\theta, \mu)$ in δ:

$$\nabla \delta_V^l = \mu_V^{l*} - \mu_V^{0*} \quad \forall l \in [1, K]. \quad (16)$$

The optimal μ^* corresponds to the single node and edge marginals of the tree-structured MRF given in Figure 4, which can be computed by running belief propagation with the following log-potentials:

$$\bar{\theta}_V^0 = \theta^0 - \sum_{l=1}^K \delta_V^l, \qquad \bar{\theta}_V^l = \theta^0 + \delta_V^l \;\; \forall l \in [1, K],$$
$$\bar{\theta}_E^l = (K+1)\theta^l \;\; \forall l \in [1, K]. \quad (17)$$

Gradient Ascent: The marginals computed at δ^* can then be used to compute gradients for our main objective.

Recall that our aim is to maximize the objective function $\bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) = N \times \mathcal{L}(\theta, \mathbf{x}; \delta^*(\theta))$, where $\delta^*(\theta)$ is the output of Algorithm 1. For any value of δ, even before optimality, we have:

$$\nabla_\theta \max_{\mu \in \mathcal{LC}} O^\delta(\theta, \mu) = (K+1) \arg\max_{\mu \in \mathcal{LC}} O^\delta(\theta, \mu).$$

Hence:

$$\nabla_\theta \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) = N \times \left(\hat{\mu}_{\mathbf{x}} - \arg\max_{\mu \in \mathcal{LC}} O^{\delta^*}(\theta, \mu) \right). \quad (18)$$

For the low-rank MRFs, the gradients in the parameters can then be obtained using the chain rule. For the factorization of θ presented in Eq. 5, we get for $u, v \in \mathcal{X}$, $d \in [1, D]$, $l \in [1, K]$:

$$\nabla_{U_{u,d}} \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) = \sum_{l'=1}^{K} \sum_{v' \in \mathcal{X}} \left(\nabla_{\theta_{u,v'}} \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) \times W^{l'}_{v',d} \right)$$
$$\nabla_{W^l_{v,d}} \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) = \sum_{u' \in \mathcal{X}} \left(U_{u',d} \times \nabla_{\theta_{u',v}} \bar{\mathcal{L}}(\theta, \mathbf{x}; \rho) \right) \quad (19)$$

These can be used to perform gradient ascent on the objective function, as outlined in Algorithm 2.

6. Experiments

We conducted experiments using the lifted algorithm to examine its practical efficiency, effectiveness at estimating gradients, and the properties of the tree re-weighted bound. We implemented models for two standard natural language tasks: language modelling and part-of-speech tagging.

Setup For language modelling we ran experiments on the Penn Treebank (PTB) corpus with the standard language modelling setup: sections 0-20 for training ($N = 930$k), sections 21-22 for validation ($N = 74$k) and sections 23-24 ($N = 82$k) for test. For this dataset the vocabulary size is $C = 10$k, and rare words are replaced with `UNK`.

For part-of-speech tagging we use the tagged version of the Penn Treebank corpus (Marcus et al., 1993). We use section 2-21 for training, section 22 for validation and section 23 for test. For this corpus the tag size is $T = 36$ and we use the full vocabulary size with $C \approx 30$k.

For model parameter optimization (the gradient step in Algorithm 2) we use L-BFGS (Liu & Nocedal, 1989) with backtracking line-search. For tightening the bound (Algorithm 1), we used 200 sub-gradient iterations, each requiring a round of belief propagation. Our sub-gradient rate parameter α was set as $\alpha = 10^3/2^t$ where t is the number of preceding iterations where the dual objective did not decrease. Our implementation of the algorithm uses the Torch numerical framework (`http://torch.ch/`) and runs on the GPU for efficiency.

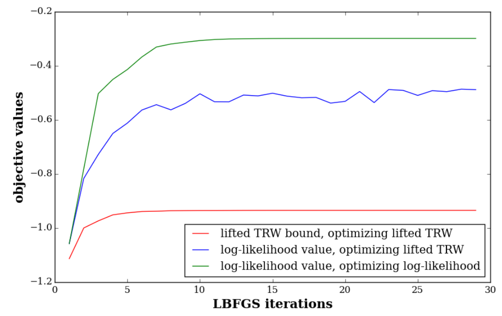

Figure 5. Comparison of a model trained by optimizing exact likelihood (green) versus the lifted TRW objective (red). The blue line shows the exact log-likelihood of the red model as it is being optimized based on the lifted TRW bound.

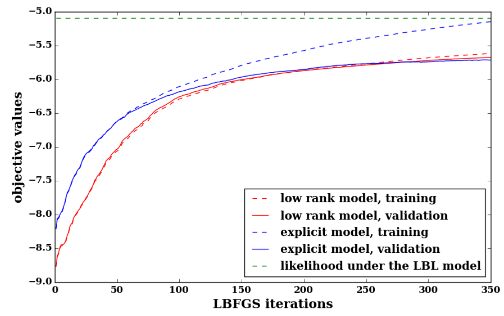

Figure 6. The red and blue lines give lower bounds on the log-likelihood (lifted objective). The green line shows the fixed value of the validation log-likelihood of an LBL model trained on PTB.

Experiments First, to confirm the properties of the algorithm, we ran experiments on a small synthetic data set with $N = 12$, $K = 1$ and $C = 4$. The small size of this data set allows us to exactly compute the log-partition for the original conditional model (Equation 3).

Figure 5 shows a comparison of a model trained using the exact gradients on the conditional likelihood to a model trained by gradient ascent with the lifted TRW objective. As expected, the latter gives an underestimate of the log-likelihood, but the learned parameters yield an exact log-likelihood close to the model learned with exact gradients.

Next we applied the lifted algorithm to a language modelling task on PTB. We trained both the explicit full-rank model and the model with low-rank log-potentials from Section 3, $\theta^{|j-i|}_{t_i,t_j} = U_{t_i} W^{|j-i|}_{t_j}$, for $D = 30$ and $K = 2$. The results are presented in Figure 6. The lower bound on the likelihood given by our algorithm is only slightly lower than the exact log-likelihood computed for a left-context LBL model with $K = 2$. We also note that the explicit model is prone to over-fitting, and gets to a worse validation objective.

Another advantage of using low-rank potentials is that they produce embedded representations of the vocabulary. Ta-

WORD	MRF LIFTED	MRF SGD	WORD2VEC
company	firm industry group	he it corp.	holding anacomp uniroyal
red	conservative freedom black	vietnamese delegation judge	cross tape delicious
has	had is was	have had n't	had been have
dollar	currency economy government	intergroup market uses	currency pound stabilized
jack	richard david carl	like needed first	kemp porter timothy

Table 1. Nearest neighbours in different embeddings. MRF LIFTED are the embeddings learned by our algorithm. MRF SGD are obtained by running stochastic gradient descent for 48 hours on the pseudo-likelihood objective: the algorithm did not converge in that time. WORD2VEC are the vectors learned by the Word2Vec software of (Mikolov et al., 2013)

ble 1 shows a sample of embeddings learned for the MRF compared to those obtained with the Word2Vec algorithm (with $D = 100$ and a window size of 4, training run for 5 epochs). We also tried training our algorithm by performing stochastic gradient descent on the pseudo-likelihood of the corpus under our model. The column MRF SGD shows the embeddings obtained after 48 hours of training. In comparison, the GPU implementation of our algorithm reached its optimal objective value on the validation dataset in 45 minutes on the Penn Treebank dataset.

Finally we ran experiments on part-of-speech tagging. For this task we use a different MRF graphical structure. Each tag node is connected to its K neighbors as well as the L nearest-words. We use a different set of covering forests which is shown in Figure 7. As with language modelling the partition function for this model would be very inefficient to compute explicitly. However, given a sentence, the best tagging can be found efficiently by dynamic programming.

For this model, we also employ explicit features for pairwise potentials, i.e. $\theta^m_{t_i,w_{i+m}} = U^m f(t_i, w_{i+m})$ and $\theta^l_{t_i,t_{i+l}} = V^l g(t_i, t_{i+l})$ where U, V are parameter matrices and f, g are predefined feature functions. For g we use tag-pair indicator features, and for f we use standard features on capitalization, punctuation, and prefixes/suffixes (given in Appendix B). This model and features are analogous to a standard conditional random field tagger; however, we optimize for joint likelihood.

It is known that joint models are less effective than discriminative conditional models for this task (Liang & Jordan, 2008), but we can compare performance to a similar joint model. We compare this model with $K = 1$ to a standard first-order HMM tagging model using the TnT tagger (Brants, 2000) with simple rare word smoothing. Table 2 shows the results. The lifted model achieves similar total accuracy, but has much better performance on unseen words, due to its feature structure.

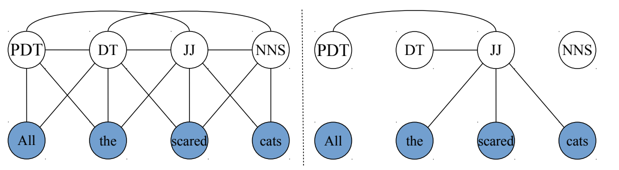

Figure 7. The POS tagging model for $K = 2, L = 3$, and a decomposition for the lifted inference algorithm

Model	Total Acc	Unk Acc
HMM	95.8	65.4
Lifted MRF	96.0	76.0

Table 2. Comparison of tagging accuracy between the lifted MRF and an HMM in total and on unseen words.

7. Conclusion

This work introduces a Markov random field language model that extends upon NLMs, and presents a fast lifted inference algorithm with complexity independent of the length of the corpus. We show experimentally that this technique is efficient and estimates useful parameters on two common NLP tasks. The use of low-rank MRFs may also be useful in other applications where random variables have very large state spaces.

Our paper presents a new application area for lifted inference, and could potentially lead to its broader adoption in machine learning. For example, one could apply our methodology to efficiently learn the parameters of grid-structured MRFs commonly used in computer vision, where symmetry is obtained using an approximation which wraps the grid around left-to-right and top-to-bottom. Our dual decomposition algorithm may also be more broadly useful for efficiently performing lifted variational inference.

Our approach opens the door to putting a much broader class of word embeddings used for language into a probabilistic framework. One of the most exciting directions enabled by our advances is to combine latent variable models together with neural language models. For example, one could imagine using our approach to perform semi-supervised or fully unsupervised learning of part-of-speech tags using vast unlabeled corpora. Our lifted variational inference approach can be easily combined with Expectation Maximization or gradient-based likelihood maximization.

Acknowledgments

YJ and DS gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Probabilistic Programming for Advancing Machine Learning (PPAML) Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-14-C-0005. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

Bengio, Yoshua, Schwenk, Holger, Senécal, Jean-Sébastien, Morin, Fréderic, and Gauvain, Jean-Luc. Neural probabilistic language models. In *Innovations in Machine Learning*, pp. 137–186. Springer, 2006.

Besag, Julian. Statistical analysis of non-lattice data. *The statistician*, pp. 179–195, 1975.

Brants, Thorsten. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pp. 224–231. Association for Computational Linguistics, 2000.

Bui, Hung, Huynh, Tuyen, and Riedel, Sebastian. Automorphism groups of graphical models and lifted variational inference. In *Proceedings of the Twenty-Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-13)*, pp. 132–141, Corvallis, Oregon, 2013. AUAI Press.

Bui, Hung Hai, Huynh, Tuyen N., and Sontag, David. Lifted tree-reweighted variational inference. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI-14)*, 2014.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

Globerson, Amir and Jaakkola, Tommi. Convergent propagation algorithms via oriented trees. In Parr, Ronald and van der Gaag, Linda C. (eds.), *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 133–140. AUAI Press, 2007.

Graff, David, Kong, Junbo, Chen, Ke, and Maeda, Kazuaki. English gigaword. *Linguistic Data Consortium, Philadelphia*, 2003.

Hazan, Tamir and Urtasun, Raquel. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *Advances in Neural Information Processing Systems*, pp. 838–846, 2010.

Jancsary, Jeremy and Matz, Gerald. Convergent decomposition solvers for tree-reweighted free energies. In *International Conference on Artificial Intelligence and Statistics*, pp. 388–398, 2011.

Liang, Percy and Jordan, Michael I. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proceedings of the 25th international conference on Machine learning*, pp. 584–591. ACM, 2008.

Liu, Dong C and Nocedal, Jorge. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Meshi, Ofer, Sontag, David, Jaakkola, Tommi, and Globerson, Amir. Learning efficiently with approximate inference via dual losses. 2010.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Mnih, Andriy and Hinton, Geoffrey. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pp. 641–648. ACM, 2007.

Mnih, Andriy and Hinton, Geoffrey E. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pp. 1081–1088, 2009.

Mnih, Andriy and Teh, Yee Whye. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(12):1–305, 2008. ISSN 1935-8237.

Wainwright, Martin J, Jaakkola, Tommi S, and Willsky, Alan S. A new class of upper bounds on the log partition function. *Information Theory, IEEE Transactions on*, 51(7):2313–2335, 2005.

Yanover, C., Schueler-Furman, O., and Weiss, Y. Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology*, 15(7):899–911, 2008.

A. Proof of Lemma 1

Let us start by reviewing some of the notions supporting the model presented in the main paper. All probability distributions defined in this paper correspond to Markov random field sequence models, so we begin by describing these models in detail.

A.1. Sequence models

Linear-chain Markov sequence model We start by considering a sequence $\mathbf{x} = (x_1, \ldots, x_n)$ of n variables with state space $\mathcal{X}$. We define an order K Markov sequence model as a Markov random field where each element of the sequence is connected to its K left and right neighbours. Figure 8 presents such a model for $n = 8$, $K = 2$.

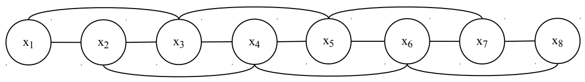

Figure 8. Second-order Markov sequence model for $n = 8$.

As mentioned in Section 3, a pairwise MRF with this structure gives the following distribution $p^n_{\text{seq}_K}$ over $\mathcal{X}^n$:

$$\forall \mathbf{x} \in \mathcal{X}^n, \quad \log(p^n_{\text{seq}_K}(\mathbf{x})) = \sum_{i=1}^{n-K} \sum_{l=1}^{K} \theta^{(i,i+l)}_{x_i, x_{i+l}} - A^n_{\text{seq}_K}(\theta) \tag{20}$$

Where:

$$A^n_{\text{seq}_K}(\theta) = \log\left(\sum_{\mathbf{y} \in \mathcal{X}^n} \exp\left(\sum_{i=1}^{n-K} \sum_{l=1}^{K} \theta^{(i,i+l)}_{y_i, y_{i+l}}\right)\right)$$

Cyclic Markov sequence model Now consider the cyclic version of the above sequence model, where the last K tokens are connected to the first K (specifically, edges are added between v_{n-k} and v_l, $\forall 1 \le k + l \le K$), as illustrated in Figure 9.

This gives the following distribution $p^n_{\text{cycl}_K}$ over $\mathcal{X}^n$:

$$\forall \mathbf{x} \in \mathcal{X}^n, \quad \log(p^n_{\text{cycl}_K}(\mathbf{x})) = \sum_{i=1}^{n} \sum_{l=1}^{K} \theta^{(i,i+l)}_{x_i, x_{i+l}} - A^n_{\text{cycl}_K}(\theta) \tag{21}$$

Where:

$$A^n_{\text{cycl}_K}(\theta) = \log\left(\sum_{\mathbf{y} \in \mathcal{X}^n} \exp\left(\sum_{i=1}^{n} \sum_{l=1}^{K} \theta^{(i,i+l)}_{y_i, y_{i+l}}\right)\right)$$

And $\forall l \in [1, K]$, $x_{n+l} = x_l$, $y_{n+l} = y_l$.

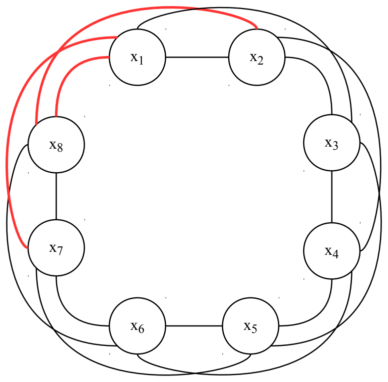

Figure 9. Cyclic second order Markov sequence model for $n = 8$.

A.2. Language modelling

To apply a Markov sequence model to language modelling we also need to explicitly handle the boundary cases of a sentence.

Consider a linear-chain Markov sequence model over a sentence of size M, let $\mathcal{T}$ denote the vocabulary of our corpus, and define the bidirectional context of a word as its K left and right neighbouring tokens. By adding K "padding" or "separator" tokens $\langle S \rangle \notin \mathcal{T}$ to the left and right boundary of the sentence, this notion of context also allows us to bias the distribution of tokens at the beginning and end of the sentence.

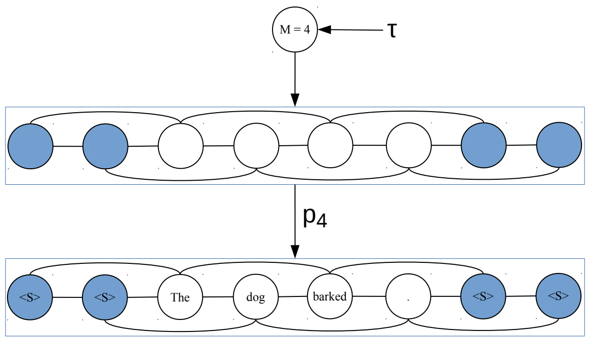

Figure 10. The full generative model for a sentence.

In terms of the sequence model defined above, a sentence $\mathbf{t} \in \mathcal{T}^M$ will then correspond to a sequence $\mathbf{x}(\mathbf{t}) \in \mathcal{X}^{M+2K}$, with $\mathcal{X} = \mathcal{T} \cup \{\langle S \rangle\}$, such that $\mathbf{x}(\mathbf{t})^{K+M}_{K+1} = \mathbf{t}$ and $\mathbf{x}(\mathbf{t})^{K}_{1} = \mathbf{x}(\mathbf{t})^{M+2K}_{M+K+1} = \langle S \rangle^K$.

This allows us to define the following distribution p^M over sentences of length M:

$$\forall \mathbf{t} \in \mathcal{T}^M,\ p^M(\mathbf{t}) = p_{\text{seq}_K}^{M+2K}\left(\mathbf{x}_{K+1}^{K+M} = \mathbf{t} \,\middle|\, \mathbf{x}_1^K = \mathbf{x}_{M+K+1}^{M+2K} = \langle S\rangle^K\right) \quad (22)$$

We can then define the following generative process for sentences (as illustrated in Figure 10):

- Draw the sentence length M from a distribution over integers τ.
- Draw a sequence of M tokens: $\mathbf{t}^M = (t_1, \ldots, t_M) \sim p^M(\mathbf{t}^M)$.

Under this model, the likelihood of a corpus $\mathbf{t}^c = (\mathbf{t}^1, \ldots, \mathbf{t}^{n_c})$ is then:

$$\begin{aligned} p(\mathbf{t}^c) &= \prod_{i=1}^{n_c} \tau(M_i) p^{M_i}(\mathbf{t}^i) \\ &= \tau(M_1, \ldots, M_{n_c}) \prod_{i=1}^{n_c} p^{M_i}(\mathbf{t}^i) \end{aligned}$$

Since the maximum likelihood parameters of τ can easily be estimated, we focus on the second part $\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t}^i)$ in the rest of the proof.

A.3. Proving the Lemma

Now we consider the lemma of interest relating the linear-chain Markov sequence model to the cyclic model. We restate the lemma here:

Lemma. *Let* $\mathcal{S} = \{\mathbf{x}(\mathbf{t^c}) | \mathbf{t^c} \in \mathcal{T}^{M_1} \times \ldots \times \mathcal{T}^{M_{n_c}}\}$. *Then,*

$$\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t}_i) = \frac{p_{cycl_K}^N(\mathbf{x} = \mathbf{x}(\mathbf{t}))}{p_{cycl_K}^N(\mathbf{x} \in \mathcal{S})}.$$

Our proof first shows how to chain together sentences in a corpus, and then applies the cyclic Markov sequence model.

Concatenating sentences Consider a corpus of c sentences $\mathbf{x}(\mathbf{t^c}) = (\mathbf{t}^1, \ldots, \mathbf{t}^{n_c})$ (of lengths $(M_1, \ldots, M_c)$) independently drawn from the above model. As above, we can use a mapping $\mathbf{x}$ of $\mathbf{t}$ to $\mathcal{X}^{N+K}$, where: $N = K + M_1 + \ldots + K + M_{n_c}$, by adding $\langle S\rangle$ tokens at the beginning and end of the corpus and between adjacent sentences:

$$\mathbf{x}(\mathbf{t}^1, \ldots, \mathbf{t}^{n_c}) = \quad (23)$$
$$\Big(\underbrace{\langle S\rangle, \ldots, \langle S\rangle}_{\times K}, t_1^1, \ldots, t_{M_1}^1, \underbrace{\langle S\rangle, \ldots, \langle S\rangle}_{\times K}, t_1^2, \ldots, t_{M_c}^c, \underbrace{\langle S\rangle, \ldots, \langle S\rangle}_{\times K}\Big)$$

Let us first consider the base case where $c = 2$. From Equations 20 and 22, we get that:

$$\begin{aligned} \forall j \in \{1, 2\} \quad p^{M_j}(\mathbf{t}^j) &\propto p_{\text{seq}_K}^{M_j+2K}(\mathbf{x}(\mathbf{t}^j)) \\ &\propto \exp\Big(\sum_{i=1}^{M_j+K} \sum_{l=1}^{K} \theta^l_{\mathbf{x}(\mathbf{t}^j)_i, \mathbf{x}(\mathbf{t}^j)_{i+l}}\Big) \end{aligned}$$

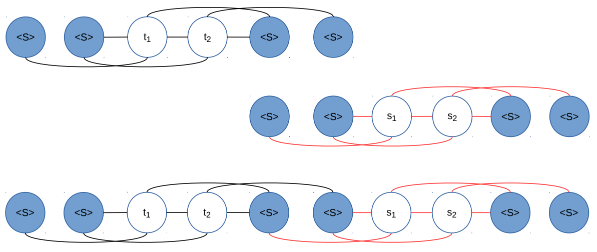

Figure 11. Concatenating sentences $\mathbf{t}^1$ and $\mathbf{t}^2$

Additionally, we have by construction:

$$\begin{aligned} \forall l \in [1, K], \quad \mathbf{x}(\mathbf{t^1})_{M_1+K+l} &= \mathbf{x}(\mathbf{t^2})_l \\ &= \mathbf{x}(\mathbf{t^1}, \mathbf{t^2})_{M_1+K+l} \\ &= \langle S\rangle \end{aligned}$$

Hence:

$$\begin{aligned} \sum_{i=1}^{M_1+M_2+2K} \sum_{l=1^K} \theta^l_{\mathbf{x}(\mathbf{t^1},\mathbf{t^2})_i, \mathbf{x}(\mathbf{t^1},\mathbf{t^2})_{i+l}} = \\ \sum_{i=1}^{M_1+K} \sum_{l=1^K} \theta^l_{\mathbf{x}(\mathbf{t^1})_i, \mathbf{x}(\mathbf{t^1})_{i+l}} \\ + \sum_{j=1}^{M_2+K} \sum_{l=1^K} \theta^l_{\mathbf{x}(\mathbf{t^2})_j, \mathbf{x}(\mathbf{t^2})_{j+l}} \end{aligned}$$

In other words:

$$\begin{aligned}
p^{M_1}(\mathbf{t}^1)p^{M_2}(\mathbf{t}^2) &\propto \exp(\sum_{i=1}^{M_1+K}\sum_{l=1}^{K}\theta^l_{\mathbf{x}(\mathbf{t^1})_i,\mathbf{x}(\mathbf{t^1})_{i+l}}) \\
&\times \exp(\sum_{i=1}^{M_2+K}\sum_{l=1}^{K}\theta^l_{\mathbf{x}(\mathbf{t^2})_i,\mathbf{x}(\mathbf{t^2})_{i+l}}) \\
&\propto \exp(\sum_{i=1}^{M_1+M_2+2K}\sum_{l=1^K}\theta^l_{\mathbf{x}(\mathbf{t^1},\mathbf{t^2})_i,\mathbf{x}(\mathbf{t^1},\mathbf{t^2})_{i+l}}) \\
&\propto p^{M_1+M_2+3K}_{\text{seq}_K}(\mathbf{x}=\mathbf{x}(\mathbf{t^1},\mathbf{t^2}))
\end{aligned}$$

By induction, we get that:

$$\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t^i}) \propto p^{N+K}_{\text{seq}_K}(\mathbf{x}=\mathbf{x}(\mathbf{t}))$$

Now, let $\mathcal{S}_N = \{\mathbf{x}(\mathbf{t})|\mathbf{t}\in\mathcal{T}^{M_1}\times\ldots\times\mathcal{T}^{M_c}\}$. Since the text model is defined for $\mathbf{x}\in\mathcal{S}_N$, by normalization, it then follows that:

$$\begin{aligned}
\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t^i}) &= \frac{p^{N+K}_{\text{seq}_K}(\mathbf{x}=\mathbf{x}(\mathbf{t}))}{p^{N+K}_{\text{seq}_K}(\mathbf{x}\in\mathcal{S}_N)} \\
&= p^{N+K}_{\text{seq}_K}(\mathbf{x}=\mathbf{x}(\mathbf{t})|\mathbf{x}\in\mathcal{S}_N) \qquad (24)
\end{aligned}$$

Using a cyclic model Finally, $\forall\mathbf{x}\in\mathcal{S}_N$, we have that $\forall l\in[1,K], x_l = x_{N+l} = \langle S\rangle$. According to Equations 20 and 21, this means that:

$$\forall\mathbf{x}\in\mathcal{S}_N,\ \ p^{N+K}_{\text{seq}_K}(\mathbf{x}) \propto p^{N}_{\text{cycl}_K}(\mathbf{x})$$

Hence:

$$\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t^i}) \propto p^{N}_{\text{cycl}_K}(\mathbf{x})$$

Which by normalization gives us:

$$\begin{aligned}
\prod_{i=1}^{n_c} p^{M_i}(\mathbf{t^i}) &= \frac{p^{N}_{\text{cycl}_K}(\mathbf{x}=\mathbf{x}(\mathbf{t}))}{p^{N}_{\text{cycl}_K}(\mathbf{x}\in\mathcal{S}_N)} \\
&= p^{N}_{\text{cycl}_K}(\mathbf{x}=\mathbf{x}(\mathbf{t})|\mathbf{x}\in\mathcal{S}_N) \qquad (25)
\end{aligned}$$

Which proves the lemma.

B. Implementation Details

Synthetic data generation The synthetic data used to obtain the results presented in Figure 5 consists of a sequence of 12 tokens sampled uniformly at random from $\mathcal{T}$ = {a,b,c,d}. For $K = 2$ this gives a sequence of the form:

$\langle S\rangle$ $\langle S\rangle$ a b c d b a b d c b a c $\langle S\rangle$ $\langle S\rangle$

Language modelling experiments In our implementation of the inner loop of the algorithm (Algorithm 1 of the main paper), we use LBFGS to find the optimal value of δ. However, as mentioned in Section 4, the inner loop does not need to be run to optimality to find an ascent direction.

The LBL model in Figure 6 was trained using SGD on minibatches of size 64. The learning rate was initialized at 0.1, and halved any time the error validation went up.

The second model presented in Table 1 (MRF SGD) was trained by running SGD on the model pseudo-likelihood, with minibatches of size 100. The learning rate was initialized at 0.025 and decayed as $\frac{1}{t^{0.4}}$.

Sequence tagger features For part-of-speech tagging experiments we make use of two feature functions $g(t_i, t_{i+l})$ and $f(t_i, w_{i+m})$. The tag-tag function g simply consists of indicator features for all possible pairs of tags. The feature function $f(t_i, w_{i+m})$ conjoins an indicator of the tag t_i with surface-form features including:

- An indicator for the word w_{i+m} itself.
- Prefixes and suffixes of w_{i+m} up to length 4.

When $m = 0$, i.e. the potential with the tag directly above a word, the tag is further conjoined with a standard set of morphology features including:

- Is w_i completely upper case?
- Is the first letter of w_i upper case?
- Does w_i end with 's'?
- Is the first letter of w_i upper case and it ends with 's'?
- Is w_i completely upper-case and it end with 'S'?
- Does w_i contain a digit?
- Is w_i all digits?
- Does w_i contain a hyphen?