INVITED PAPER

Conditional Random Fields in Speech, Audio, and Language Processing

In this paper, we provide a tutorial overview of conditional random fields—a discriminative sequence model—and their applications in audio, speech, and language processing.

By Eric Fosler-Lussier, *Senior Member IEEE*, Yanzhang He, Preethi Jyothi, and Rohit Prabhavalkar, *Graduate Student Member IEEE*

ABSTRACT | Conditional random fields (CRFs) are probabilistic sequence models that have been applied in the last decade to a number of applications in audio, speech, and language processing. In this paper, we provide a tutorial overview of CRF technologies, pointing to other resources for more in-depth discussion; in particular, we describe the common linear-chain model as well as a number of common extensions within the CRF family of models. An overview of the mathematical techniques used in training and evaluating these models is also provided, as well as a discussion of the relationships with other probabilistic models. Finally, we survey recent work in speech, audio, and language processing to show how the same CRF technology can be deployed in different scenarios.

KEYWORDS | Automatic speech recognition (ASR); natural language processing (NLP); random fields; statistical learning

I. INTRODUCTION

In many speech, audio, and language processing problems, a system must extract a higher level description of a sequential input signal; the exact type of input and extracted information depends on the problem definition. As one example, consider determining whether word sequences in running text correspond to a person or a place (or neither): in isolation, we can identify that *Johnston*, by virtue of its capitalization in English, is likely a proper noun, but it could be a person, a city, or a street. Similarly, the abbreviation *Dr.* is ambiguous between *Doctor* and *Drive*. In sequence, however, we can combine this evidence and predict that *Dr. Johnston* is likely a person, while *Johnston Dr.* is likely a street name.

Probabilistic sequence modeling has long been used to tackle a variety of these speech, audio, and language tasks: for example, the well-known hidden Markov model (HMM) has been the backbone of automatic speech recognition (ASR) research for decades. In this tutorial, we explore conditional random fields (CRFs)—a model that allows for combination of local information to predict a global probability model over sequences. Originally proposed in 2001 [1], this model has been applied to a significant number of tasks across these domains.

CRFs are really a class of models: one purpose of this paper is to document the specific properties of CRFs across different implementations. In the original Lafferty *et al.* paper [1], a CRF is defined as a discriminative model that predicts the global probability of a sequence of random variables, assuming a first-order Markov dependency between the variables (also known as a linear-chain model); the probability structure depends on a log-linear combination of evidence "features" derived from the input. This is typically the first style of model that comes to mind when CRFs are mentioned. However, there are many modeling choices that are encapsulated in the dense description above. Several researchers have sought to extend CRFs in one or more directions because of the particular problem they are working with: examples include developing richer graph structures that model collections of

Manuscript received April 6, 2012; revised September 3, 2012; accepted December 27, 2012. Date of publication April 12, 2013; date of current version April 17, 2013. This work was supported by the National Science Foundation (NSF) CAREER Grant IIS-0643901 and the NSF Grant IIS-0905420. The opinions and conclusions expressed in this work are those of the authors and not of any funding agency.
The authors are with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: fosler@cse.ohio-state.edu; hey@cse.ohio-state.edu; jyothi@cse.ohio-state.edu; prabhava@cse.ohio-state.edu).
Digital Object Identifier: 10.1109/JPROC.2013.2248112

variables, introducing hidden structure into the model, changing the optimization criteria, or adjusting the probability structure to include binary switching variables rather than pure log-linear probabilities. In this paper, we take the view that all of these variants fall under the general class of CRFs, and we provide a roadmap that can help readers understand the properties of CRFs, both in the canonical linear-chain case, as well as some of the extensions that have been studied in the literature.

CRFs were originally developed in the machine learning community and evaluated using natural language processing (NLP) tasks; as we discuss in Section VI, the model has since found utility in the speech and audio processing areas. This paper also explores the relationships of CRFs to other types of models utilized in these other fields [particularly HMMs and multilayer perceptrons (MLPs)]: knowing these relationships can allow researchers to cross-fertilize techniques, such as deep-learning CRFs [2], which hybridize deep learning techniques for MLPs with the sequence modeling of CRFs. We also point to various resources for further reading, including additional tutorial material [3] and work describing the relationships between HMMs and CRFs [4], [5].

In the next section, we describe the historical development of CRFs; we also discuss the particular properties that are attributed to linear-chain CRFs and how those properties have been extended in subsequent research. Section III gives a more in-depth view of the mathematics behind linear-chain CRFs, including training and decoding algorithms, defining feature functions for describing observations, and methods of avoiding overfitting. We then extend the discussion to graphical structures that are not linear chain in Section IV, and examine the relationships to other classifier technologies in Section V. With this background in mind, we then turn to specific applications in audio, speech, and language processing in Section VI, followed by concluding remarks.

II. HISTORICAL PERSPECTIVE AND ANALYSIS

The initial impetus for the development of CRFs was the desire to construct direct, discriminative models of hidden sequences that depend on the observations. In this section, we first describe the progression from HMMs to maximum entropy Markov models (MEMMs) and then to CRFs, as laid out by Lafferty *et al.* [1]. In order to illustrate the model, we draw on an example from the classic HMM paper of Rabiner [6], and show how the parameterizations of the three models differ.

After providing the historical overview of CRF development, we break down the model and describe its component properties, which allows us to understand more fully the assumptions in the model described in [1], as well as see how variants of the model might fit into the CRF family.

A. Why Were CRFs Developed?

We begin our discussion of CRF development by recalling Rabiner's tutorial on HMMs [6]. Rabiner motivates the power of the HMM statistical model using a ball-and-urn analogy: consider a set of n urns that holds balls of several colors; the distributional mix of ball colors differs from urn to urn. At every time step, a ball is selected from an urn and its color announced (and then replaced), and then with some probability the selector moves to a different urn; this process continues to generate a sequence of observations (Fig. 1). Crucially, which urn the ball is selected from is hidden from the observer; only the sequence of ball colors is reported to the observer.

Throughout this paper, we will be using a sequence $S = \{s_1, s_2, \ldots, s_T\}$ to represent hidden states of various problems (usually the quantity that one would like to predict) and $O = \{o_1, o_2, \ldots, o_T\}$ to represent a sequence of observation variables. In Section VI, we introduce each problem by defining the values that S and O can take on. For example, Rabiner's ball-and-urn problem can be specified as:

Ball-and-Urn Problem
S: Sequence of urns
O: Color of balls

In order to understand the assumptions in the ball-and-urn problem better, let us lay out the statistical assumptions made in the HMM. Briefly stated, HMMs compute a joint probability between a set of hidden labels (S) and observations (O)

$$P(S, O) = \prod_{t=1}^{T} P(s_t|s_{t-1})P(o_t|s_t) \tag{1}$$

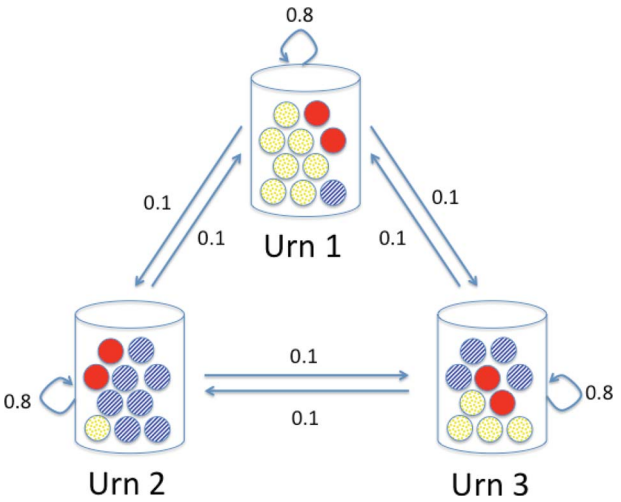


Fig. 1. *The ball-and-urn problem: three urns with associated transition probabilities and observation distributions over yellow (speckled), red (solid), or blue (striped) balls. A ball is drawn from an urn and the selector transitions probabilistically to a successor urn. Each urn is equally likely to be selected as the first urn to draw from.*

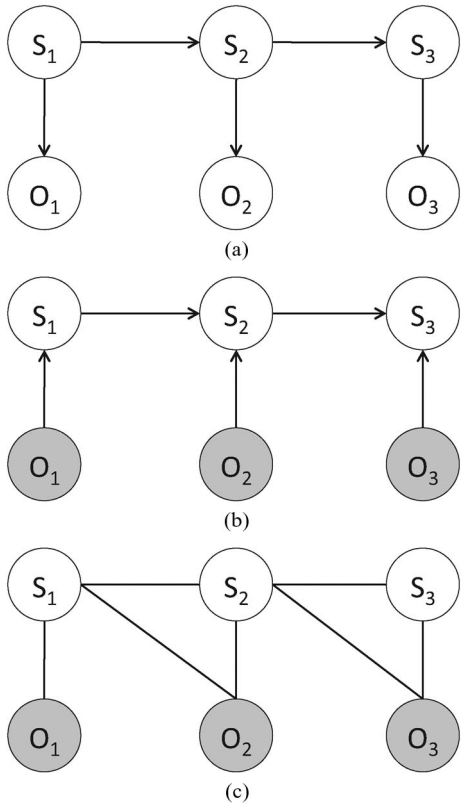


Fig. 2. ***Graphical models for three formalisms describing factorizations of relationships between S and O. HMMs treat observations as random variables with associated probability distributions, and thus are unshaded. MEMMs and CRFs treat observations (shaded) as fixed and thus do not model the distribution. The CRF factorization given here mimics the MEMM factorization in that there is an observational dependence on the transitions. (a) HMM graphical model describing $P(S,O)$. (b) MEMM graphical model describing $P(S|O)$. (c) Linear-chain CRF graphical model describing $P(S|O)$.***

where the joint probability is composed of local probabilities over the state sequence transitions $(P(s_t|s_{t-1}))$ and observation probabilities $(P(o_t|s_t))$.[1] The corresponding graphical model for this factorization can be seen in Fig. 2(a).

The HMM can be considered a generative model in that the distributions for observations are determined by the states; the states are seen as generating the observations.[2]

We assume that the distributions remain stationary over time (which is true for the ball-and-urn problem, since we replace the ball after selecting it). To make this more concrete, let us pick some distributions over yellow, blue, and red balls for $n =$ three urns, as depicted in Fig. 1. Equation (1) can be used to compute the probability distribution over joint (S,O) sequences; for example, $P(111222333, YYRBBRBYR) = 2.7e^{-7}$. For recognition tasks where we want to find a sequence S given a fixed O, finding the S that maximizes $P(S,O)$ also maximizes the conditional probability $P(S|O) = P(S,O)/P(O)$.

[1]For time $t = 1$, we assume an unconditional prior distribution over the starting state $P(s_1|s_0) = P(s_1)$; equivalently, we can add a start symbol and assume $P(s_0 = \text{start}) = 1$. We carry this notational convenience throughout the rest of the paper.

[2]It is important to draw the distinction here between models and training criteria: generative models can be trained using discriminative criteria that improve end task performance.

Table 1 MEMM Ball-and-Urn Probability Distributions for Noninitial States Assuming IID Distribution of Observations

s_{t-1}	o_t	$P(s_t = \{1,2,3\}\|s_{t-1}, o_t)$
1	Y	{0.92, 0.02, 0.07}
1	B	{0.42, 0.37, 0.21}
1	R	{0.80, 0.10, 0.10}
2	Y	{0.37, 0.42, 0.21}
2	B	{0.02, 0.92, 0.07}
2	R	{0.10, 0.80, 0.10}
3	Y	{0.18, 0.03, 0.80}
3	B	{0.03, 0.18, 0.80}
3	R	{0.10, 0.10, 0.80}

Distributions do not sum to one because of rounding.

The HMM decomposition of the joint probability requires finding a good generation model of the likelihood of observations given each individual label $P(o_t|s_t)$. McCallum *et al.* observed that this seems antithetical to the idea of predicting a label sequence; in essence, we want to directly compute the conditional label posterior $P(S|O)$ by predicting the current label given the previous label and the current observation [7]:

$$P(S|O) = \prod_{t=1}^{T} P(s_t|s_{t-1}, o_t). \tag{2}$$

The corresponding graphical model, referred to as an MEMM, can be seen in Fig. 2(b); note the inversion of the arrow directions, which indicate that the hidden state distributions now are conditioned on the observations. Observations are now taken as fixed (and, hence, are shaded in the figure). Table 1 demonstrates some interesting properties of the observation-dependent transition probability distribution corresponding to the ball-and-urn problem. Observing yellow when the previous selection came from urn 1 is a good indicator that the current selection also comes from urn 1 (and similarly for blue and urn 2). Red is a poor discriminator of which urn a ball came from; when red is observed the distribution reverts to the prior $P(s_t|s_{t-1})$.

Why might it be advantageous to model transitions dependent on observations? The HMM model assumes that observations are drawn from a state independently and identically distributed (IID), which might not hold for all situations.[3] Consider a modification to the ball-and-urn

[3]Speech observations are not IID, for example: two adjacent frames in a steady-state vowel will be relatively close in observation space.

rule: the selector continues to draw from the same urn until she draws a red ball, at which point one of the other two urns is selected at the next turn (with equal probability). Under these rules, $P(11, RY) = 0$, but the HMM will give a nonzero probability.[4]

Since it can be difficult to predict a current label conditioned on both the previous label and the current observation, McCallum *et al.* use a maximum entropy distribution to model the component probabilities, in which an exponential model is used to relate the observations and previous label to the current label via a set of weighted descriptive feature functions [7][5]:

$$P(s_t|s_{t-1}, o_t) = \frac{\exp\left(\sum_i^{\#\text{func}} \lambda_i f_i(o_t, s_{t-1}, s_t)\right)}{Z(o_t, s_{t-1})} \quad (3)$$

where #func is the number of feature functions in the model, and $Z(o_t, s_{t-1}) = \sum_{s'_t} \exp(\sum_i^{\#\text{func}} \lambda_i f_i(o_t, s_{t-1}, s'_t))$ is the partition function, or normalization constant, that ensures that the probability distribution sums to one. The feature functions of the MEMM can be as simple as binary indicators of particular transitions between labels (ignoring the observations), or complicated functions of the observation; the feature functions must be defined by the system designer and should reflect knowledge of the problem at hand. The learned weights λ_i indicate the importance of each function in relating a label to the previous label and current observation.[6]

For the ball-and-urn problem, a single set of binary indicator feature functions can be used to relate (3) to the probabilities in Table 1. For example, to model $P(s_t = 1|s_{t-1} = 1, o_t = Y)$, we can define a feature function

$$f_1(o_t, s_t, s_{t-1}) = \begin{cases} 1, & \text{if } o_t = Y,\ s_t = 1,\ s_{t-1} = 1 \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Similarly, we can have indicator functions f_2 select for $(o_t = Y, s_t = 2, s_{t-1} = 1)$, f_3 select for $(o_t = Y, s_t = 3, s_{t-1} = 1)$, etc. By setting $\lambda_1 = \log(0.92)$, $\lambda_2 = \log(0.02)$, $\lambda_3 = \log(0.07)$, and so forth, we can encode the probabilities in Table 1 using this weighted feature representation.[7]

However, one potential downfall of MEMMs is what is known as the label bias problem [9]. Factoring $P(S|O)$ into terms of $P(s_t|s_{t-1}, o_t)$ means that if there are very few states s_t that can follow s_{t-1}, then the role of o_t in distinguishing them is very diminished. In the extreme case when only one value of s_t (say a) can follow $s_{t-1} = b$, then $P(s_t = a|s_{t-1} = b, o_t) = 1\ \forall o_t$, and thus $P(s_t \neq a|s_{t-1} = b, o_t) = 0$; the impact of o_t is completely ignored by the model.[8] In essence, the problem arises because every state is required to be locally normalized to sum to a probability distribution [cf., the denominator in (3)].

CRFs were first introduced by Lafferty *et al.* [1] to address the label-bias problem found in MEMMs. CRFs are still conditional models [i.e., they directly compute $P(S|O)$], but handle the label bias problem by jointly modeling the S sequence and globally normalizing over the entire sequence structure. The CRF factorization for an HMM-like structure (often termed a linear-chain CRF)[9] can be given by

$$P(S|O) = \frac{1}{Z(O)} \prod_{t=1}^{T} \phi_{\text{state}}(s_t, o_t) \phi_{\text{trans}}(s_t, s_{t-1}, o_t) \quad (5)$$

where $Z(O)$ is a normalization term which ensures that (5) forms a valid probability distribution, and $\phi(\cdot)$ are nonnegative potential functions over state/observation configurations. State potentials associate observations with a particular state; transition potentials associate pairs of states with observations. Similar to MEMMs, one can model these as an exponential family function, thus ensuring that each potential is positive

$$\phi_{\text{state}}(s_t, o_t) = \exp\left(\sum_i^{\#\text{func}} \lambda_i f_i(s_t, o_t)\right) \quad (6)$$

$$\phi_{\text{trans}}(s_t, s_{t-1}, o_t) = \exp\left(\sum_i^{\#\text{func}} \lambda_i f_i(s_t, s_{t-1}, o_t)\right). \quad (7)$$

Fig. 2(c) describes the graphical model corresponding to this linear-chain CRF. Note that the model is undirected (indicating relationships between nodes defined by potential functions rather than conditional probability distributions); the cliques in the graph structure define the domain of the potential functions associated with the global

[4]The HMM can model this situation by subdividing the state space into two states for each urn—the "stay with this urn" state and the "leave this urn" state—and keeping separate distributions for each.

[5]The notation in this equation differs somewhat from that in [7] in that the previous state is explicitly represented, following notation in subsequent work [1].

[6]It should be noted that multinomial logistic regression has exactly the same mathematical form as maximum entropy classification; in both systems weights have exactly the same role and are estimated similarly [8].

[7]Note the Z-term will always sum to one in this case, since we already have a normalized probability distribution.

[8]Whether this is a problem in any particular domain is really an empirical question.

[9]Nonlinear-chain CRFs are discussed in Section IV; we also restrict observations here to a single frame o_t for convenience of comparison to HMMs, but, in general, potential functions can be defined over arbitrary input O; see (5).

probability distribution.[10] As a generalization, one can just define all functions over the maximal clique $\phi(s_t, s_{t-1}, o_t)$, merging (6) into (7), and allow the associated functions to ignore any subset of the variables, providing for definitions of state or transition bias functions that ignore the observations.

The conditional probabilities for the MEMM can be used to define one parameter setting for the ball-and-urn problem: for any pair of states and observation found in Table 1, we can define a potential function $\phi(s_t, s_{t-1}, o_t) = \lambda_i f_i(s_t, s_{t-1}, o_t)$ where $\lambda_i = \log P(s_t|s_{t-1}, o_t)$ and $f_i(s_t, s_{t-1}, o_t) = 1$ iff o_t is observed and we are hypothesizing a transition from s_{t-1} to s_t.[11] This represents only one possible parameter setting corresponding to the same probability distribution; because potential functions in CRFs are only constrained to be nonnegative (unlike MEMMs, which have a requirement for all transitions from a state to sum to 1 as a probability distribution), there are multiple potential function values that can lead to the same global probability distribution [5].[12]

In short, there are two basic changes going from MEMMs to CRFs. First, the observations O have an impact both on the hidden states individually [see (6)] and on the transitions between the states [see (7)]. Second, note the change in the normalization constant $Z(O)$: instead of local normalization for each state (as in the MEMM), we compute the accumulated potentials over all possible S sequences. For linear-chain CRFs, there is an efficient forward–backward computation to compute $Z(O)$ outlined by Lafferty *et al.* They also demonstrate improvement over MEMMs using a part of speech tagging task [1].

B. What Are the Properties of CRFs?

We presented above the historical progression that introduced CRFs into the NLP community, and described various concepts that were an integral part of the development of that line of research. We can tease apart four interrelated parts within a CRF system; different users of CRFs often stress one part or another as a desirable property over some baseline model (e.g., HMMs), and in fact some claims are really combinations of multiple properties. Take the claim that Lafferty *et al.* make about conditional models such as CRFs [1]:

> "A conditional model specifies the probabilities of possible label sequences given an observation sequence. Therefore, it does not expend modeling effort on the observations, which at test time are fixed anyway."

In the above quote, there is a conflation between two different (but interrelated) aspects of the statistical system: model structure and training criteria (which we define below); Lafferty suggests that the generative model structure of HMMs (where observations are considered to be "generated" by hidden states) requires them to expend "modeling effort" on capturing all of the observed data. However, this is only true when HMMs are trained with a generative criterion (e.g., maximum likelihood), which maximizes the likelihood of the observations given the states. Discriminative training criteria allow the association between observations and hidden states to be focused on separating hidden states, rather than on modeling the data [10]. Heigold *et al.* argue quite elegantly that HMMs and CRFs with the same model structure will give the same theoretical results given equivalent training criteria [5]. This is elaborated on further in Section V.

We can decompose a CRF into four components: model structure, parameterization, inference methods, and parameter estimation.

Model structure: The structure of a model is given by the random variables in the system and the connections (factors) that the modeler wants to draw between the variables. For an HMM, there is one factor for the relationships between observations O and hidden states $S(f(o_t, s_t) = P(o_t|s_t))$, and another factor for the relationships between hidden states $(f(s_t, s_{t-1}) = P(s_t|s_{t-1}))$. Heigold *et al.* point out that when the CRF does not have observational dependence on transitions, the factorization of a linear-chain CRF, and an HMM is identical (state observations are factors $f(o_t, s_t)$, and transition functions are factors of the form $f(s_{t-1}, s_t)$), and they provide proofs that HMMs and CRFs of this form can be transformed into one another [5].

In Rabiner's ball-and-urn example, the discrete HMM probability distribution can be represented in a log-linear CRF by having a set of observation functions $f(s_t = n, o_t = c) = 1$ iff the state value is n and the observation is c at time t (and 0 otherwise), with the weight $\lambda_{s=n,o=c} = \log P(o = c|s = n)$. A similar weighted bias feature can be set up for the transition probabilities as well.

However, for linear-chain CRFs that have transition dependence on observations (similar to MEMMs), factorization equivalence with HMMs no longer holds. As pointed out above, such CRFs have feature functions that relate states to observations $(f(s_t, o_t))$, as well as separate functions that relate transitions to observations (e.g., $f(s_t, s_{t-1}, o_t)$); this factorization allows for a different decomposition of the joint probability of the hidden variables. In short, the HMM assumes that s_{t-1} is independent of o_t given s_t: there is no way for the standard HMM to represent a covariance of the three variables.[13]

[10] A clique is a set of nodes forming a fully connected subgraph in a graph; in Fig. 2(c), S_1 and O_1 form one clique, while S_1, S_2, and O_2 form another clique.

[11] For example, the weight corresponding to $\phi(1, 2, Y)$ would be $\lambda_i = \log(0.37)$.

[12] Consider the case where all potential functions ϕ are multiplied by a constant c; this just scales the unnormalized potential score of any sequence of length T by c^T, thus leaving the posterior $P(S|O)$ unchanged.

[13] Again, the state space of the HMM can be augmented to incorporate this dependence.

In general, whether using directed graphical models [e.g., HMMs and other dynamic Bayesian networks (DBNs)] or undirected graphical models (CRFs), the key elements of the model structure are the factorizations represented by the graph. For undirected models, factorization is given in terms of the cliques of the graph. For directed models, the factorization is given by the dependence between a particular variable and its parents. It is possible to use richer graph structures beyond those seen in Fig. 2(c); for example, hidden conditional random fields (HCRFs) introduce intermediate hidden variables into the graph that marginalize over different substate configurations. These richer models are explored further in Section IV.

Another point of departure for CRFs is that Lafferty *et al.*'s definition generally allows for any dependence on the observations $(f(s_t, O), f(s_{t-1}, s_t, O))$, which would require explicit dependencies and generative probability distributions in an HMM-like DBN representation. What makes this feasible is that the O are generally fixed observations, rather than treated as random variables; this distinction has implications for the probability distributions that are modeled.

Parameterization: While the factorizations of the probability distributions of the HMM and simple CRFs can be equivalent, the parametric forms of the models typically are not the same. In an HMM, the state transition factor $P(s_t|s_{t-1})$ is typically represented by a stochastic transition matrix, and the observation factor $P(o_t|s_t)$ is often modeled with a conditional probability table for discrete observations, or a distribution such as a mixture of Gaussians for continuous observations. For a CRF, factors are typically combined using a log-linear model [cf., (5)–(7)]. In general, the joint probability of hidden variables in any Markov random field is given by defining positive potential functions over the configuration cliques in the graph. By defining the potential function as the exponential of a sum of weighted features (log-linear model), we can ensure that every potential is positive.

It is important to note that we are indicating the typical forms of probability distributions for HMMs and CRFs. One could specify a different form for $P(o_t|s_t)$ in an HMM, for example, using discrete vector quantization probabilities. Similarly, one could use models outside the exponential family to derive clique potentials in a CRF. Yet most CRF implementations take the form of the log-linear model because it is relatively easy to estimate parameters according to the conditional maximum-likelihood criterion.

Parameter estimation: The typical method for estimating parameters is to iteratively update parameters with one of a number of methods (e.g., iterative scaling or gradient ascent) to optimize the conditional maximum likelihood (CML) of the correct S sequence conditioned on O, directly optimizing $P(S|O)$. Again, the Lafferty *et al.* criticisms of HMMs are based on the traditional method of maximum-likelihood training, which optimizes $P(S, O)$ via the expectation–maximization (EM) algorithm [63]. However, other discriminative techniques for training HMMs have become popular in fields such as ASR; as we discuss later, the CML criterion for CRF training is equivalent to the maximum mutual information estimation (MMIE) criterion for training discriminative HMMs.[14]

As a practical matter, CML training of linear-chain CRFs, which utilize a forward–backward recursion similar to that of ML training for HMMs, can train discriminative models relatively efficiently in terms of space; many implementations of MMIE (and other criteria) for HMMs require the generation of competitor lattices, which can be cumbersome to store. We discuss the CML training criterion further in Section III.

Inference mechanism: The forward–backward recurrence mentioned above is a nice property of linear-chain CRFs: like an HMM, linear-chain CRFs have exact inference mechanisms, both in terms of the alpha–beta recurrences for parameter updating as well as a Viterbi algorithm for determining best paths (also discussed in Section III). However, like nonpolytree DBNs, CRFs with richer structures may require inference that is exponential in the number of hidden variables. Fortunately, approximate inference mechanisms, such as Monte Carlo sampling or beamwidth pruning techniques, can be successfully applied to richer CRF structures, such as factorial CRFs [11]. Such algorithms are described further in Section IV.

By considering these four components as separate, but interrelated, contributions to CRF technology, one can consider how variants of the original linear-chain CRF model may fit into the CRF family. In Section III, we take a closer look at probability distributions, parameter estimation, and inference in linear-chain CRFs, followed by a discussion of models that move beyond simple linear-chain structures.

III. THE MATHEMATICS BEHIND LINEAR-CHAIN CRFs

The mathematical formalism for training and decoding CRFs has been extensively discussed elsewhere [1], [4], [11]–[13]; in this section, we provide a summary of the mathematics for parameter estimation and inference for linear-chain CRFs for those who are familiar with traditional HMM techniques. We begin by describing the process for estimating the parameters of a linear-chain CRF in Section III-A1. We then describe the process for decoding the most likely state assignment given a trained

[14]Different fields will use the terms CML or MMIE training; for consistency with previous CRF papers, we will refer to this criterion as CML in this paper.

CRF in Section III-A2. We defer a discussion of inference in nonlinear-chain structured CRFs until Section IV. In Section III-B, we describe some of the issues involved in defining feature functions for CRF systems. We end this section with a discussion of various techniques that have been employed in order to avoid overfitting to training data in Section III-C.

A. Training and Decoding

In Section II-A, we noted that the probability distribution of $P(S|O)$ (over an entire sequence) is defined over the cliques in the undirected graph representing the CRF [14]. For a linear-chain CRF, (5)–(7) show that the factors of the probability distribution are defined over single states and pairs of states. More generally, we can rewrite (5)–(7) as follows:

$$\begin{aligned} P(S|O) &= \frac{\prod_{t=1}^{T} \phi(s_t, s_{t-1}, O, t)}{Z(O)} \\ &= \frac{\prod_{t=1}^{T} \exp\left(\sum_i \lambda_i f_i(s_t, s_{t-1}, O, t)\right)}{Z(O)} \\ &= \frac{\exp\left(\sum_{t=1}^{T} \sum_i \lambda_i f_i(s_t, s_{t-1}, O, t)\right)}{Z(O)} \end{aligned} \tag{8}$$

where i ranges over the set of possible descriptions of the input O, and S corresponds to the unknown state sequence. Some functions f_i can be degenerate functions that ignore some of their inputs: this enables CRFs to have bias terms for states and transitions (ignoring the input O), or be state-only functions (ignoring s_{t-1}); generally functions that are conditioned on input O will only utilize a window of O around the current frame t (and sometimes only o_t).

1) Estimating Parameters of Linear-Chain CRFs: The objective of the linear-chain CRF is, given a set of observations O and the corresponding states S, to maximize the conditional likelihood of the labels given the observations $P(S|O)$.[15] Given (8), the question is how to estimate the λ_i parameters, which weight the contributions of each function to the probability of the sequence S. To estimate the parameters, we try to find the parameters that maximize $P(S|O)$, where S is the correct label sequence; increasing $P(S|O)$ will inherently decrease the probability $P(S'|O)$ of any incorrect sequence S'.[16] To do this, optimizing the CML criterion takes partial derivatives of the log probability $\mathcal{L} = \log P(S|O)$ with respect to the parameters $(\partial \log P(S|O)/\ \partial\lambda_i)$ in forming the likelihood gradient $\nabla\mathcal{L}$[17]

$$\nabla\mathcal{L} = \nabla_\lambda \log P(S|O) = \begin{bmatrix} \vdots \\ \frac{\partial}{\partial\lambda_i} \log P(S|O) \\ \vdots \end{bmatrix} \tag{9}$$

and then find the zero of the gradient, which maximizes $P(S|O)$, since it is concave [11].

Computing the gradient: Let the vector of functions $f_i(s_t, s_{t-1}, O, t)$ at a particular time t be denoted $\mathbf{f}(S, O, t)$ and the vector of weights $\boldsymbol{\lambda} = [\lambda_i]$. Since $\boldsymbol{\lambda}$ does not depend on t, following the conventions of [13], we can use $\mathbf{F}(S, O) = \sum_{t=1}^{T} \mathbf{f}(S, O, t)$ to rewrite $P(S|O)$ [see (8)] and $\mathcal{L}$

$$\begin{aligned} P(S|O) &= \frac{\exp(\boldsymbol{\lambda} \cdot \mathbf{F}(S, O))}{Z(O)} \\ \mathcal{L} &= \boldsymbol{\lambda} \cdot \mathbf{F}(S, O) - \log Z(O). \end{aligned} \tag{10}$$

For any one particular λ_i, the partial derivative of the log likelihood is

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\lambda_i} &= F_i(S, O) - \frac{\partial}{\partial\lambda_i} \log Z(O) \\ &= F_i(S, O) - \sum_{S'} \frac{\exp(\boldsymbol{\lambda} \cdot \mathbf{F}(S', O))}{Z(O)} F_i(S', O) \\ &= F_i(S, O) - \sum_{S'} P(S'|O) F_i(S', O) \end{aligned} \tag{11}$$

and, thus, the gradient is

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L} = \mathbf{F}(S, O) - \sum_{S'} P(S'|O)\mathbf{F}(S', O). \tag{12}$$

What this equation says is that when the difference between the observed functions and the expected value of the functions according to the probability distribution is zero, we have reached CML. The gradient over an entire training corpus can be computed by summing the gradients for each utterance. For those used to more traditional ASR terminology, Hifny and Renals [15] cast the log-likelihood

[15]One should take care not to confuse CML [maximizing $P(S|O)$] with the more traditional maximum likelihood, which is often cast in terms of maximizing the joint probability $P(S, O)$ or the data likelihood $P(O|S)$.

[16]See the relationship to MMI estimation in Section V-A.

[17]Given a training set of N IID examples, $\mathcal{T} = \{(S_1, O_1), \cdots, (S_N, O_N)\}$, the conditional log likelihood of the entire training set can be written as $\mathcal{L} = \sum_i \log P(S_i|O_i)$, and thus $\nabla\mathcal{L} = \sum_i \nabla_\lambda \log P(S_i|O_i)$. Since the gradient of the log-likelihood for the entire training set can be obtained by summing together the gradients for all training examples, the derivations in this section only describe the gradient computation for a single training example.

derivative as being a difference between stochastic counts of observations of the correct hypothesis and of all incorrect hypotheses: for state-level features, the state occupancy probabilities (typically known as gamma probabilities in HMM parlance [6]) determine the weighting of each observation; summing these weighted observations for the correct path and all paths gives the two types of counts.

The remaining question is how to efficiently compute the expected value of the features, that is, the second term in (12). For those acquainted with computing parameters of Gaussian acoustic models in HMM systems, the answer is exactly analogous: compute the local probability of being in a state s_t at a particular time t (for state-only features) or the local probability of a transition between s_{t-1} and s_t (for transition features), and multiply in the value of the observation feature to get the expected value. Sha and Pereira have a particularly nice formulation that sums the potentials for partial sequences in matrices via alpha–beta recurrences [13]: build a transition matrix M_t such that

$$M_t[s_{t-1}, s_t] = \exp(\boldsymbol{\lambda} \cdot \mathbf{f}(s_t, s_{t-1}, O, t)). \tag{13}$$

The value of each element $M_t[s_{t-1}, s_t]$ of the transition matrix is the weighted contribution of all functions related to that transition.[18] We can compute an alpha–beta recurrence, where the row vector alpha represents the accumulated potentials (not probabilities) from the beginning of the sequence, and the row vector beta represents the backward-accumulated potentials from the end of the sentence (again using notation from [13])

$$\begin{aligned} \boldsymbol{\alpha}_t &= \begin{cases} \boldsymbol{\alpha}_{t-1} M_t, & 0 < t \leq T \\ \mathbf{1}, & t = 0 \end{cases} \\ \boldsymbol{\beta}_t^\top &= \begin{cases} M_{t+1} \boldsymbol{\beta}_{t+1}^\top, & 1 \leq t < T \\ \mathbf{1}, & t = T. \end{cases} \end{aligned} \tag{14}$$

Given the alphas and betas we can now efficiently compute both the expectation of the features and the normalization term $Z(O)$

$$\sum_{S'} P(S'|O)\mathbf{F}(S', O) = \sum_t \frac{\boldsymbol{\alpha}_{t-1}(f_t * M_t)\boldsymbol{\beta}_t^\top}{Z(O)}, \quad \text{where } Z(O) = \mathbf{1}^\top \boldsymbol{\alpha}_\mathbf{T} \tag{15}$$

where the $*$ operator is component-wise multiplication.

[18]Here we assume that state functions are folded into this matrix by adding in a state vector to each column of the matrix, although it is more efficient to factorize the state functions separately in the subsequent alpha–beta computations.

Using the gradient for parameter estimation: Finding the zero of the gradient function directly is intractable, so many optimization techniques have been used for parameter estimation. Various papers have made comparisons of the different types of techniques in the NLP domain [3], [13], and the reader is referred to those works for a longer discussion.

The original approach used by Lafferty *et al.* [1] is an iterative scaling technique [in particular, the improved iterative scaling (IIS) algorithm] and is reminiscent of maximum entropy classifier training techniques [16]. This approach makes weight updates at every pass through the training data that are guaranteed to increase the posterior $P(S|O)$; the size of the update is related to the ratio of the empirical feature count to the expected count under the current model. Statistics can be collected in parallel across the corpus (much like standard HMM techniques), allowing for parallelized training.

However, iterative scaling techniques are generally slow. Hifny and Renals [15] propose a variant of the IIS algorithm called the approximate iterative scaling (AIS) algorithm, in which each parameter λ is only allowed to join the model (i.e., deviate from zero) when the absolute value of the partial derivative for that λ is greater than some threshold α. When a sparse feature representation is used (Section III-B), this can lead to a significant acceleration in training.

Several authors have pointed out that iterative scaling techniques are generally very slow to converge because they do not pay attention to additional information, such as the second derivative of the likelihood [3], [12], [13], [17]. The Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm [18] estimates the second derivative of the likelihood (i.e., what will increase the likelihood the fastest) by keeping track of previous gradients. Conjugate gradient methods have also been used to modify the update equation so that the update direction is a linear combination of the gradient and the previous search direction [13].

In contrast to the second-order techniques described above, other techniques used to speed convergence include stochastic gradient [4] techniques and Rprop [19]. In stochastic gradient-based techniques, the trainer randomly samples the corpus and updates the weights after collecting the gradient for some small n number of utterances (often $n = 1$). The net effect of having more frequent weight updates is that the weights converge much faster (on some tasks, more than an order of magnitude faster). The disadvantage to this technique is that for small n, one cannot effectively parallelize over training utterances; it may be more efficient to collect gradients from a training set using a large compute cluster and then use the batch update techniques listed above. A different strategy for fast-convergence batch training is embodied in the Rprop algorithm [19]: rather than directly using the gradient, Rprop uses the sign of each component of the gradient to determine the direction of weight adjustment. The amount

of adjustment does not depend on the gradient itself, rather on a step size that increases as long as the sign of the partial derivative does not change; the magnitude of the weight update decreases (and sign changes) with a change in gradient so that the system will come increasingly close to the zero gradient. Rprop is designed to be robust against large changes in the gradient (which can be caused by feature scaling issues; see Section III-B. While this algorithm was developed for training MLPs, it has been found to be effective in training CRFs as well [20].

While this paper has focused mostly on training with the CML criterion, it is possible to train CRFs with other optimization criteria.[19] As an example, we have the perceptron update [23], which replaces the expected feature count in (11) with the features corresponding to the current best hypothesis, effectively requiring a Viterbi decoding rather than the full forward decoding described above. This has the effect of minimizing prediction error on the training set: once the best hypothesis is the same as the reference hypothesis, the system no longer updates the weights. Other optimization criteria such as the minimum classification error (MCE) criterion [24] or a minimum tag error criterion (approximating the sentence labeling error and inspired by the popular minimum phone error (MPE) criterion [25] in speech recognition) [26] can also be used to train the model. An alternative approach to training CRFs is based on maximizing the margin between the reference labeling and all competing labelings [27]. We elaborate further on these large-margin techniques in Section III-C.

2) Decoding Linear-Chain CRFs: In decoding, we are interested in computing the most likely assignment to the variables S given O

$$S^* = \arg\max_S P(S|O) = \arg\max_S \frac{\exp\left(\sum_{t=1}^{T}\sum_i \lambda_i f_i(s_t, s_{t-1}, O, t)\right)}{Z(O)}. \quad (16)$$

Since the normalization constant $Z(O) = \sum_{S'} \exp(\sum_{t=1}^{T}\sum_i \lambda_i f_i(s'_t, s'_{t-1}, O, t))$ remains constant relative to an input O for any hypothesis S, computing the most likely assignment S^* does not require the computation of $Z(O)$

$$S^* = \arg\max_S \exp\left(\sum_{t=1}^{T}\sum_i \lambda_i f_i(s_t, s_{t-1}, O, t)\right). \quad (17)$$

Although the maximization in (17) involves an exponential number of state sequences S, the problem can be efficiently decomposed over the edges of the linear-chain CRF, thus allowing S^* to be computed efficiently with dynamic programming using the Viterbi algorithm. In particular, computing S^* can be accomplished by a recursion similar to the alpha–beta recurrence in (14)

$$\delta_t[s] = \begin{cases} \max_{s'} M_t[s', s]\delta_{t-1}[s'], & 0 < t \le T \\ 1, & t = 0. \end{cases} \quad (18)$$

Notice that the recursions presented above are identical to the alpha recursion presented in (14) with summation being replaced by maximization. Once the $\delta_t[s]$ have been computed for all t, s, the most likely assignment of state labels S^* can be computed starting from time T

$$s_t^* = \begin{cases} \arg\max_s \delta_T[s], & t = T \\ \arg\max_s M_{t+1}[s, s_{t+1}^*]\delta_t[s], & 1 \le t < T. \end{cases} \quad (19)$$

B. Defining Feature Functions for Log-Linear Representations

As noted above, log-linear models are a popular way to realize potential functions in CRFs; we make heavy reference to the functions $f_i(S, O, t)$ that relate the observations with states at various time steps t. The definition of functions varies from problem to problem (we describe how feature functions are defined for different problems in Section VI), but some general observations can be made about differing classes of features.

The features described in the ball-and-urn problem in Section II-A were based on discrete observations; direct models of observations are represented via binary functions representing the presence ($f = 1$) or absence ($f = 0$) of the observation, such as "is the previous word X?" [28], [29]. More general binary functions can be developed based on the domain: for example, in developing a part of speech tagger for German, observing that a word is capitalized and not sentence initial may be a strong indicator that the word has a nominal form; words in English ending in *-ing* are more likely to be participles.

Binary functions also play a role when observations are not involved: binary functions that are defined on the state–space variables are known as *bias* terms, and can be thought of playing an equivalent role to the Markovian prior in an HMM.

When the observations are continuous (for example, in speech recognition applications where the observations may be a multidimensional representation of the spectrum such as MFCCs), the modeler needs to take into consideration how to represent these in the log-linear model. One approach is to use the features directly: for example, when the original input and their squares are used, the feature representation provides the sufficient statistics for

[19] Discriminative training techniques for HMMs also have a close connection to CRF training; see [21] and [22] for a review.

a Gaussian model [4], [5]. An example for this conversion is discussed in Section V. The quadratic transformation of MFCC features in this system can be thought of as an expansion of the original MFCC features to a higher dimensional space: this is necessary because the phonetic classes are not log-linearly separable.

It is possible to combine discrete and continuous observations in one system (e.g., Chien and Chueh [30] augmented the observation statistics above with n-gram statistics over the state symbols). It is also possible to use external classifiers to convert the original features into a feature representation that may be more amenable to log-linear classification. One approach is to use the log likelihoods and their partial derivatives from maximum-likelihood trained Gaussians to provide a scoring space relating continuous observations to the underlying states [31], [32]. Posterior classifiers have also been used to assign the probability of putatively important events describing the input, giving a softer version of the binary indicator function [33]. Log likelihoods of sequence models are usually represented in weighted finite state transducers (WFSTs) for efficient decoding. Learning of WFSTs could be achieved by encoding the arc weights as the features in a CRF, which is trained with the CML criterion or the perceptron update. Such a discriminative learning method effectively allows for joint training of different models composed with WFSTs, e.g., acoustic models, pronunciation models, and language models for speech recognition [34]–[36].

Binary indicator functions on discrete inputs, especially when the observation space is large, lead to a relatively sparse representation (many feature functions are zero for a particular input). The continuous functions described above, however, tend to be lower dimensional, but dense; most features are nonzero. While theoretically this has no impact on the algorithm, from an engineering perspective it is important: using sparse features, it is more efficient to use sparse matrix multiplication in computing the recurrences in (14), whereas dense matrix multiplication packages can speed computation on a multicore machine for dense features.[20] However, continuous observations can also have sparse representations: for example, finding the k highest scoring Gaussians [15], [37] or vector quantizing the observation space [38] can lead to sparse spaces, which can motivate particular learning algorithms such as the AIS algorithm described above [15].

An interesting thread running through this line of research is that features can be derived from a first-pass statistical model, which in effect forms a set of basis functions for a higher dimensional space to facilitate classification: divisions between phonetic classes that are not linearly separable in a low-dimensional space may be separable in the higher dimensional space. In that way, it is reminiscent of kernel-based techniques for dimensionality expansion, such as that used in support vector machine classification [39]. In fact, Lafferty *et al.* [40] explicitly use feature functions that are generated via kernel methods: a kernel CRF (KCRF). In a KCRF, a basis function is used to construct feature functions from the training data. Since this can result in a large number of feature functions, a greedy selection algorithm selects a representative sample from the possible feature functions that fit the training set. This technique showed improvements over kernel-based logistic regression models for protein sequencing [40].

Feature functions in CRFs need not be defined *a priori*, but may be learned automatically in a process that is analogous to feature induction in the hidden layer of an MLP; features at the input layer are projected nonlinearly into a higher dimensional space [41]–[43]. These approaches are closely related to work by Kingsbury [44], which discusses training MLPs with sequential classification criteria such as MMI or MPE for neural network acoustic modeling. Peng *et al.* [41] observe large gains over standard CRFs on a protein structure prediction task using this approach.

Extensions of the linear-chain CRF model based on recent work on deep architectures in machine learning have also been explored in the literature. Deep belief networks [45] incorporate multiple hidden layers, each of which successively transforms the input representation into an intermediate hidden layer representation used as the input to the next layer. A powerful feature of these models is their ability to learn these hidden representations in a completely unsupervised fashion; once the hidden layer weights have been learned, the entire network can be trained discriminatively in a supervised manner. In the deep-structured CRF [46], first- or zeroth-order linear-chain CRFs are stacked together so that the marginal probabilities obtained from the outputs of a lower layer serve as the inputs to the higher layer. The model can be further extended where label sequences are known but the corresponding segmentations are unknown [2].

In concluding the discussion on features, it is helpful to consider some practical aspects of feature design in these systems. It is clear from the discussion above that CRFs can utilize a wide range of feature function definitions. However, the interaction between features and learning rates is crucial. One important consideration is feature scaling: it is often beneficial to have most feature activations lie within a limited range (say, $[0, 1]$ as in posteriors, or $[-k, +k]$ for small k, for mean and variance normalized MFCCs), as bias terms for states and transitions are typically set to 1; having features in the same range enables the system designer to use the same learning rate for all weight updates. The relationship between learning rates and feature scale is approximately inverse: consider a single feature system with posterior $\exp(\lambda f)/Z(O)$ and a

[20]For those using off-the-shelf toolkits, it may be important to pay attention to the internal algorithm to ensure that it matches the desired feature space.

scaled version $\exp(\lambda 2f)/Z(O)$. To express the same distribution, the $2f$ system will require λ to be half the size; smaller steps in changing λ (i.e., the learning rate) are required. Wiesler and Ney have shown, with theoretical analysis and empirical justification, that mean and variance normalization and decorrelation for features are important for the good convergence behavior of gradient-based optimization algorithms for log-linear models [47].

The weighted sum of the features across a sequence also becomes an important consideration. If the weighted sum exceeds 700 or so within a sequence, computing the forward–backward probabilities will overflow the size of double-precision floating point using the straightforward implementation described in Section III-A1.[21] This is typically not an issue for the natural language community, where sequences are short, but for the ASR community utterances can easily exceed thousands of frames. Converting all of the operations to log space has the advantage of allowing many more features, at the cost of training speed: the matrix-multiply operations in (14) become much more cumbersome.

C. Avoiding Overfitting

As is noted throughout the machine learning literature, one of the potential hazards of learning parameters from a particular training set is overfitting: the parameters will match the particular training set well but not generalize to a separate test set. There are four basic techniques that have been used (alone or in combination) in CRF training to minimize the effects of overfitting.

The most common approach is regularization: a penalty term is added to the log likelihood in order to ensure that the weights of the CRF remain small. A common penalty is the ℓ_2-norm, which changes the log likelihood of the system [from (10)] to

$$\mathcal{L}_{\ell_2\text{-penalized}} = \boldsymbol{\lambda} \cdot \mathbf{F}(S, O) - \log Z(O) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma^2} \quad (20)$$

where σ^2 indicates a free parameter that determines the width of a spherical prior over the weights [1], [3]. When the partial derivative of the penalized likelihood is taken with respect to a particular λ_i (which determines the scale of the update), the derivative is decreased by λ_i/σ^2. Thus, for a fixed σ^2, larger values of λ_i will receive proportionately smaller increases in the gradient ascent. When σ^2 is large, this term becomes increasingly vanishing; for NLP tasks, a value between 10 and 100 works relatively well [12], although the sensitivity of the answer to this parameter seems to be quite low [3].

Hifny and Renals [15] find, for their sparse augmented representation, that the ℓ_1-norm is more appropriate; their penalized log-likelihood equation takes the form

$$\mathcal{L}_{\ell_1\text{-penalized}} = \boldsymbol{\lambda} \cdot \mathbf{F}(S, O) - \log Z(O) - \alpha \sum_{i=1}^{n} |\lambda_i| \quad (21)$$

where α is a parameter used to control the influence of the ℓ_1 regularizer. This regularizer is called the Lasso penalty, and they claim that this penalty will often drive many of the λ weights to zero [48]—a very desirable property with high-dimensional sparse representations.

Another approach to avoiding overfitting is one proposed for perceptron learning, most recently by Collins [23], which has been adapted for use in CRFs [4]. The voted perceptron works by continually maintaining two sets of weights: the current set of unnormalized weights, and a running average of all weights maintained by the system—in essence, the average of all updates made to the system during training. In our experience, this technique is very effective when using stochastic gradient training of CRFs, because it mitigates the effects of any one individual example's updates to the weights; in the iterative scaling variants of training, updates are withheld until the end of the training epoch and, thus, all examples have an effect on the weight update contemporaneously.

Cross-validation techniques on held-out data can also be used to judge the progress of the training; these techniques have long been used for training, e.g., neural networks [49]. The held-out set is decoded at set intervals (often an epoch of the training data). While the log posterior of the labels continues to increase with further iterations of training, the prediction accuracy on a development test set often levels off, and suggests that any further training will overfit the data. One can combine these techniques as well (e.g., using weight averaging with cross validation).

In the context of structured prediction and avoiding overfitting, it would be amiss not to mention large-margin approaches: changing the training criterion so that the margin between the correct label sequence and competing sequences is maximized, rather than maximizing the likelihood [27]. Related to this approach are structured support vector machines (structured SVMs) [50], [51]: these are similar to CRFs in that both approaches minimize a convex regularized loss function (the negative regularized conditional log likelihood in (20) is the loss function for CRFs). The optimization problem in structured SVMs introduces a max-margin regularization term and slack variables to prevent overfitting. We refer

[21] In our experience, standard bookkeeping techniques that renormalize the alpha–beta computation help, but only up to a certain point. The main issue becomes the computation of $Z(O)$, where the renormalizations must be taken into account. This issue is discussed in the HMM context in [6].

the reader to [52] for a detailed comparison of various structured discriminative models, particularly in speech recognition.

IV. GOING BEYOND LINEAR-CHAIN CRFs

The discussion of CRFs has so far been restricted to linear chains, so named because they correspond to the graphical model structure in Fig. 2(c). Before discussing possible generalizations of the linear-chain CRFs, we first explore the connection between graphical model structure and the corresponding probability distributions in greater detail. The discussion in the following section must necessarily be brief; for a more detailed description we refer the interested reader to [53].

A. Undirected Graphical Models and CRFs

An undirected graphical model such as the one in Fig. 3 is intended to represent the conditional independence assumptions inherent in the probability distribution defined on a set of random variables. The vertices of the graph represent particular random variables, and there is no loss of generality in assuming that the vertices are labeled by the corresponding random variable, as we have done in Fig. 2(c). We may thus refer to the neighbors of a particular random variable s in the graph, by a set of vertices (denoted by $\mathcal{N}(s)$) that are adjacent to it in the graph. The edges in the undirected graphical model encode the conditional independence assumptions among the variables, in terms of the Markov property: a random variable s is independent of any other random variable in the graph given the values of its neighbors. A graph defined on the set of variables (S, O) is a CRF if the variables in S obey the Markov property with respect to the graph when conditioned on the observations O [1]

$$P(s|s', \mathcal{N}(s), O) = P(s|\mathcal{N}(s), O). \tag{22}$$

Now, the structure of a graph is completely specified by stating the set of cliques, $\mathcal{C}$ in the graph, where a clique $\mathbf{c} \in \mathcal{C}$ is a set of vertices that are mutually connected by an edge. A surprising fact about CRFs (and random fields in general) is that the set of all cliques in the graph also completely characterizes the set of probability distributions on the graph variables that satisfy the Markov property with respect to the graph. If we denote by $\mathbf{s_c}$ the set of random variables in S that correspond to a particular clique $\mathbf{c} \in \mathcal{C}$, the Hammersley–Clifford theorem [14] states that a positive probability distribution $P(S, O)$ satisfies the Markov property in (22) if and only if it can be written in the form

$$P(S|O) = \frac{1}{Z(O)} \prod_{\mathbf{c} \in \mathcal{C}} \phi(\mathbf{s_c}, O) \tag{23}$$

where $\phi(\mathbf{s_c}, O)$ are a set of positive functions defined on the random variables and $Z(O)$ is a normalization term that ensures that (23) forms a valid probability distribution

$$Z(O) = \sum_{S'} \prod_{\mathbf{c} \in \mathcal{C}} \phi(\mathbf{s'_c}, O). \tag{24}$$

In the particular case where the graphical structure corresponding to the CRF is a linear chain, the cliques in the graph correspond to edges s_{t-1}, s_t, and individual nodes s_t. Consequently, the corresponding functions are represented as $\phi(s_{t-1}, s_t, O)$ and $\phi(s_t, O)$, as defined in (6) and (7).

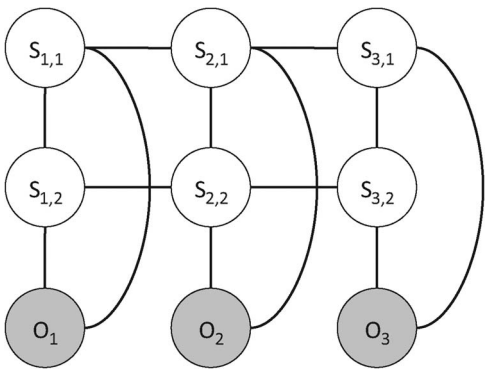


Fig. 3. *An example of a more general (nonlinear-chain) CRF, with an interlinked chain of two state variables depending on a sequence of observations. As we discuss in Section VI-A, such a graphical model structure has been used to jointly model noun phrase segmentation (the upper chain of variables $s_{i,1}$) and part of speech tags (as part of the lower chain of variables $s_{i,2}$) in [11].*

B. General CRFs

The discussion in the previous section, and (23) in particular, provides a clear roadmap for the extension of CRFs beyond a linear-chain structure. All that is required is that a suitable graphical structure be defined that captures the conditional independence assumptions that one would want to hold for the set of random variables S given the observations O and to choose an appropriate form for the feature functions $\phi(\mathbf{s_c}, O)$.

In many applications suitably modeled by a general CRF, it is useful to think of the complete graphical structure as corresponding to a set of clique templates that are repeated over time to obtain a dynamic conditional random field (DCRF) [3], [11]. This is similar to the process by which a Bayesian network is unrolled over time to obtain a dynamic Bayesian network [54]. We use the notion of

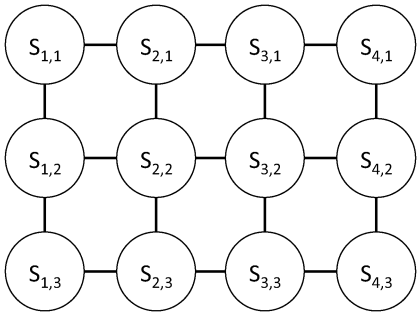


Fig. 4. *Two-dimensional (grid structured) CRF. For clarity, links to observations have been omitted. We have previously applied such a graphical model structure for speech segregation [55], where each node $s_{i,j}$ represents a time-frequency unit. We describe this approach in greater detail in Section VI-C.*

"repeating over time" loosely here, since there are alternative graphical structures such as 2-D grids (Fig. 4) that are suitable for image processing applications; the essential feature that we wish to stress is that it is useful to think of the overall structure as being decomposable into a number of templates that may be repeated to obtain the overall structure. With this understanding, let us denote the set of variables associated with a clique template $\mathbf{c} \in \mathcal{C}$, at a particular "time" t, as $\mathbf{s}_{t,\mathbf{c}}$. If we choose to represent the set of functions $\phi(\cdot)$ of (23) to be in the exponential family, we may write

$$P(S|O) = \frac{1}{Z(O)} \prod_{\mathbf{c}\in\mathcal{C}} \prod_{t=1}^{T} \exp\left(\boldsymbol{\lambda}_{\mathbf{c}}^{T} \cdot \mathbf{f}(\mathbf{s}_{t,\mathbf{c}}, O)\right) \qquad (25)$$

where $\boldsymbol{\lambda}_{\mathbf{c}}$ is the set of parameters associated with the clique template $\mathbf{c}$. In the case of a linear-chain CRF, clique templates could be defined corresponding to individual edges $\mathbf{c_1}$ and nodes $\mathbf{c_2}$, thus $\mathbf{s}_{t,\mathbf{c}_1} = \{s_{t-1}, s_t\}$ and $\mathbf{s}_{t,\mathbf{c}_2} = \{s_t\}$. Note that the characterization of the general graphical structure in terms of templates makes explicit an additional feature of general graphical models that must be exploited in practical systems: the weights $\boldsymbol{\lambda}_{\mathbf{c}}$ associated with clique templates are explicitly tied across repetitions.

C. Inference in General CRFs

The parameters $\{\boldsymbol{\lambda}_{\mathbf{c}}\}$ in (25) can be learned analogously to a linear-chain CRF, as described in Section III-A1, which involves the computation of the marginal probabilities $P(\mathbf{s}_{t,\mathbf{c}}|O)$ for variables in the graph. In the case where the underlying CRF graph structure corresponds to a tree,[22] these marginal probabilities can be computed exactly by a generalization of the forward–backward algorithm for linear chains, which is known as the sum–product algorithm or belief propagation [56], [57].

For non-tree-structured CRFs, the marginal distributions can be computed exactly using the junction tree algorithm [53]. The basic idea behind this algorithm is to successively cluster the variables in the graph to obtain a new tree-structured graph, called the junction tree, whose vertices are subsets of the original graph nodes and then to run inference algorithms on the junction tree. Unfortunately, the junction algorithm is exponential in the tree width of the graph (one less than the size of the largest clique in the junction tree), and, thus, the cost is prohibitive for graphs of large tree-width (2-D grids, for example).

In cases where exact inference is intractable, it is possible to utilize a number of approximate inference techniques. Conceptually, one of the simplest approaches is loopy belief propagation, which is an application of the basic belief propagation algorithm to graphs with cycles. Although the algorithm is not provably correct for such graphs, applying the algorithm in these conditions has been found to be empirically successful for some applications [58]. It should be noted, however, that in general graphs, the algorithm may not converge, or if it does converge, it may not converge to the correct marginals for graphs with cycles. An alternative approach is to use Markov chain Monte Carlo (MCMC) techniques such as Gibbs sampling to obtain samples from the distribution that can then be used to approximate the marginals of interest. Since Gibbs sampling requires one to sample from the distribution of a particular random variable conditioned on all other variables in the network, the Markov properties of the graph [see (22)] can be readily exploited to simplify the sampling process. MCMC techniques can, however, be very slow in practice, which is an issue if the marginal distributions in the graph need to be computed repeatedly during training for each input example. Variational techniques [59] can also be used for approximate inference. A full discussion of variational methods is outside the scope of this paper; we refer the interested reader to [60].

Although exact inference using the junction tree algorithm is intractable for a number of graphical structures, knowledge of deterministic task-dependent constraints can greatly simplify the inference problem. For example, we have considered the use of CRFs to model the various streams of articulators (lips, tongue, glottis, velum, etc.) while explicitly accounting for the fact that the articulators can desynchronize from each other in conversational speech [61]. Although the factor graph for this model appears fairly complicated, the deterministic constraints that we impose on the problem allow inference to be carried out in an equivalent linear-chain graphical model, with a manageable state space.

[22] More technically, the sum–product algorithm can be used for computing marginals in the case when the corresponding factor graph [56] is a tree.

D. Introducing Hidden Variables in Training: HCRFs

For many problems of interest, there is unknown substructure that links the state space with observations: for example, in speech recognition, one often uses a subphonetic state representation to statistically describe the beginning, middle, and end of a subword unit (phone); in many continuous domains, a mixture of Gaussians is used for observation probabilities, where the mixture class is unknown. During training, the system will often have access to some labels of interest, but will not have a labeling available for substructures. Moreover, the substructure is often not desired at decode time, so it can be marginalized out of the probability estimation.

Hidden CRFs [4], [62] take the approach that an observed state is paired with one that is hidden during training; the CML training algorithm can be adapted to estimate hidden state values using the expectation–maximization algorithm [63] or to compute the (n-best) most likely hidden sequences during training. These approaches have the advantage of being able to represent a more complex state space than the log-linear probability model in a standard CRF; however, the optimization problem is no longer convex, so a global optimum on the training set is not guaranteed.

E. Segmental CRFs

Semi-Markov CRFs [64] or segmental conditional random fields (SCRFs) [20] (Fig. 5) are extensions of linear-chain CRFs that allow each of the label states to span a segment of consecutive observed units, each of which is in turn assigned the same label. These models relax the Markov assumption from the frame level to the segment level, while the transitions within segments can be non-Markovian, which enables long-span dependencies.

A segmental CRF models a segment-level label sequence $S = \{s_1, s_2, \ldots, s_K\}$ along with the corresponding segment boundaries (encoded in terms of end times E). Let $E = \{e_1, e_2, \ldots, e_K\}$ denote the end times of the sequence of observation segments corresponding to S, and $o_{e_{j-1}}^{e_j}$ denote the observation segment from $o_{e_{j-1}}$ (exclusive) to o_{e_j} (inclusive). The clique template $\mathbf{c}$ at a particular segment index j could be defined on s_{j-1}, s_j, e_{j-1}, and e_j, and the features f_i could be defined on a pair of consecutive segment labels and the corresponding observed segment. According to (25), we can model the joint probability distribution of S and E conditioned on O as

$$P(S, E|O) = \frac{\exp\left(\sum_{j=1}^{|E|} \sum_i \lambda_i f_i\left(s_{j-1}, s_j, o_{e_{j-1}}^{e_j}\right)\right)}{Z(O)} \tag{26}$$

where

$$Z(O) = \sum_{\substack{S,E \text{ s.t.} \\ |E|=|S|}} \exp\left(\sum_{j=1}^{|E|} \sum_i \lambda_i f_i\left(s_{j-1}, s_j, o_{e_{j-1}}^{e_j}\right)\right). \tag{27}$$

One has a choice whether to optimize the joint label and segmentation space $P(S, E|O)$ (as in [64] and [65]), or only the label sequence $P(S|O)$ (as in [20]). In the former case, both correct label sequences S and associated correct segmentations E are known during training, thus making it a convex optimization problem. In the latter case, only label sequences are required for training but the correct segmentations are treated as hidden structure, and one needs to sum over all possible segmentations E' for S [as shown in (28)] leading to a nonconvex optimization problem, similar to HCRFs; it is an open question whether the marginalization over segmentations is necessary for most tasks

$$P(S|O) = \frac{\sum_{\substack{E \text{ s.t.} \\ |E|=|S|}} \exp\left(\sum_{j=1}^{|E|} \sum_i \lambda_i f_i\left(s_{j-1}, s_j, o_{e_{j-1}+1}^{e_j}\right)\right)}{Z(O)}. \tag{28}$$

The standard forward–backward algorithm for linear-chain CRFs can be extended in either case for efficient parameter estimation and inference, with an increase in complexity by a factor of the maximum duration of segments. Furthermore, we have shown that if each observation-dependent transition feature is defined on a fixed number of observation units instead of a variable-length observation segment, the time complexity of the forward–backward algorithm for SCRFs can be reduced to that of standard linear-chain CRFs [65] under mild assumptions. We refer interested readers to [20], [64], and [65] for details of these algorithms.

The SCRF is a desirable framework for tasks where the labels of interest are inherently segmental in nature, for

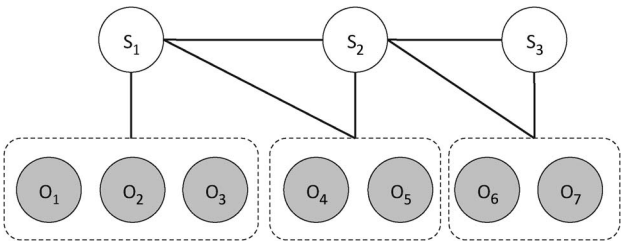


Fig. 5. *Example semi-Markov CRF or SCRF graphical model. Unlike a regular CRF, each SCRF state can refer to a variable-length sequence of observations, also known as a segment. For example, in speech recognition, each of the states could represent a word along with the corresponding segment of acoustic observations [20].*

example, named entity recognition [64], [66], Chinese word segmentation [66], [67], word recognition [20], [68], and phone classification and phone recognition [38], [65].

F. Conditional Augmented Models

Conditional augmented (C-Aug) models [31] are discriminative segment-based models that incorporate the sufficient statistics of observation log likelihoods from a base generative model. The models predict a single multiclass label for a sequence of observations that are assumed to share the same label. The posterior probability of the single label s for the entire sequence O is given by

$$P(s|O) = \frac{\exp(\boldsymbol{\alpha}^\top \mathbf{T}(s, O; \boldsymbol{\lambda}))}{Z(O; \boldsymbol{\lambda}, \boldsymbol{\alpha})} \tag{29}$$

where $\boldsymbol{\lambda}$ refer to the parameters of the base model, and $\mathbf{T}(s, O; \boldsymbol{\lambda})$ are the sufficient statistics of the observation log likelihoods from a base generative model. C-Aug models have the ability to represent a wide range of temporal and spatial dependencies since the feature functions can be dependent on the entire observation sequence.

C-Aug models can be extended to sentence models for recognition, with a sequence of segmental labels. These models are then similar to SCRFs except that the features are observation log likelihoods and their derivatives, which are obtained from a base generative model, when $\boldsymbol{\lambda}$ are fixed or pretrained. A first-pass lattice from a base model (e.g., HMMs) is used to significantly constrain the possible segmentations and label sequences for C-Aug models. Additional details about this can be found in [69].

V. Relationships of CRFs to Other Classifier Technologies

In the previous sections, we detailed the particular mathematical techniques that allow for CML training of CRFs over a range of feature definitions. In this section, we relate in particular the structure and learning methods for CRFs to several current classifier technologies used in speech and language technologies.

A. Relationship to HMMs

Many comparisons have been made between CRFs and HMMs [3], [4], but it is instructive to show the relationship between these models in a slightly more explicit manner. We highlight here work that is explained more fully in Heigold *et al.*'s paper [5] comparing Gaussian-based HMMs for continuous observations and log-linear models, but we walk through the explicit steps in understanding the relationships between Gaussian and log-linear models, using ASR as a domain example.

In speech recognition, HMMs typically model the probability of a word sequence W given an observed acoustics sequence O via a state sequence S over time $t = 1 \ldots T$

$$\begin{aligned}
&\arg\max_W P(W|O) \\
&\quad = \arg\max_W P(O|W)P(W) \\
&\quad \approx \arg\max_W \max_S P(O|S)P(S|W)P(W) \\
&\quad \approx \arg\max_W \max_S \left[\prod_{t=1}^{T} P(o_t|s_t)P(s_t|s_{t-1})\right] P(W).
\end{aligned} \tag{30}$$

Let us focus on the acoustic computation in the inner brackets, and first consider the case where $P(o_t|s_t)$, the acoustic likelihood for a particular state, is represented by a single Gaussian with a diagonal covariance matrix. In this case, the likelihood for data o_t (with dimensionality n) at state $s_t = s$ with multivariate mean $\boldsymbol{\mu}$ and standard deviation vector $\boldsymbol{\sigma}$ (where $i = 1 \ldots n$ ranges over the components of these vectors and both are implicitly indexed by s) is given by

$$P(o_t|s_t) = \frac{1}{(2\pi)^{\frac{n}{2}} \prod_i^n \sigma_i} \exp\left(-\frac{1}{2}\sum_i \frac{(o_{t,i} - \mu_i)^2}{\sigma_i^2}\right). \tag{31}$$

We can rewrite this equation so that it is an exponential function of a sum over the dimensions i of terms of o_i and o_i^2. Notice that

$$\frac{1}{(2\pi)^{\frac{n}{2}} \prod_i^n \sigma_i} = \exp\left(-\log\left(\prod_i (2\pi)^{\frac{1}{2}}\sigma_i\right)\right) \tag{32}$$

$$= \exp\left(\sum_i -\frac{1}{2}\log\left(2\pi\sigma_i^2\right)\right). \tag{33}$$

Since

$$-\frac{1}{2}\sum_i \frac{(o_{t,i} - \mu_i)^2}{\sigma_i^2} = \sum_i \left(\frac{-o_{t,i}^2}{2\sigma_i^2} + \frac{o_{t,i}\mu_i}{\sigma_i^2} + \frac{-\mu^2}{2\sigma_i^2}\right) \tag{34}$$

we can rewrite (31) by gathering terms in $o_{t,i}$

$$P(o_t|s_t) = \exp\left[\sum_i \left(c + \frac{o_{t,i}\mu_i}{\sigma_i^2} + \frac{-o_{t,i}^2}{2\sigma_i^2}\right)\right],$$
$$\text{where } c = -\frac{1}{2}\left(\log 2\pi\sigma_i^2 + \frac{\mu_i^2}{\sigma_i^2}\right). \tag{35}$$

We can see that this Gaussian model can be expressed as a sum of weighted feature functions $\mathbf{f}$ and weights $\boldsymbol{\lambda}$. Let

$$\begin{aligned} f^0_{i,s_t} &= \delta(s_t = s)\forall i \\ f^1_{i,s_t} &= \delta(s_t = s)o_{t,i} \\ f^2_{i,s_t} &= \delta(s_t = s)o^2_{t,i} \\ \lambda^0_{i,s} &= -\frac{1}{2}\left(\log 2\pi\sigma_i^2 + \frac{\mu_i^2}{\sigma_i^2}\right) \\ \lambda^1_{i,s} &= \frac{\mu_i}{\sigma_i^2} \\ \lambda^2_{i,s} &= \frac{1}{2\sigma_i^2} \end{aligned}$$

where the delta function selects for a particular state s, that is, when a postulated HMM state $s_t = s$ the delta function takes on a nonzero value, and is zero for $s_t \neq s$.[23] Replacing terms in (35) with the above terms, we obtain the following expression (for a particular time t):

$$P(o_t|s_t) = \exp\left(\sum_{i=1}^{n}\sum_{j=0}^{2}\lambda^j_{i,s}f^j_{i,s_t}\right). \tag{36}$$

In an HMM, however, it is not only important to model the acoustic probabilities at each state, but also the transitions between states. We can easily incorporate state-to-state transition probabilities by having transition functions between pairs of state values (s, s')

$$\begin{aligned} f^{\text{trans}}_{s,s'} &= \delta(s_t = s, s_{t+1} = s') \\ \lambda^{\text{trans}}_{s,s'} &= \log P(s_{t+1} = s'|s_t = s) \end{aligned}$$

and, thus, when considering a state sequence $S = \{s_0, s_1, \ldots, s_t\}$ one can compute the likelihood in terms of weighted functions of the state s_t and transition (s_t, s_{t+1})

$$P(O, S|W) = \exp\left(\sum_t\left[\left(\sum_i\sum_j\lambda^j_{i,s}f^j_{i,s_t}\right) + \lambda^{\text{trans}}_{s_t,s_{t+1}}f^{\text{trans}}_{s_t,s_{t+1}}\right]\right). \tag{37}$$

One can analogously introduce feature functions that incorporate the probability of the words $P(W)$. When expressing a Gaussian HMM as a CRF, (37) is compatible with the numerator of the CRF equation, that is, a likelihood-based HMM can be represented as an unnormalized CRF. Remember that the partition function (denominator) of the CRF is the sum of the exponential functions over all paths

$$P(W|O) \approx \max_S \frac{P(O, S|W)P(W)}{\sum_{S'} P(O, S'|W)P(W)}. \tag{38}$$

This is the same quantity that is maximized during MMIE training [70].[24]

Thus, the structure of the equations when sufficient statistics are used as feature functions within a CRF is the same as that of MMIE. Both methods also have parameter updates for the probability distribution that are based on gradient–ascent methods. In practice, MMIE leads to a constrained optimization problem, in that the updated HMM parameters must still form a probability distribution [70], whereas CML updates for log-linear CRFs lead to an unconstrained optimization because potentials are globally normalized.

It should also be pointed out that it is possible to represent a subclass of linear-chain CRFs within an HMM, effectively giving a discriminatively trained HMM. Heigold *et al.* show that, for any set of linear-chain CRF parameters $\boldsymbol{\lambda}$ that have time-aligned state-observation relationships only (i.e., only the observation at time t is associated with the state at time t and no transition–observation feature functions), one can derive a set of HMM parameters $\boldsymbol{\theta}$ that can exactly represent the same probability distributions [72]. They demonstrate this by showing equivalence of the two models using a semantic concept tagging task. The equivalence has been shown to be bidirectional, and has been extended to mappings between HMMs with a mixture of Gaussian observations and a mixture of log-linear models [5].[25] This equivalence proof makes use of linear-chain CRFs with subphonetic HMM state sequences (S) as hidden variables, also known as HCRFs. We refer the reader to Section IV-D for more details on HCRFs.[26]

B. Relationship to Perceptrons and MLPs

The basic equations for CRFs also bear some resemblance to MLPs, which have been used for local phone classification tasks in ASR for many years ([73] *inter alia*), and as part of larger hybrid artificial neural network–hidden Markov model (ANN–HMM) and tandem ANN–HMM systems [74]. MLPs attempt to discriminate

[23]Note that while the functions vary as a combination of the input and postulated state over time, the weights (λ) are not dependent on the time t, in keeping with the stationarity assumption of HMMs.

[24]A good review reference for MMIE can be found in [71].

[25]It should be noted that the results here apply to one form of CRF: linear-chain models that do not have transition functions that depend on observations.

[26]We note that this idea of using HMM states as hidden variables was first introduced by Gunawardana *et al.*. They also point out that this approach extends naturally to hidden component weights of mixtures of Gaussians that are typically used as observation densities in speech recognition [4].

between classes by projecting the input to a high-dimensional space (via one or more hidden layers) and then combining these inputs to give an estimate of (phone) class membership. At each layer, a (possibly nonlinear) function is used to determine the outputs of the layer; Bridle [75] showed that when the softmax function was used (and trained appropriately), the outputs could be interpreted as a posterior probability. In short, if W_c^T is a vector of weights between the hidden nodes and the node for class c in the output layer, and X is the vector of outputs from the hidden layer of an MLP (which is in turn a nonlinear function of the input or a previous hidden layer), then the probability of class c given the input can be computed as

$$P(c|\text{input}) = \frac{\exp\left(W_c^T X\right)}{\sum_{c'} \exp\left(W_{c'}^T X\right)}. \tag{39}$$

The form of this equation is remarkably similar to the CRF equation where only state feature functions are considered, that is, the weights W_c^T correspond to the CRF weights λ and the X are the individual feature functions. For single layer perceptrons (SLPs) with a softmax transfer function, the form of the equation is in fact identical to a maximum entropy classifier, and, thus, the CRF extends this idea by allowing statistical dependencies between adjacent states.[27] Maximum entropy training is equivalent to finding the CML of (39) over all the training data; following Klein [8] one can write the log likelihood of an (unregularized) MaxEnt model where the feature functions are based on the data as

$$\begin{aligned} L(W) &= \log \prod_i P(c_i|X_i, W) = \log \prod_i \frac{\exp\left(W_{c_i}^T X_i\right)}{\sum_{c_i'} \exp\left(W_{c_i'}^T X_i\right)} \\ &= \sum_i \left(W_{c_i}^T X_i - \log \sum_{c_i'} \exp\left(W_{c'}^T X_i\right) \right). \end{aligned} \tag{40}$$

As derived earlier, this quantity is maximized through gradient search over W.

Similarly, with softmax perceptrons, one tries to minimize an error criterion—the cross entropy of the target distribution and the probability distribution currently predicted given the weights—by searching with the gradient of the error function. Given a true distribution P_T and predicted distribution P_P, the cross entropy $H(P_T, P_P) = \sum_i \sum_{c_i'} -P_T(c_i') \log P_P(c_i')$ is a measure of distortion between these distributions. In the specific case where the "true" (target) distribution is a one-hot encoding (the correct class has probability 1, the incorrect classes 0), this reduces to the log probability of the correct class

$$H(P_T, P_P) = \sum_i -\log P_P(c_i). \tag{41}$$

Given that the forms of P_P are the same for the MaxEnt and softmax perceptron classifiers, the partial derivatives of (41) with respect to the weight are identical to those of MaxEnt classifiers for the case of one-hot targets. In trying to minimize the cross entropy, therefore, the one-hot softmax perceptron is identical to the maximum entropy formulation.

The discussion above relates softmax perceptrons and MaxEnt classifiers; the additional power of an MLP is in its hidden layers—these allow for nonlinear boundaries between classes to be learned. On the other hand, SLPs, MaxEnt models, and CRFs (as well as SVMs) are linear classifiers of their input. CRFs and SVMs can model nonlinear functions by suitably transforming the input into a nonlinear space (through explicit design of feature functions in CRFs such as second-order statistics, or by kernel functions in SVMs). However, if one considers the MLP's hidden layer (X, often the output of a sigmoid function applied to weighted linear inputs) as a set of automatically learned feature functions in an exponential model, we can utilize these functions with other log-linear classifiers. Recent approaches based on learning deep architectures [45] can be thought of as intermediate between these positions. The hidden layer representations in the deep belief network are initially learned directly from the data in an unsupervised fashion as part of the pretraining step; the weights of the network can then be discriminatively optimized for classification in a supervised fashion.

The relationships between neural networks and CRFs has led to a number of exciting hybrid approaches, as discussed previously in this paper. Using sequence-based optimization criteria such as CML/MMIE has improved local posterior estimation in neural networks [44]. Similarly, using MLP or deep neural network techniques can automatically learn the feature functions for CRFs [42], [46].

VI. APPLICATIONS IN AUDIO, SPEECH, AND LANGUAGE PROCESSING

To this point, we have talked about CRF technology from the perspectives of historical development, uncovering their mathematical structure, and pointing out their relationship to other common classifier techniques. In this section, we explore the involvement of CRFs in natural language, speech, and audio processing tasks; we give brief summaries of different applications of CRFs by defining the problem in terms of the state sequence S that is desired

[27] One can also formulate MLPs with dependencies between states (see, for example, the REMAP paradigm [76]), although the training regimen becomes much more complex and computationally expensive than the usual MLP training method.

and the observations O. The aim of this section is not to be an exhaustive review of all possible applications, but rather to highlight the diversity of the range of problems that can be addressed using CRFs.

A. Natural Language Processing

Consider a series of operations that could be applied to a natural language sentence: one could begin with low-level tasks such as predicting part-of-speech (POS) tags or recognizing named entities, followed by operations at a syntactic level (parsing) and ending with high-level, semantically oriented tasks, such as question answering or information extraction. CRFs have been empirically successful in each of these sequence labeling problems; this section attempts to briefly review some of the main contributions. In order to maintain consistent notation throughout the paper, we will introduce each problem by defining the values that the hidden variables (S) and observation variables (O) can take.

Part of speech tagging
S: Part of speech tags
O: Words in a sentence

CRFs were first applied to predict POS tags for an input word sequence [1] (see Fig. 6); in that work, CRFs were found to outperform both HMMs and MEMMs when evaluated on the Penn Treebank. The CRF error rate when compared with the HMM drops considerably when the model is augmented with a small set of orthographic features: the system used linguistically motivated features, for example, whether the word was capitalized, the presence of particular suffixes (-ed, -ion, etc.). In subsequent work, Sutton *et al.* [11] observe that jointly modeling noun phrase (NP) segmentation and POS tagging using DCRFs improves POS prediction accuracies. POS tagging using CRFs has also been applied to languages such as Chinese [77], Arabic, Czech [78], and Japanese [79], with complex morphological structures, and they have been found to outperform their generative counterparts.

Named Entity Recognition
S: Named entities
O: Words in a sentence

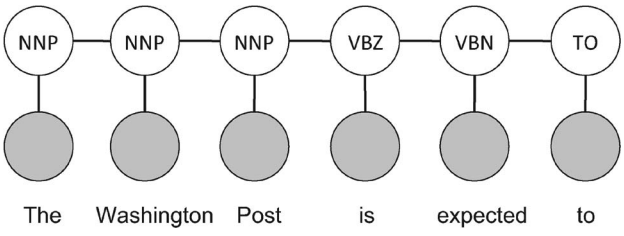


Fig. 6. *Part of speech tags for a fragment of a sample sentence. The tags correspond to the Penn Treebank tag set. For example, NNP corresponds to singular proper noun, VBZ corresponds to verb (present tense, third person singular), etc.*

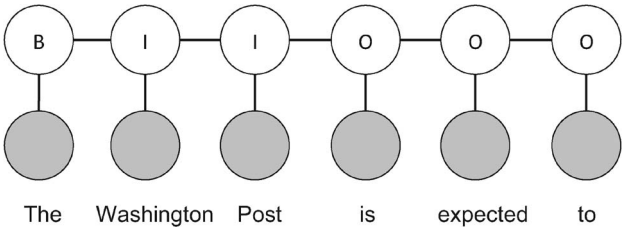


Fig. 7. *NP chunk tags for a fragment of a sample sentence. Here, the NP is "The Washington Post" specified in terms of the B, I, and O labels representing words that begin an entity, are inside an entity, or are outside the entity.*

For a high level task like question answering or information extraction, it would be beneficial to identify all the people, locations, and organizations in a document (or sentence) of interest; this is commonly referred to as named entity recognition (NER). McCallum and Li [80] describe one of the first attempts at applying a CRF to this problem. They use feature induction and select only those feature conjunctions that significantly increase log likelihood; the atomic features are derived using Web-augmented lexicons. Finkel *et al.* [81] report further improvements by modeling nonlocal structure with Gibbs sampling that allow imposing various sorts of long range constraints.

Parsing
S: Grammatical productions (NP chunks, grammar rules)
O: Words in a sentence

Shallow parsing is typically used as a precursor to full parsing or further semantic analysis. NP chunking is an archetypal shallow parsing problem that finds NPs in a sentence. This could be modeled as a sequence tagging task that takes words in a sentence, annotated with POS tags, as input and generates NP chunks in terms of **B**egin, **I**nside, and **O**utside as shown in Fig. 7. Sha and Pereira [13] use a CRF model for this task with the labels being bigrams of the B, I, and O tags to impose a second-order Markov dependency between them. Their CRF model delivers superior F-score performance beating all single-model NP chunking results on the CoNLL-2000 shared task.

A recent approach by Finkel *et al.* [82] presents a general, feature-rich CRF-based parser attaining a state-of-the-art F score on the Penn Treebank parsing task (on sentences of length ≤ 15). Their model uses grammar rules defined by probabilistic context-free grammars, along with additional information such as the span of words encompassed by the rule and its position in the sentence, as labels for their CRF model. This discriminative parser was further used to build a model of nested named entities [83] and a joint model of parsing and NER [84].

Information Extraction
S: Fields to be extracted (e.g., Title, Speaker, etc.)
O: Words in a document

We end this section on CRFs in NLP applications with a reference to the semantically oriented task of information extraction. Peng and McCallum [85] achieve state-of-the-art performance in extracting standard header fields such as title, author, institution, etc., from research papers on a standard benchmark data set. The authors use a number of local features (such as "contains a dot"), layout features (such as "end of a line"), and external lexicon features ("match word in author lexicon") in their linear-chain CRF model. Sutton and McCallum [86] approach this extraction problem differently by using a skip-chain CRF whose structure is a linear chain with additional edges between similar words.

B. Speech Processing: Phone Recognition, Word Recognition, and Other Tasks

As we have discussed throughout the paper, various ASR tasks lend themselves to be modeled as sequence labeling problems; it comes as no surprise that CRFs find wide application across all of these tasks. We discuss some of these applications in the following sections. The description in this section is intentionally brief since the systems described herein have been elaborated upon in previous sections.

Phone or word recognition
S: Phone or word labels
O: Acoustic features representing spectral properties

CRFs were first applied in ASR to the task of phone classification [4] by using feature functions explicitly derived from a baseline HMM system; this was extended by Sung and Jurafsky [62] to phone recognition. Subsequent work has explored the use of alternative feature functions such as Gaussian responsibility scores [37] and phone and phonological feature posteriors [33]. Recently, segmental CRFs have been employed for phone classification and recognition, making use of segmental features and complex state space [38], [65]. Phone-based CRF approaches have shown large improvements over baseline HMM systems on the standard TIMIT data set, in particular, when observation-dependent transition factors are employed. Extending these approaches to word recognition for medium to large vocabulary tasks is complicated by the large increase in the state space; Zweig and Nguyen [20] address this challenge via a segmental CRF by restricting the set of hypotheses to a lattice obtained from a baseline HMM system. An alternative approach [87], [88] uses CRFs to produce phone lattices that are used in subsequent processing steps for word recognition.

Letter-to-sound conversion
S: Graphemes from orthographic transcription
O: Phonemes corresponding to pronunciation

The letter-to-sound conversion task (also known as pronunciation prediction or grapheme-to-phoneme conversion) attempts to predict the pronunciation of a word (in terms of phones) given the spelling of the word (for example, "phoenix" $\Leftrightarrow$ /F IY N IH K S/). Since the orthographic transcription and the phonemic pronunciation are generally of different lengths, systems for this task need to derive alignments between these sequences. Wang and King [89] use a linear-chain CRF with a set of simple indicator functions by examining pairs of graphemes and phonemes, finding a large improvement over a generative baseline system.

Spoken language understanding: concept tagging
S: Semantic concept labels
O: (Possibly errorful) word transcripts of speech

Spoken language understanding attempts to extract the intended meaning of a speaker from speech recognition output. One formalization of this idea is the extraction of task relevant attribute-value pairs (or concept labels) from text; e.g., "I want to fly to Austin on Tuesday" in a flight reservation domain should result in concept pairs *destination: Austin* and *travel.date=Tuesday*. Hahn *et al.* present several systems for concept extraction in multiple languages, and find that CRFs perform quite well in terms of concept error rate across three languages, using a mixture of features based on word bigrams and features of the words themselves (such as prefixes and capitalization) [90].

C. Audio Processing

We end this section on CRF applications by drawing attention to recent work in relatively new areas of audio processing: speech segregation using binary masks and two tasks intended to support music information retrieval.

Speech segregation
S: Binary labeling of time-frequency units
O: Mixture of target and interference acoustic signals

An area of speech processing that lends itself well to the use of CRFs is computational auditory scene analysis (CASA). Within this paradigm, a body of research has been directed toward the estimation of the ideal binary mask (IBM) in order to perform speech segregation. The IBM is a 0/1 mask defined on the time–frequency (T–F) representation of a mixture of two signals (target and interference), which identifies T–F units where the target energy dominates that of the interference and is computed from the premixed signals [91]. In previous work, we have experimented with the use of a 2-D grid structured CRF for speech segregation [55]. The vertices of the graph represent T–F units; each T–F unit is connected by an edge to its neighbor in time and frequency. We observed improved T–F unit classification performance over previous heuristically guided approaches.

Audio-to-score alignment
S: Concurrency labels describing simultaneous notes
O: Acoustic features

Audio-to-score alignment involves aligning an audio recording with its corresponding symbolic score potentially aiding applications in music information retrieval. Joder *et al.* [92] propose the use of CRF-based models for aligning polyphonic music that includes multiple voices or melodies. This alignment problem is formalized as a sequence labeling task; the musical score is represented as a sequence of concurrencies—a set of notes that occur simultaneously. They experiment with increasingly detailed CRF structures that attempt to model concurrency durations and tempo. Their model incorporates several acoustic features characterizing different aspects of musical content, such as harmony and tempo, extracted from local neighborhoods. In experimental evaluations, they observe improved performance in predicting alignments when evaluated on a large-scale database of polyphonic, classical, and popular music.

Music transcription

S: Notes

O: Pitch estimates from the audio signal

CRFs have been employed in automatic music transcription [93] in a postprocessing step as a smoothing technique to reduce single-frame errors. The authors employ a linear-chain CRF for each pitch value that takes as input frame-level pitch estimates, labeling these observations with the notes. The addition of the CRF preprocessing step results in improved performance over several previously reported results on a standard database used for the evaluation of polyphonic music transcription approaches.

VII. CONCLUSION

This paper has explored some of the different properties of CRFs, particularly in relationship to speech, audio, and language technologies. CRFs are in some sense a bridge between HMMs and MLPs, utilizing the Markovian and observational structure of HMMs, while at the same time using a training criterion not unlike that of softmax-based perceptrons. Understanding the relationships of the different components of a CRF (model structure, parameterization, parameter estimation, and inference mechanism) to current technology can not only help bridge the gap between existing technologies but also facilitate innovation in combining different components.

A short paper can only scratch the surface of the topic of CRFs, but the literature cited here provides good further reading on the topic. In particular, the original Lafferty *et al.* paper [1], the review chapter by Sutton and McCallum [3], and the NAACL tutorial by Klein [8] make a good trio of papers for understanding general perspectives on CRFs and their training. More detail on structured discriminative models such as those described in Section IV can be found in [52], which has a particular emphasis on speech recognition. We hope that this paper has provided connections from that literature to the particular perspectives of researchers interested in this technology. ■

Acknowledgment

The authors would like to thank J. Morris, I. Heintz, and C. Brew for discussions of this work.

REFERENCES

[1] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. Int. Conf. Mach. Learn.*, Williamstown, MA, USA, Jun. 2001, pp. 282–289.

[2] D. Yu and L. Deng, "Deep-structured hidden conditional random fields for phonetic recognition," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, Makuhari, Chiba, Japan, Sep. 2010, pp. 2986–2989.

[3] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 93–128.

[4] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification," in *Proc. Eur. Conf. Speech Commun. Technol.*, Lisbon, Portugal, Sep. 2005, pp. 1117–1120.

[5] G. Heigold, H. Ney, P. Lehnen, T. Gass, and R. Schlüter, "Equivalence of generative and log-linear models," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 5, pp. 1138–1148, Jul. 2011.

[6] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[7] A. McCallum, D. Freitag, and F. Pereira, "Maximum entropy Markov models for information extraction and segmentation," in *Proc. Int. Conf. Mach. Learn.*, Stanford, CA, USA, Jun. 2000, pp. 591–598.

[8] D. Klein, "Introduction to classification: Likelihoods, margins, features, and kernels," in *Proc. Human Lang. Technol. Conf. NAACL—Tut.*, Rochester, NY, USA, Apr. 2007.

[9] L. Bottou, "Une approche théorique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Université de Paris XI, Paris, France, 1991.

[10] E. McDermott, "Discriminative training for speech recognition," Ph.D. dissertation, Dept. Comput. Sci. Eng., Waseda Univ., Tokyo, Japan, 1997.

[11] C. Sutton, A. McCallum, and K. Rohanimanesh, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," *J. Mach. Learn. Res.*, vol. 8, pp. 693–723, Mar. 2007.

[12] H. Wallach, "Efficient training of conditional random fields," M.S. thesis, Schl. Cogn. Sci., Univ. Edinburgh, Edinburgh, 2002.

[13] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proc. Human Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, Edmonton, ON, Canada, May 2003, vol. 1, pp. 134–141.

[14] J. Hammersley and P. Clifford, "Markov fields on finite graphs and lattices," 1971. [Online]. Available: http://www.statslab.cam.ac.uk/~grg/books/hammfest/hamm-cliff.pdf

[15] Y. Hifny and S. Renals, "Speech recognition using augmented conditional random fields," *IEEE Trans. Audio Speech Lang. Process.*, vol. 17, no. 2, pp. 354–365, Feb. 2009.

[16] A. L. Berger, S. D. Della Pietra, and V. J. D. Della Pietra, "A maximum entropy approach to natural language processing," *Comput. Linguist.*, vol. 22, no. 1, pp. 39–71, Mar. 1996.

[17] T. Minka, "Algorithms for maxiumum-likelihood logistic regression," Stat. Dept., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. 758, 2001.

[18] R. H. Byrd, P. Lu, J. Nocedal, and C. Y. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 6, pp. 1190–1208, Sep. 1995.

[19] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, San Francisco, CA, USA, Mar. 1993, vol. 1, pp. 586–591.

[20] G. Zweig and P. Nguyen, "A segmental CRF approach to large vocabulary continuous speech recognition," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Merano, Italy, Dec. 2009, pp. 152–157.

[21] X. He, L. Deng, and W. Chou, "Discriminative learning in sequential pattern recognition,"

IEEE Signal Process. Mag., vol. 25, no. 5, pp. 14–36, Sep. 2008.

[22] C.-H. Lee, "Discriminative training—Fundamentals and applications," presented at the Interspeech Tut. Session, Makuhari, Chiba, Japan, Sep. 2010.

[23] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Prague, Czech Republic, Jul. 2002, pp. 1–8.

[24] J. Suzuki, E. McDermott, and H. Isozaki, "Training conditional random fields with multivariate evaluation measures," in *Proc. Annu. Meeting Assoc. Comput. Linguist.*, Sydney, Australia, Jul. 2006, pp. 217–224.

[25] D. Povey and P. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2002, vol. 1, pp. 105–108.

[26] Y. Xiong, J. Zhu, H. Huang, and H. Xu, "Minimum tag error for discriminative training of conditional random fields," *Inf. Sci.*, vol. 179, no. 1–2, pp. 169–179, Jan. 2009.

[27] M. Kim. (2010). Large margin cost-sensitive learning of conditional random fields. *Pattern Recognit.* [Online]. *43(10)*, pp. 3683–3692. Available: http://www.sciencedirect.com/science/article/pii/S0031320310002141

[28] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, E. Brill and K. Church, Eds., Somerset, NJ, USA, May 1996, pp. 133–142.

[29] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" Proc. IEEE, vol. 88, no. 8, pp. 1270–1278, Aug. 2000.

[30] J. Chien and C. Chueh, "Joint acoustic and language modeling for speech recognition," *Speech Commun.*, vol. 52, no. 3, pp. 223–235, Mar. 2010.

[31] M. I. Layton and M. J. F. Gales, "Augmented statistical models for speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Toulouse, France, May 2006, vol. 1, pp. 129–132.

[32] A. Ragni and M. J. F. Gales, "Derivative kernels for noise robust ASR," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2011, pp. 119–124.

[33] J. Morris and E. Fosler-Lussier, "Conditional random fields for integrating local discriminative classifiers," *IEEE Trans. Audio Speech Lang. Process.*, vol. 16, no. 3, pp. 617–628, Mar. 2008.

[34] M. Lehr and I. Shafran, "Learning a discriminative weighted finite-state transducer for speech recognition," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 5, pp. 1360–1367, Jul. 2011.

[35] B. Roark, M. Saraclar, M. Collins, and M. Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm," in *Proc. Annu. Meeting Assoc. Comput. Linguist.*, Barcelona, Spain, Jul. 2004, pp. 47–54.

[36] Y. Kubo, S. Watanabe, and A. Nakamura, "Decoding network optimization using minimum transition error training," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 4197–4200.

[37] Y. H. Abdel-Haleem, "Conditional random fields for continuous speech recognition," Ph.D. dissertation, Dept. Comput. Sci., Univ. Sheffield, Sheffield, U.K., 2006.

[38] G. Zweig, "Classification and recognition with direct segment models," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 4161–4164.

[39] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.

[40] J. Lafferty, X. Zhu, and Y. Liu, "Kernel conditional random fields: Representation and clique selection," in *Proc. Int. Conf. Mach. Learn.*, Banff, AB, Canada, Jul. 2004, pp. 64–71.

[41] J. Peng, L. Bo, and J. Xu, "Conditional neural fields," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2009, pp. 1419–1427.

[42] R. Prabhavalkar and E. Fosler-Lussier, "Backpropagation training for multilayer conditional random field based phone recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Dallas, TX, USA, Mar. 2010, pp. 5534–5537.

[43] Y. Fujii, K. Yamamoto, and S. Nakagawa, "Automatic speech recognition using hidden conditional neural fields," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Prague, Czech Republic, May 2011, pp. 5036–5039.

[44] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Taipei, Taiwan, Apr. 2009, pp. 3761–3764.

[45] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, May 2006.

[46] D. Yu, S. Wang, and L. Deng, "Sequential labeling using deep-structured conditional random fields," *IEEE J. Sel. Top. Signal Process.*, vol. 4, no. 6, pp. 965–973, Dec. 2010.

[47] S. Wiesler and H. Ney, "A convergence analysis of log-linear training," in *Proc. Adv. Neural Inf. Process. Syst.*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., Granada, Spain, Mar. 2011, pp. 657–665.

[48] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer-Verlag, 2001.

[49] D. Johnson, D. Ellis, C. Oei, C. Wooters, P. Faerber, N. Morgan, and K. Asanovic, "ICSI Quicknet software package," 2004. [Online]. Available: http://www.icsi.berkeley.edu/Speech/qn.html

[50] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Sep. 2005.

[51] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2003, pp. 25–32.

[52] M. Gales, S. Watanabe, and E. Fosler-Lussier, "Structured discriminative models for speech recognition," *Signal Process. Mag.*, vol. 29, no. 6, pp. 70–81, Nov. 2012.

[53] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 1st ed. Cambridge, MA, USA: MIT Press, 2009.

[54] K. P. Murphy, "Dynamic Bayesian networks: Representation, inference and learning," Ph.D. dissertation, Comput. Sci. Div., Dept. Electr. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, 2002.

[55] R. Prabhavalkar, Z. Jin, and E. Fosler-Lussier, "Monaural segregation of voiced speech using discriminative random fields," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, Brighton, U.K., Sep. 2009, pp. 856–859.

[56] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[57] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artif. Intell.*, vol. 29, no. 3, pp. 241–288, Sep. 1986.

[58] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. Int. Conf. Uncertainty Artif. Intell.*, Stockholm, Sweden, Jul. 1999, pp. 467–475.

[59] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, Nov. 1999.

[60] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1–2, pp. 1–305, Jan. 2008.

[61] R. Prabhavalkar, E. Fosler-Lussier, and K. Livescu, "A factored conditional random field model for articulatory feature forced transcription," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2011, pp. 77–82.

[62] Y.-H. Sung and D. Jurafsky, "Hidden conditional random fields for phone recognition," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Merano, Italy, Dec. 2009, pp. 107–112.

[63] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc.*, vol. 39, no. 1, pp. 1–38, 1977.

[64] S. Sarawagi and W. W. Cohen, "Semi-Markov conditional random fields for information extraction," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2004, pp. 1185–1192.

[65] Y. He and E. Fosler-Lussier, "Efficient segmental conditional random fields for phone recognition," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, Portland, OR, USA, Sep. 2012, pp. 1898–1901.

[66] P. Liang, "Semi-supervised learning for natural language," M.S. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol. (MIT), Cambridge, MA, USA, 2005.

[67] G. Andrew, "A hybrid Markov/semi-Markov conditional random field for sequence segmentation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sydney, Australia, Jul. 2006, pp. 465–472.

[68] G. Zweig, P. Nguyen, D. Van Compernolle, K. Demuynck, L. Atlas, P. Clark, G. Sell, M. Wang, F. Sha, H. Hermansky, D. Karakos, A. Jansen, S. Thomas, G. S. V. S. Sivaram, S. Bowman, and J. Kao, "Speech recognition with segmental conditional random fields: A summary of the JHU CLSP 2010 summer workshop," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Prague, Czech Republic, May 2011, pp. 5044–5047.

[69] M. Layton, "Augmented statistical models for classifying sequence data," Ph.D. dissertation, Dept. Eng., Cambridge Univ., Cambridge, U.K., 2006.

[70] B. H. Juang and S. Katagiri, "Discriminative learning for minimum error classification,"

IEEE Trans. Signal Process., vol. 40, no. 12, pp. 3043–3054, Dec. 1992.

[71] X. Huang, A. Acero, and H. W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[72] G. Heigold, P. Lehnen, R. Schlüter, and H. Ney, "On the equivalence of Gaussian and log-linear HMMs," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, Brisbane, Australia, Sep. 2008, pp. 273–276.

[73] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer, 1993.

[74] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Jun. 2000, vol. 3, pp. 1635–1638.

[75] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures and Applications*, Fogelman-Soulie and Herault, Eds. New York, NY, USA: Springer-Verlag, 1990, pp. 227–236.

[76] Y. Konig, H. Bourlard, and N. Morgan, "REMAP: Recursive estimation and maximization of a posteriori probabilities—Application to transition-based connectionist speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, USA, Nov. 1995, pp. 388–394.

[77] Y. Shi and M. Wang, "A dual-layer CRFs based joint decoding method for cascaded segmentation and labeling tasks," in *Proc. Int. Joint Conf. Artif. Intell.*, Hyderabad, India, Jan. 2007, pp. 1707–1712.

[78] N. A. Smith, D. A. Smith, and R. W. Tromble, "Context-based morphological disambiguation with random fields," in *Proc. Human Lang. Technol. Conf./Conf. Empirical Methods Natural Lang. Process.*, Vancouver, BC, Canada, Oct. 2005, pp. 475–482.

[79] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying conditional random fields to Japanese morphological analysis," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Barcelona, Catalunya, Spain, Jul. 2004, pp. 230–237.

[80] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proc. Conf. Natural Lang. Learn.*, Edmonton, AB, Canada, May 2003, pp. 188–191.

[81] J. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," in *Proc. Annu. Meeting Assoc. Comput. Linguist.*, Ann Arbor, MI, USA, Jun. 2005, pp. 363–370.

[82] J. Finkel, A. Kleeman, and C. Manning, "Efficient, feature-based, conditional random field parsing," in *Proc. Annu. Meeting Assoc. Comput. Linguist., Human Lang. Technol.*, Columbus, OH, USA, Jun. 2008, pp. 959–967.

[83] J. Finkel and C. Manning, "Nested named entity recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Suntec, Singapore, Aug. 2009, pp. 141–150.

[84] J. Finkel and C. Manning, "Joint parsing and named entity recognition," in *Proc. Human Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, Boulder, CO, USA, May 2009, pp. 326–334.

[85] F. Peng and A. McCallum, "Accurate information extraction from research papers using conditional random fields," in *Proc. Human Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguist.*, Boston, MA, USA, May 2004, pp. 329–336.

[86] C. Sutton and A. McCallum, "Collective segmentation and labeling of distant entities in information extraction," Univ. Massachusetts, Amherst, MA, USA, Tech. Rep., 2004.

[87] J. Morris and E. Fosler-Lussier, "Crandem: Conditional random fields for word recognition," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, Brighton, U.K., Sep. 2009, pp. 3063–3066.

[88] J. J. Morris, "A study on the use of conditional random fields for automatic speech recognition," Ph.D. dissertation, Dept. Comput. Sci. Eng., The Ohio State Univ., Columbus, OH, USA, 2010.

[89] D. Wang and S. King, "Letter-to-sound pronunciation prediction using conditional random fields," *IEEE Signal Process. Lett.*, vol. 18, no. 2, pp. 122–125, Feb. 2011.

[90] S. Hahn, M. Dinarelli, C. Raymond, F. Lefevre, P. Lehnen, R. de Mori, A. Moschitti, H. Ney, and G. Riccardi, "Comparing stochastic approaches to spoken language understanding in multiple languages," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 6, pp. 1569–1583, Aug. 2011.

[91] D. Wang, "On ideal binary mask as the computational goal of auditory scene analysis," in *Speech Separation by Humans and Machines*, P. Divenyi, Ed. New York, NY, USA: Springer-Verlag, 2005, pp. 181–197.

[92] C. Joder, S. Essid, and G. Richard, "A conditional random field framework for robust and scalable audio-to-score matching," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 8, pp. 2385–2397, Nov. 2011.

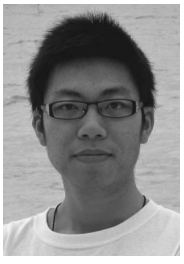
[93] E. Benetos and S. Dixon, "Joint multi-pitch detection using harmonic envelope estimation for polyphonic music transcription," *IEEE J. Sel. Top. Signal Process.*, vol. 5, no. 6, pp. 1111–1123, Oct. 2011.

ABOUT THE AUTHORS

Eric Fosler-Lussier (Senior Member, IEEE) received the B.A.S degree in computer and cognitive studies and the B.A. degree in linguistics from the University of Pennsylvania, State College, PA, USA, in 1993 and the Ph.D. degree from the University of California Berkeley, Berkeley, CA, USA, in 1999. His Ph.D. research was conducted at the International Computer Science Institute.

As an Associate Professor with the Department of Computer Science and Engineering (and Linguistics by courtesy) at The Ohio State University, Columbus, OH, USA, he directs the Speech and Language Technologies (SLaTe) Laboratory.

Dr. Fosler-Lussier currently serves on the IEEE Speech and Language Technical Committee, and received the 2010 Signal Processing Society Best Paper Award.



Yanzhang He received the B.E. degree in software engineering from Beihang University, Beijing, China, in 2008. He is currently working toward the Ph.D. degree at the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA.

His current research interests include conditional random fields and discriminative learning for automatic speech recognition and keyword spotting.

Preethi Jyothi received the B.Tech degree from the National Institute of Technology, Calicut, India. in 2006. She is currently working toward the Ph.D. degree at the Speech and Language Technologies (SLaTe) Laboratory, Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA.

Her research interests include automatic speech recognition (ASR), discriminative models, and speech production-oriented models for ASR.

Miss Jyothi received the Best Student Paper Award at Interspeech 2012.

Rohit Prabhavalkar (Graduate Student Member, IEEE) received the B.E. degree in computer engineering from the University of Pune, Maharashtra, India, in 2007 and the M.S. degree in computer science and engineering from The Ohio State University, Columbus, OH, USA, in 2012, where he is currently working toward the Ph.D. degree in the Speech and Language Technologies (SLaTe) Laboratory, Department of Computer Science and Engineering.

His research interests include acoustic and pronunciation modeling for automatic speech recognition, natural language processing, and machine learning.