Triangular-Chain Conditional Random Fields

Minwoo Jeong and Gary Geunbae Lee, *Member, IEEE*

***Abstract*—Sequential modeling is a fundamental task in scientific fields, especially in speech and natural language processing, where many problems of sequential data can be cast as a sequential labeling or a sequence classification. In many applications, the two problems are often correlated, for example named entity recognition and dialog act classification for spoken language understanding. This paper presents triangular-chain conditional random fields (CRFs), a unified probabilistic model combining two related problems. Triangular-chain CRFs jointly represent the sequence and meta-sequence labels in a single graphical structure that both explicitly encodes their dependencies and preserves uncertainty between them. An efficient inference and parameter estimation method is described for triangular-chain CRFs by extending linear-chain CRFs. This method outperforms baseline models on synthetic data and real-world dialog data for spoken language understanding.**

***Index Terms*—Conditional random fields (CRFs), probabilistic sequence modeling, spoken language understanding, triangular-chain structure.**

I. Introduction

PROBABILISTIC sequence modeling is fundamental to many statistical approaches in speech and natural language processing. Speech processing often deals with sequential data arising through measurement of time series, for example the acoustic features at successive time frames used for speech recognition. Natural language processing can also involve sequential data, for example the sequence of words in an English sentence. A basic assumption here is that the sequences exhibit a significant sequential correlation [1].

The problem of modeling sequential data has been extensively studied in the last two decades by both the speech and language communities. Statistical classification attempts can be mapped into two areas: sequential labeling and sequence classification. Given a sequence of observations, the former predicts "a sequence of labels" while the latter predicts "a single label." More concretely, we define the two problems as follows.

1) *Sequential labeling* is a problem of predicting a sequence of labels $\mathbf{y}$ given an input sequence $\mathbf{x}$. One typically assumes that an output y_t at time t corresponds to an input $\mathbf{x}_t$ at time t. Probabilistic models wish to learn $p(\mathbf{y}|\mathbf{x})$ given sequence data $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})_{n=1}^{N}$. Many tasks are forms of this problem, such as part-of-speech tagging [2], shallow parsing [3], named entity recognition [4], and speech recognition [5].
2) *Sequence classification* is a problem of predicting a single label z that applies to an entire input sequence $\mathbf{x}$. Probabilistic models wish to find $p(z|\mathbf{x})$ given data $(\mathbf{x}^{(n)}, z^{(n)})_{n=1}^{N}$. Examples include dialog act classification [6], gesture recognition [7], phone classification [8], and speaker verification and identification [9]. Some applications (e.g., document classification [2]) simply ignore the sequential aspects of the input sequence.

Although each problem is applied separately to sequences, some applications involve both problems. For example, the dialog manager in the slot-filling spoken dialog system [10] requests information consisting of object names and user's intent. The problem of recognizing an object is sequential labeling, and the problem of identifying user's intent is sequence classification. A cascaded-approach language understanding component examines a user's utterances (a sequence of words, $\mathbf{x}$) to first predict particular objects' names (a sequence of labels $\mathbf{y}$) and to then infer the user's intent (a label z).

The key idea in this paper is to concurrently solve the two related problems, that is, to jointly predict best labels $(\hat{\mathbf{y}}, \hat{z})$ using a probabilistic model $p(\mathbf{y}, z|\mathbf{x})$ that is trained from the joint data $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}, z^{(n)})_{n=1}^{N}$. Our framework regards $\mathbf{y}$ as a *sequence* and z as a *topic* (meta-sequence information), so the problem can be seen as a multitopic sequence labeling. Our goal is to learn dependencies between sequences and their corresponding topics, which provide relevant information to improve the performance. Intuitively, co-occurrence patterns between $\mathbf{y}$ and z are important because they can be exploited to improve the prediction accuracies of both problems. This intuition is applied to probabilistic sequence modeling to develop a novel method, triangular-chain conditional random fields (CRFs).

Our framework is a particular type of joint modeling approach that yields better performance than the cascaded approach for applications consisting of several subtasks. Jointly modeling the subtasks is an appealing method for natural language processing because most enterprise applications comprise a set of correlated components. Recently, many studies have addressed this issue to obtain promising results in various text and spoken language problems such as part-of-speech tagging and noun–phrase chunking [4], parsing and information extraction [11], semantic role labeling [12], and speech recognition [13]. However, our method is distinct from prior works since we seek a unified model combining sequence labeling and sequence classification. Our previous work [14] introduced the triangular-chain CRFs with application to joint prediction of dialog

Manuscript received June 29, 2007; revised April 11, 2008. Published August 13, 2008 (projected). This work was supported by the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Knowledge Economics of Korea, and also funded by the second BK21 program by the Ministry of Education of Korea. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Bill Byrne.

The authors are with Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang 790-784, Korea (e-mail: stardust@postech.ac.kr; gblee@postech.ac.kr)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2008.925143

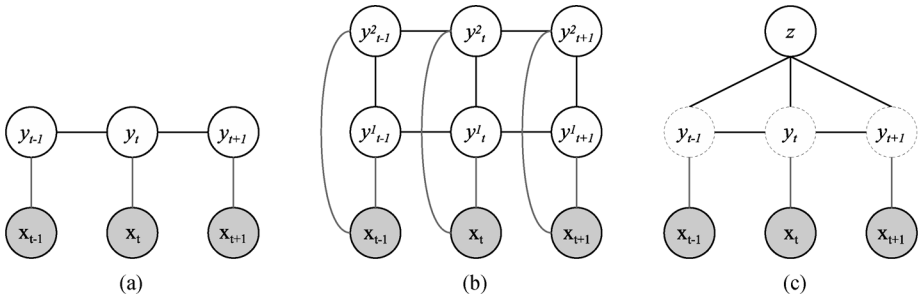


Fig. 1. Family of CRFs: An open node denotes a random variable, and a shaded node has been set to its observed value. In (c), a dotted node indicates a latent variable that cannot be observed in training data. (a) Linear-chain CRFs. (b) Factorial CRFs. (c) Hidden CRFs.

act and named entity. Triangular-chain structured models can solve the two problems by jointly representing the subtasks in a single graphical model that both explicitly represents their dependencies and preserves the uncertainty between them. This work in the current paper gives several new results. It describes two versions of triangular-chain CRFs and relates them to other works. Moreover, it introduces entirely new methods of reducing the time complexity: parameter initialization and inference with partial space constraint. New experiments are also reported, including synthetic data and spoken dialog data. Finally, the algorithms were evaluated extensively for spoken language understanding tasks and compared to various baseline systems.

The remainder of this paper is structured as follows. Section II reviews the CRFs closely related to our work. Next, Section III presents triangular-chain CRFs, a novel method for learning a probability distribution $p(\mathbf{y}, z|\mathbf{x})$. First, the triangular-chain structures are described (Section III-B), and then the inference (Section III-C) and parameter estimation (Section III-D) are explained. Section IV shows the experimental results for synthetic data to assess the method's potential usefulness. Section V applies the triangular-chain CRFs to spoken language understanding problems, where sequential labeling and sequence classification can be correlated. The empirical results show an improvement in the domains of both dialog act classification and named entity recognition. Finally, Section VI gives our conclusions.

II. Review of CRFs

CRFs are undirected graphical models used to specify the conditional probability of an assignment of output labels given a set of input observations [15]. Recently, CRFs have been applied to many scientific fields, including speech and language processing. This section reviews three types of CRFs which are closely related to our work: linear-chain, factorial, and hidden CRFs.

A. Linear-Chain CRFs

Linear-chain CRFs, shown in Fig. 1(a), are conditional probability distributions over label sequences which are conditioned on the input sequences [15], [16]. Linear-chain CRFs typically assume that the dependencies are encoded in a left-to-right chain structure. Formally, linear-chain CRFs are defined as follows. Let $\mathbf{x}$ be a random vector that is observed,[1] and $\mathbf{y}$ be a random vector that we wish to predict. For example, in the part-of-speech tagging task, $\mathbf{x}$ may be a sequence of words and $\mathbf{y}$ may be a sequence of part-of-speech tags. The model is typically simplified by assuming that the lengths of the two vectors are equal, and then a paired sequence $(\mathbf{x}, \mathbf{y})$ denotes $(\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, \ldots, \mathbf{x}_T, y_T)$. For any scalar variable x and y, we write that $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, where $\mathcal{X}$ is the input space, and $\mathcal{Y}$ is the finite set of labels. Then, a first-order linear-chain CRF defines the conditional probability for a label sequence $\mathbf{y}$ to be

$$p_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \phi_t(y_t, y_{t-1}, \mathbf{x}) \tag{1}$$

where ϕ_t is the local potential, which denotes the factor at time t. $Z(\mathbf{x})$ is the partition function that ensures the probability of all state sequences sum to one; that is, $Z(\mathbf{x})$ is defined as $Z(\mathbf{x}) \triangleq \sum_{\mathbf{y}} \prod_{t=1}^{T} \phi_t(y_t, y_{t-1}, \mathbf{x})$. Note that $\phi_t(y_t, y_{t-1}, \mathbf{x}) = \phi_t(y_t, \mathbf{x})$ if $t = 1$.

We assume that the potentials factorize according to a set of features $\{f_k\}$, as

$$\phi_t(y_t, y_{t-1}, \mathbf{x}) = \underbrace{\phi^1_t(y_t, \mathbf{x})}_{\text{observation}} \cdot \underbrace{\phi^2_t(y_t, y_{t-1})}_{\text{transition}} \tag{2}$$

$$\phi^1_t(y_t, \mathbf{x}) = \exp\left(\sum_k \lambda^1_k f^1_k(y_t, \mathbf{x}_t)\right) \tag{3}$$

$$\phi^2_t(y_t, y_{t-1}) = \exp\left(\sum_k \lambda^2_k f^2_k(y_t, y_{t-1})\right) \tag{4}$$

where the feature functions can encode any aspect of the observation, $f^1_k(y_t, \mathbf{x}_t)$, and a state transition, $f^2_k(y_t, y_{t-1})$, centered at the current time, t.[2] Large positive values for λ^1_k and λ^2_k indi-

[1] More concretely, $\mathbf{x}$ is a sequence of input feature vectors, and $\mathbf{x}_t$ denotes an input feature vector at time t, that is, $\mathbf{x}_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,M})$, where M is the size of the dimension for the input feature vector. In many natural language tasks, input vectors may include rich and largely unconstrained features (e.g., words, prefixes and suffixes, capitalization, n-gram, part-of-speech tags, phrase chunk labels, syntactic parse trees, membership in domain-specific lexicon, and so on); hence, $\mathbf{x}_t$ is likely to be a high-dimensional sparse vector.

[2] Although CRFs allow the score of a transition $y_{t-1} \rightarrow y_t$ to depend on the current observation vector as $f^2_k(y_t, y_{t-1}, \mathbf{x}_t)$, for clarity we have shown that the transition receives the same score as $f^2_k(y_t, y_{t-1})$, regardless of the input. This model is defined as a hidden Markov model (HMM)-like linear-chain CRF in [16]. This paper keeps the same factorization throughout the paper, but our formulas can be extended to allow the transition to depend on the observations with minor changes.

cate a preference for such an event, while large negative values make the event unlikely.

There are two common inference problems for linear-chain CRFs, both of which have been efficiently solved via dynamic programming methods. First, the marginal distributions can be calculated and $Z(\mathbf{x})$ computed during training by the forward–backward algorithm [17]. Second, the most likely labeling $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x})$ can be computed during prediction by the Viterbi algorithm [18]. Moreover, parameter estimation for linear-chain CRFs is typically accomplished by maximizing the conditional log-likelihood; it is known to be a convex function, which guarantees convergence to the global optimum [15], [16].

When the dependencies are excluded between y_t and y_{t-1}, this special case is identical to a logistic regression model (known as a maximum entropy classifier in the natural language community [19]). A logistic regression model can therefore be viewed as an unstructured version of CRFs, in which state transitions are ignored and the probability of sequence $p_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x})$ is a product of per-state probability $p_{\boldsymbol{\lambda}}(y_t|\mathbf{x}_t)$ as follows:

$$p_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} \underbrace{\frac{1}{Z(\mathbf{x}_t)} \exp\left(\sum_k \lambda_k f_k(y_t, \mathbf{x}_t)\right)}_{p_{\boldsymbol{\lambda}}(y_t|\mathbf{x}_t)} \tag{5}$$

where $Z(\mathbf{x}_t)$ is a per-state partition function, $Z(\mathbf{x}_t) \triangleq \sum_{y_t} \exp\left(\sum_k \lambda_k f_k(y_t, \mathbf{x}_t)\right)$. To give a unified view to readers, we call the probability distribution $p_{\boldsymbol{\lambda}}(y_t|\mathbf{x}_t)$ the 0-order CRF. Note that 0-order CRFs can be applied to the previously mentioned sequence classification problems.

Linear-chain CRFs have been applied to obtain promising results in various natural language tasks including part-of-speech tagging [15], shallow parsing [3], information extraction [20], biomedical named entity recognition [21], and spoken language understanding [22].

B. Factorial CRFs

Factorial CRFs are conditional distributions which have multiple layers of linear-chains with connections between co-temporal labels [4], as shown in Fig. 1(b). Similarly to linear-chain CRFs, factorial CRFs are defined as follows. Let $\mathbf{x}$ be a random vector that is observed, and $\mathbf{y}$ be a set of layers, $\mathbf{y} = \{\mathbf{y}^l\}$, where $\mathbf{y}^l$ is a random vector of the lth chain for $l = 1, \ldots, L$. For example, in joint part-of-speech tagging and noun–phrase chunking [4], $\mathbf{y}^1$ may be part-of-speech tags and $\mathbf{y}^2$ may be noun phrase chunks given an input word sequence $\mathbf{x}$. Therefore, we can write the factorial sequences as $(\mathbf{x}_1, y_1^{1:L}, \mathbf{x}_2, y_2^{1:L}, \ldots, \mathbf{x}_T, y_T^{1:L})$, where $y_t^{1:L} = \{y_t^1, \ldots, y_t^L\}$. Then, a factorial CRF is defined as follows:

$$p_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \prod_{l=1}^{L} \phi_t(y_t^l, y_{t-1}^l, y_t^{l-1}, \mathbf{x}) \tag{6}$$

where $Z(\mathbf{x})$ is the normalization factor over all state sequences, $Z(\mathbf{x}) \triangleq \sum_{\mathbf{y}} \prod_{t=1}^{T} \prod_{l=1}^{L} \phi_t(y_t^l, y_{t-1}^l, y_t^{l-1}, \mathbf{x})$. Note that $\phi_t(y_t^l, y_{t-1}^l, y_t^{l-1}, \mathbf{x}) = \phi_t(y_t^l, y_{t-1}^l, \mathbf{x})$, if $l = 1$. To factorize a cross-product space, local factor functions ϕ_t are divided by between-chain edges and within-chain edges

$$\phi_t(y_t^l, y_{t-1}^l, y_t^{l-1}, \mathbf{x}) = \underbrace{\phi_t^1(y_t^l, \mathbf{x})}_{\text{observation}} \cdot \underbrace{\phi_t^2(y_t^l, y_{t-1}^l)}_{\text{within-chain}} \cdot \underbrace{\phi_t^3(y_t^l, y_t^{l-1})}_{\text{between-chain}} \tag{7}$$

$$\phi_t^1(y_t^l, \mathbf{x}) = \exp\left(\sum_k \lambda_k^1 f_k^1(y_t^l, \mathbf{x}_t)\right) \tag{8}$$

$$\phi_t^2(y_t^l, y_{t-1}^l) = \exp\left(\sum_k \lambda_k^2 f_k^2(y_t^l, y_{t-1}^l)\right) \tag{9}$$

$$\phi_t^3(y_t^l, y_t^{l-1}) = \exp\left(\sum_k \lambda_k^3 f_k^3(y_t^l, y_t^{l-1})\right). \tag{10}$$

Factorial and linear-chain CRFs are special cases of *dynamic* CRFs, which factorize according to an undirected graphical model whose structure and parameters are repeated over a sequence [4]. Although dynamic CRFs enlarge their objective to arbitrary structures, for practicality a special case such as the linear-chain or factorial structures is widely adopted for natural language fields. Learning factorial CRFs is, like learning linear-chain CRFs, based on convex optimization of the likelihood function. Unfortunately, exact inference and parameter estimations for factorial CRFs are generally intractable, and are too expensive even for dual-layered factorial CRFs. To alleviate this problem, approximate methods such as belief propagation using dynamic scheduling and cascaded training have been proposed [4].

Factorial CRFs have been applied to multiple sequence labeling tasks. [4] has shown an improvement in the domain of joint part-of-speech tagging and noun–phrase chunking and in named entity recognition.

C. Hidden CRFs

Hidden CRFs, as shown in Fig. 1(c), are conditional distributions over a class label and a set of hidden states conditioned on a set of input observations, with dependencies between hidden variables [23]. They were originally defined not only for chains, but also for arbitrary structures such as a minimum spanning tree for object recognition [23]. This paper, for brevity, assumes that a hidden structure of $\mathbf{y}$ is a linear-chain. Formally, the hidden CRFs are defined as follows. Let $\mathbf{x}$ be a random vector that is observed, $\mathbf{y}$ be a random vector that is "unobserved" and represents hidden underlying structures, and z be a random variable that we wish to predict. Similarly, we write that $z \in \mathcal{Z}$, where $\mathcal{Z}$ is a finite set. For example, in the gesture recognition task [23], $\mathbf{x}$ may be a sequence of video, $\mathbf{y}$ may be a part (or subgesture) representation of a scene, and z may be a label for a gesture. Also, we can write the sequences as $(z, \mathbf{x}_1, y_1, \mathbf{x}_2, y_2, \ldots, \mathbf{x}_T, y_T)$, where $\mathbf{y}$ is a hidden sequence. Then a hidden CRF is defined as follows:

$$p_{\boldsymbol{\lambda}}(z|\mathbf{x}) = \sum_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y}, z|\mathbf{x}) = \frac{Z(z, \mathbf{x})}{Z(\mathbf{x})} \tag{11}$$

where $Z(z, \mathbf{x})$ is the partition function over the hidden state $\mathbf{y}$, $Z(z, \mathbf{x}) \triangleq \sum_{\mathbf{y}} \prod_{t=1}^{T} \phi_t(z, y_t, y_{t-1}, \mathbf{x})$, and $Z(\mathbf{x}) = \sum_z Z(z, \mathbf{x})$.

We assume that the potentials factorize according to a set of features $\{f_k\}$, as

$$\phi_t(z, y_t, y_{t-1}, \mathbf{x}) = \underbrace{\phi_t^1(y_t, \mathbf{x})}_{\text{observation}} \cdot \underbrace{\phi_t^2(z, y_t, y_{t-1})}_{\text{hidden-state}} \tag{12}$$

$$\phi_t^1(y_t, \mathbf{x}) = \exp\left(\sum_k \lambda_k^1 f_k^1(y_t, \mathbf{x}_t)\right) \tag{13}$$

$$\phi_t^2(z, y_t, y_{t-1}) = \exp\left(\sum_k \lambda_k^2 f_k^2(z, y_t, y_{t-1})\right). \tag{14}$$

Note that an output label z is independent of $\mathbf{x}$ given $\mathbf{y}$, i.e., hidden CRFs aim to predict a label z with an abstract representation $\mathbf{y}$ of noisy observations $\mathbf{x}$. Hence, in hidden CRFs, the space $\mathcal{Y}$ is typically limited as $|\mathcal{Y}| < |\mathcal{X}|$. The key difference between hidden CRFs and the two previous models is that the hidden CRFs predict a label z but not a sequence $\mathbf{y}$. Thus, they belong to the sequence classification problem defined in Section I.

Since the conditional log-likelihood functions of hidden CRFs are no longer convex, the training algorithm is not guaranteed to reach the global maximum. Despite this disadvantage, hidden CRFs have succeeded in computer vision and speech recognition fields such as object recognition, gesture recognition [23], and phone classification [8].

III. Triangular-Chain CRFs

A. Motivation

The most basic assumption used in classical supervised learning is that data are drawn from independent and identical distributions (i.i.d). Unfortunately, in many applications, this assumption is inaccurate. For example, text and speech data can be drawn from different speakers, genres, styles, or topics (to reduce confusion, we call this meta-data information *topic*). The problem of modeling multitopic sequences is a primary motivation of this paper. Formally, we design the following classification problem. Let $\mathcal{D} = (\mathbf{x}^{(n)}, y^{(n)})_{n=1}^{N}$ be a set of N training examples. We assume that the data can be grouped by z, and each single group of data is drawn from i.i.d. By adding z, we extend the data representation to $\mathcal{D} = (\mathbf{x}^{(n)}, y^{(n)}, z^{(n)})_{n=1}^{N}$. Now, the goal is to find a probability distribution $p(y, z|\mathbf{x})$ to correctly predict a label y with a topic label z.

If we extend an output label y to a sequence $\mathbf{y}$, then the cross-product space is composed as $((y_1, z_1), (y_2, z_2), \ldots, (y_T, z_T))$, which is similar to the form of factorial CRFs. However, the cross-product space is often very large, especially for sequence data, where this approach is infeasible. We assume that a single topic z depends on the entire sequence $(\mathbf{x}, \mathbf{y})$. Therefore, the vector representations $\mathbf{z} = \{z_1, z_2, \ldots, z_T\}$ are redundant, and we only consider the single topic variable z. This representation naturally leads us to a new graphical structure, *triangular-chain*. The next section presents an instantiation of this framework to CRFs.

B. Triangular-Chain Structure Model

A probability distribution $p(\mathbf{y}, z|\mathbf{x})$ can be learned through a novel structured model, triangular-chain CRFs. Triangular-chain CRFs are defined as follows. Let a random vector $\mathbf{x}$ be a sequence of input observations, a random vector $\mathbf{y}$ be a sequence of output labels that we wish to predict, and a random variable z be an output variable that indicates a topic or meta-sequence information. For example, in spoken language understanding task [10], $\mathbf{x}$ may be a word sequence, $\mathbf{y}$ may be a named entity, and z may be a dialog act (or speaker's intent). We can write the data as $(\mathbf{x}, \mathbf{y}, z) = (\mathbf{x}_0, z, \mathbf{x}_1, y_1, \mathbf{x}_2, y_2, \ldots, \mathbf{x}_T, y_T)$, where $\mathbf{x}_0$ indicates an observation feature vector for topic z. Using this additional notation, we can integrate other knowledge into the model, for instance, a history of dialog acts and of system actions and user profiles available in the dialog manager. Then, we formally define the triangular-chain CRFs as follows.

Definition 1: Let $\mathbf{y}$ *and* $\mathbf{x}$ *be random vectors,* z *be a random variable,* $\boldsymbol{\lambda}$ *be a parameter vector, and* $\{f_k\}$ *be a set of real-valued feature functions. Then, a* **triangular-chain conditional random field** *is a conditional probability distribution* $p_{\boldsymbol{\lambda}}(\mathbf{y}, z|\mathbf{x})$ *that takes the form of*

$$p_{\boldsymbol{\lambda}}(\mathbf{y}, z|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \left(\phi_t(z, y_t, y_{t-1}, \mathbf{x})\right) \cdot \varphi(z, \mathbf{x}) \tag{15}$$

where ϕ_t *and* φ *are the potentials over triangular-chain graphs and the partition function* $Z(\mathbf{x})$ *is defined to ensure that the distribution is normalized as* $Z(\mathbf{x}) \triangleq \sum_{\mathbf{y},z} \prod_{t=1}^{T} \left(\phi_t(z, y_t, y_{t-1}, \mathbf{x})\right) \cdot \varphi(z, \mathbf{x})$, *and where* f_k *and* $\boldsymbol{\lambda}$ *parameterize the potentials* ϕ_t *and* φ.

Triangular-chain CRFs have local factors that are divided by time-dependent potential ϕ_t and time-independent potential φ; thus, ϕ_t applies to a state of sequence at time t and φ applies to the entire sequence. Accordingly, a triangular-chain structure is constructed with these two factors. Obviously, φ plays a prior role in classifying z. If we assume that φ is uniform, then z depends on only $\mathbf{x}_{1:T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ and $\mathbf{y}_{1:T} = \{y_1, \ldots, y_T\}$.

Fig. 2(a) shows a graphical model for the triangular-chain CRF. In this figure, the triangular-chain CRF can be seen as an integration of a linear-chain CRF and a 0-order CRF. Thus, variables and potentials can naturally derive from the two CRFs: 1) from 0-order CRFs, $\mathbf{x}_0$ is an observation and φ is a potential function for predicting z, and 2) from linear-chain CRFs, $\{\mathbf{x}_t\}$ for $t = 1, \ldots, T$ are observations and ϕ_t are potential functions for predicting $\mathbf{y}$. We combine a linear-chain CRF and a 0-order CRF by adding probabilistic connections between $\mathbf{y}$ and z, i.e., adding edges between them in an undirected graphical model. In addition, the links between z and $\{\mathbf{x}_t\}$ are appended to represent the relations of topic and sequence; that is, the observation of sequence data is assumed to depend on topic. Furthermore, if we can only obtain partially labeled data, that is, if the sequence $\mathbf{y}$ is not observed, then triangular-chain CRFs become hidden CRFs whose underlying hidden structure is a linear-chain. More work on learning with hidden variables will be discussed in Section III-E.

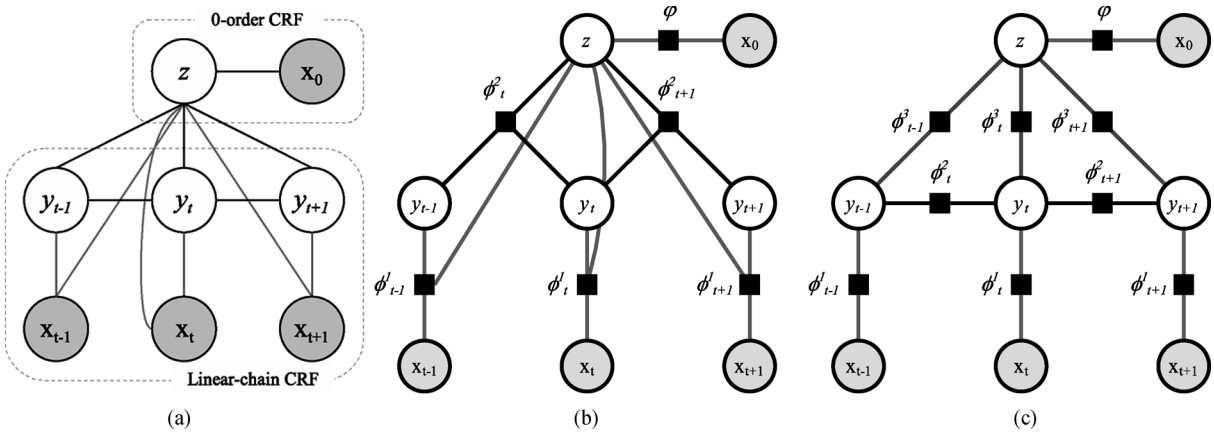


Fig. 2. Graphical representations for triangular-chain CRFs. In factor graphs (b) and (c), a solid square denotes a local factor to be used for inference. (a) Triangular-chain CRFs. (b) MODEL1. (c) MODEL2.

Now, we describe two configurations for triangular-chain CRFs.

MODEL1: We assume that a random variable z directly influences random vectors $\mathbf{x}$ and $\mathbf{y}$. Then, we factorize ϕ_t and φ as follows:

$$\phi_t(z, y_t, y_{t-1}, \mathbf{x}) = \underbrace{\phi_t^1(z, y_t, \mathbf{x})}_{\text{observation}} \cdot \underbrace{\phi_t^2(z, y_t, y_{t-1})}_{\text{transition}} \tag{16}$$

$$\underbrace{\varphi(z, \mathbf{x})}_{prior} = \exp\left(\sum_k \lambda_k^0 f_k^0(z, \mathbf{x}_0)\right) \tag{17}$$

$$\phi_t^1(z, y_t, \mathbf{x}) = \exp\left(\sum_k \lambda_k^1 f_k^1(z, y_t, \mathbf{x}_t)\right) \tag{18}$$

$$\phi_t^2(z, y_t, y_{t-1}) = \exp\left(\sum_k \lambda_k^2 f_k^2(z, y_t, y_{t-1})\right). \tag{19}$$

We name this a *hard constraint* model (MODEL1). The feature functions encode any aspect of the observation and transition for the sequence $f_k^1(z, y_t, \mathbf{x}_t)$ and $f_k^2(z, y_t, y_{t-1})$, centered at the current time t and the observation for topic $f_k^0(z, \mathbf{x}_0)$. Note that both $f_k^1(z, y_t, \mathbf{x}_t)$ and $f_k^2(z, y_t, y_{t-1})$ represent the z-dependent transitions and observations, so an assignment of sequence labels is highly dependent on topic z. Fig. 2(b) shows the factor graph representation [24] of MODEL1 for visualizing the model's factor and its structures.

MODEL2: Moreover, we can derive a special case of MODEL1 by setting $\phi_t^1(z, y_t, \mathbf{x}) = \phi_t^1(y_t, \mathbf{x})$. This factorization is reasonable for some applications, for example, topic-dependent speech recognition. The topic z (e.g., speaker, genre or dialog act) would influence the language model as $\phi_t^2(z, y_t, y_{t-1})$, but the emission of the speech $\mathbf{x}$ would depend only on the words $\mathbf{y}$, not the topic, as in $\phi_t^1(y_t, \mathbf{x})$. Furthermore, we can derive a more factorized model by assuming that the state transition $y_{t-1} \rightarrow y_t$ is not dependent of the topic z.

We assume that a random variable z does not directly influence random vectors $\mathbf{x}$. For clarity, we assume that z is independent of $\mathbf{x}_{1:T}$ given $\mathbf{y}_{1:T}$. Then, we factorize ϕ_t and φ as follows:

$$\phi_t(z, y_t, y_{t-1}, \mathbf{x}) = \underbrace{\phi_t^1(y_t, \mathbf{x})}_{\text{observation}} \cdot \underbrace{\phi_t^2(y_t, y_{t-1})}_{\text{transition}} \cdot \underbrace{\phi_t^3(z, y_t)}_{z-\mathbf{y}\text{ edge}} \tag{20}$$

$$\phi_t^3(z, y_t) = \exp\left(\sum_k \lambda_k^3 f_k^3(z, y_t)\right). \tag{21}$$

We name this a *soft constraint* model (MODEL2). This model has the same factors ϕ_t^1 and ϕ_t^2 as linear-chain CRFs ((3) and (4)) and has the same factor φ as MODEL1 ((17)). ϕ_t^3 denotes a soft constraint of interdependency between z and $\mathbf{y}$. We assume that state transition is the same for all values of z; i.e, z operates as an observation feature to predict a sequence of labels. Fig. 2(c) shows the factor graph representation of MODEL2.

This model should have a smaller parameter size than MODEL1 because MODEL2 requires only $|\mathcal{Y}| \times |\mathcal{Z}|$ additional memory to store dependencies between $\mathbf{y}$ and z. Furthermore, this fact leads to a reduction of the time complexity in large-scale applications. Theoretically, the time complexities of inference in MODEL1 and MODEL2 are the same. In practice, especially with large-scale data, MODEL2 can more efficiently calculate the potentials than MODEL1 since local factors for the observations and transitions, ϕ_t^1 and ϕ_t^2, are calculated a factor of z more times in MODEL1.

C. Inference

There are two common inference problems for triangular-chain CRFs: (1) computing the marginal distributions for each factor $p_{\boldsymbol{\lambda}}(z, y_t|\mathbf{x})$, $p_{\boldsymbol{\lambda}}(z, y_t, y_{t-1}|\mathbf{x})$, and $p_{\boldsymbol{\lambda}}(z|\mathbf{x})$ for the gradient calculation and computing a partition function $Z(\mathbf{x})$, and (2) computing the Viterbi decoding to predict a new instance. Intuitively, the former is used to estimate the parameter vector

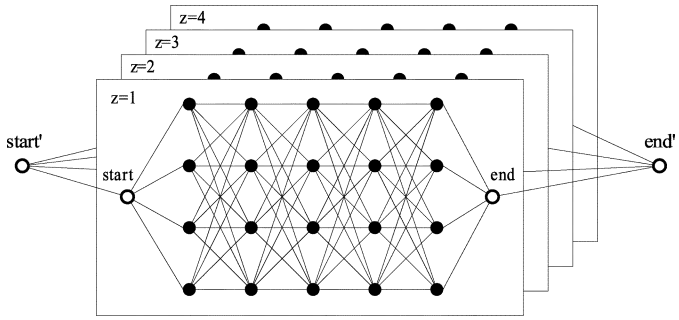


Fig. 3. Search space in a triangular-chain CRF. Each plane corresponds to one linear-chain CRF which has start and end nodes. All planes are parallel but are connected by start′ and end′ nodes.

in training, and the latter is executed for evaluating testing data. We first explain the Viterbi decoding.

Triangular-chain CRFs form parallel linear-chain structures so that there are $|\mathcal{Z}|$ planes of linear-chain search space and one plane corresponding to a realization of one z. Fig. 3 describes the search space in triangular-chain CRFs, where a 3-D space is explored rather than a 2-D one. Note that no path connects to nodes on other planes. Thus, the inference of triangular-chain CRFs straightforwardly uses multiple Viterbi searches for linear-chain CRFs; thus, for a value of $z = k$, we compute the $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y}, z = k|\mathbf{x})$, and then repeat this procedure for all values of z.

The marginal probability distributions and the partition function $Z(\mathbf{x})$ are calculated via the forward–backward algorithm on the same search space described in Fig. 3. We introduce notations which modify the original forward–backward recursions for linear-chain CRFs. The forward values $\alpha_t(k, j)$ are the sum of the unnormalized scores for all partial paths of $z = k$ starting at $t = 0$ and converging at $y_t = j$ at time t. Also, backward values, $\beta_t(k, i)$, similarly define the sum of unnormalized scores for all partial paths of $z = k$ starting at time $t + 1$ with state $y_{t+1} = j$ and continuing until the end of the sequences $t = T+1$. We define the forward and backward vectors—$\alpha_t(\cdot, \cdot)$ and $\beta_t(\cdot, \cdot)$, respectively—by the following forms:

$$\alpha_t(k, j) \triangleq \sum_{i \in \mathcal{Y}} \phi_t(k, j, i, \mathbf{x}) \cdot \alpha_{t-1}(k, i) \tag{22}$$

$$\beta_t(k, i) \triangleq \sum_{j \in \mathcal{Y}} \phi_t(k, j, i, \mathbf{x}) \cdot \beta_{t+1}(k, j) \tag{23}$$

where the base cases of α value are defined by $\alpha_0(z, y_0) = 1$, if $y_0 = \mathsf{start}$ and otherwise, $\alpha_0(z, y_0) = 0$. Similarly, the base cases of β value are defined by $\beta_{T+1}(z, y_{T+1}) = 1$, if $y_{T+1} = \mathsf{end}$ and otherwise, $\beta_{T+1}(z, y_{T+1}) = 0$. α and β values depend on plane z; hence, the structure for α and β is a 3-D array.

In the forward–backward algorithm for linear-chain CRFs, it can be shown by recursion that the sum of final α yields the mass of all state sequences; thus, $\alpha_{T+1}(k, \mathsf{end}) = \sum_{j=1}^{J} \alpha_T(k, j)$. To extend it, we add a final dummy node end', which connects all end nodes end for each z. The mass of all state sequences for triangular-chain CRFs can be calculated by summing all end nodes, then $Z(\mathbf{x}) = \sum_z \alpha_{T+1}(z, \mathsf{end}) \cdot \varphi(z, \mathbf{x})$. Note that each $\alpha_{T+1}(z, \mathsf{end})$ at plane z is multiplied by $\varphi(z, \mathbf{x})$, which plays a prior role for the z-plane. Then, we define the $Z(z, \mathbf{x})$ as

$$\begin{aligned} Z(z, \mathbf{x}) &\triangleq \sum_{\mathbf{y}} \prod_{t=1}^{T} \big(\phi_t(z, y_t, y_{t-1}, \mathbf{x})\big) \cdot \varphi(z, \mathbf{x}) \\ &= \alpha_{T+1}(z, \mathsf{end}) \cdot \varphi(z, \mathbf{x}). \end{aligned} \tag{24}$$

Using forward–backward recursions, we can define the marginal distributions as follows:

$$p_{\boldsymbol{\lambda}}(z, y_t|\mathbf{x}) = \frac{\alpha_t(z, y_t) \cdot \beta_t(z, y_t) \cdot \varphi(z, \mathbf{x})}{Z(\mathbf{x})} \tag{25}$$

$$\begin{aligned} p_{\boldsymbol{\lambda}}(z, y_t, y_{t-1}|\mathbf{x}) &= \frac{\alpha_{t-1}(z, y_{t-1}) \cdot \beta_t(z, y_t)}{Z(\mathbf{x})} \\ &\quad \cdot \phi_t^2(z, y_t, y_{t-1}) \cdot \varphi(z, \mathbf{x}) \end{aligned} \tag{26}$$

$$p_{\boldsymbol{\lambda}}(z|\mathbf{x}) = \frac{Z(z, \mathbf{x})}{Z(\mathbf{x})}. \tag{27}$$

For MODEL2, we compute the marginal distributions $p_{\boldsymbol{\lambda}}(y_t|\mathbf{x}) = \sum_z p_{\boldsymbol{\lambda}}(z, y_t|\mathbf{x})$ and $p_{\boldsymbol{\lambda}}(y_t, y_{t-1}|\mathbf{x}) = \sum_z p_{\boldsymbol{\lambda}}(z, y_t, y_{t-1}|\mathbf{x})$.

The complexity of inference for triangular-chain CRFs is $\mathcal{O}(T|\mathcal{Y}|^2|\mathcal{Z}|)$, where T is the size of the sequence. An exact inference can be conducted efficiently if $|\mathcal{Y}|$ and $|\mathcal{Z}|$ are not large. In large state spaces for $\mathcal{Y}$ or $\mathcal{Z}$, however, the inference is often prohibitively expensive. This problem is alleviated by introducing two approximate inference methods based on the reduction of the spaces for $\mathcal{Y}$ and $\mathcal{Z}$. First, the state space $\mathcal{Y}$ can be reduced to a smaller partial space $\mathcal{Y}_z$ by adding constraints. Intuitively, there are a large number of "unseen" labels of $\mathcal{Y}$ given z, that is, only a small part of the possible labeling set $\mathcal{Y}_z$ is observed given topic label z in the data. Formally, for a fixed value of z, the sequence label set can be decomposed as $\mathcal{Y} = \mathcal{Y}_z \cup \mathcal{Y}_R$, where $\mathcal{Y}_z$ is the finite set of labels given z, and $\mathcal{Y}_R$ is the remaining set of unseen labels. In this case, the triangular-chain CRFs need not model the unseen labeling ($\mathcal{Y}_R$) precisely, which can be exploited by ignoring their features. This is achieved by fixing their potentials to zero. We call this scheme *a partial space constraint*.[3] For tasks in which the partial state space is sufficiently small, as $|\mathcal{Y}_z| < |\mathcal{Y}|$, both decoding and training times are remarkably reduced in triangular-chain CRFs.

Next, the pruning method is employed to reduce the space of $\mathcal{Z}$ by removing planes with $p_{\boldsymbol{\lambda}}(z|\mathbf{x}) < \epsilon$. $p_{\boldsymbol{\lambda}}(z|\mathbf{x})$ can be calculated easily by (27). Since $Z(z, \mathbf{x})$ should be calculated before pruning is performed, α recursions are finished first. Thus, we apply the pruning method only for training time. If we wish to predict the best labels without marginal probabilities in predicting a new instance, then we can ignore calculating $Z(\mathbf{x})$; hence, we cannot prune less confident planes. However, we can take advantage of the pruning method in some applications that require marginal probabilities (e.g., active learning). The pruning technique helps to reduce the training time, which can scale triangular-chain CRFs for large-scale problems.

[3]A similar technique was addressed in [25]. The difference is that the approach described in [25] uses regularities of the transition features (y_{t-1}, y_t), while we exploit regularities of the joint labels $(\mathbf{y}, z)$.

D. Parameter Estimation

Parameter estimation of triangular-chain CRFs can be performed using the conditional maximum log-likelihood. Let $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}, z^{(n)}) : 1 \leq n \leq N\}$ denote a training set and $\boldsymbol{\lambda} = \{\lambda_k\} \in \mathbb{R}^K$ denote a parameter vector. Over-fitting can be reduced by applying the regularization term to penalize a parameter vector whose norm is too large. A common choice of penalty is a spherical Gaussian prior with mean $\mu = 0$ and covariance matrix $\Sigma = \sigma^2 I$. Then, the objective function is defined as

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k f_k(z^{(n)}, y_t^{(n)}, y_{t-1}^{(n)}, \mathbf{x}^{(n)}) - \sum_{n=1}^{N}\log Z(\mathbf{x}^{(n)}) - \sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2} \quad (28)$$

where f_k uniformly represents a feature function for all local functions, for clarity. This function is known to be convex, which guarantees convergence to the global optimum.

Differentiating (28) can give the optimal parameter vector $\boldsymbol{\lambda}^*$ as

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \underbrace{\sum_{n=1}^{N}\sum_{t=1}^{T} f_k(z^{(n)}, y_t^{(n)}, y_{t-1}^{(n)}, \mathbf{x}^{(n)})}_{E_{\tilde{p}}\langle f_k\rangle} - \underbrace{\sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{z',y,y'} f_k(z', y, y', \mathbf{x}^{(n)}) p_{\boldsymbol{\lambda}}(z, y, y'|\mathbf{x}^{(n)})}_{E_p\langle f_k\rangle} - \frac{\lambda_k}{\sigma^2} \quad (29)$$

where $E_{\tilde{p}}$ is the empirical distribution of training data, and E_p denotes the expectation with respect to distribution $p_{\boldsymbol{\lambda}}$. E_p can be calculated by (25)–(27). After calculating $E_{\tilde{p}}$ and E_p, numerical optimization techniques can be used to estimate the parameters of triangular-chain CRFs. We optimize the parameters using a limited memory version of the quasi-Newton method (L-BFGS), which uses approximation to the Hessian [26]. The L-BFGS method converges super-linearly to the solution, so it can be an efficient optimization technique on large-scale natural language processing problems.

Jointly optimizing the parameters generally converges to the maximum slowly. To alleviate this problem, we exploit the parameter initialization technique, in which the solver is started with a parameter vector from an initial model $\boldsymbol{\lambda}'$ rather than a zero parameter vector. A starting point close to the solution may allow the solution to be found in fewer iterations [15]. For efficiency, an initial model should be efficient to calculate and close to the optimal parameters. In many situations, the pseudo-likelihood parameters are preferred [27]. The initial parameters are obtained by individually optimizing the objective functions for $p(\mathbf{y}|z, \mathbf{x})$ and $p(z|\mathbf{y}, \mathbf{x})$ as follows:

$$\mathcal{L}^1(\boldsymbol{\lambda}') = \sum_{n=1}^{N}\sum_{t=1}^{T}\log p_{\boldsymbol{\lambda}'}(y_t^{(n)}|y_{t-1}^{(n)}, y_{t+1}^{(n)}, z^{(n)}, \mathbf{x}^{(n)}) \quad (30)$$

$$\mathcal{L}^2(\boldsymbol{\lambda}') = \sum_{n=1}^{N}\log p_{\boldsymbol{\lambda}'}(z^{(n)}|\mathbf{y}^{(n)}, \mathbf{x}^{(n)}). \quad (31)$$

Then we merge them to seed the training procedure for triangular-chain CRFs.

E. Learning With Hidden Variables

Like other methods, access to fully labeled samples $(\mathbf{x}, \mathbf{y}, z)$ is assumed. However, the triangular-chain CRFs can be naturally applied to partially labeled data in which one of the tasks is unobserved. This section briefly discusses possible extensions for partially labeled data. There are two learning problems for triangular-chain CRFs

$$p_{\boldsymbol{\lambda}}(\mathbf{y}|\mathbf{x}) = \sum_{z} p_{\boldsymbol{\lambda}}(\mathbf{y}, z|\mathbf{x}) = \frac{Z(\mathbf{y}, \mathbf{x})}{Z(\mathbf{x})} \quad (32)$$

$$p_{\boldsymbol{\lambda}}(z|\mathbf{x}) = \sum_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y}, z|\mathbf{x}) = \frac{Z(z, \mathbf{x})}{Z(\mathbf{x})}. \quad (33)$$

We name the above marginalized distributions the *latent* triangular-chain CRFs.

The first problem (32) can be viewed as a mixture of linear-chain CRFs, in which a latent topic z dominates the probability distributions. It can be accomplished by grouping subdata using data clustering methods. This problem can be applied to tasks on the domain adaptation of sequential classifiers [28]. The second problem (33) is similar to hidden CRFs [23]. However, our model has a prior term $\varphi(z, \mathbf{x})$ for z depending on not only $\mathbf{y}$ but also $\mathbf{x}$. The inference of the second problem is efficiently performed by computing $Z(z, \mathbf{x})$, so we do not need to find the best assignment of sequences $\mathbf{y}$.

To learn these latent triangular-chain CRFs, we can employ various latent variable methods with the EM algorithm. We do not report these extended models in our experiments because they are beyond the scope of this paper.

F. Relation to Other Models

Here, we compare the triangular-chain CRFs with other models. If we let $|\mathcal{Z}| = 1$, then a triangular-chain CRF is identical to a linear-chain CRF. Otherwise, triangular-chain CRFs have parallel linear-chains. Moreover, triangular-chain CRFs are a special form of dynamic CRFs that have a dual-layered structure and restrict the top layer $\mathbf{y}^2$ to a single random variable z. Therefore, approximate techniques developed for dynamic CRFs can be also applied to our method.

Triangular-chain CRFs are also related to expert models [29]. One plane corresponding with z can be seen as a constrained expert which learns the constrained parameters from parts of data $(\mathbf{x}, \mathbf{y})$ for $z = k$. Most discriminative mixtures of expert models

are interested in finding an output $\mathbf{y}$ but not z. In contrast, our model aims to find the best sequence $\mathbf{y}$ with its best topic z.

Comparison to other CRF models suggests that ours is closely related to hidden CRFs. However, our work is distinct from hidden CRFs for two reasons. First, hidden CRFs are primarily interested in the main variable z, and other auxiliary variables $\mathbf{y}$ are included in the model simply to aid in predicting the main variables. In contrast, triangular-chain CRFs are interested in both variables. Therefore, hidden CRFs have different decoding and training schemes. No Viterbi search for finding the best assignment of sequences is performed during decoding. However, it is well known that training CRFs with latent variables is more difficult [16]. Second, the term φ and feature vector $\mathbf{x}_0$ are clear contributions of our work. [23] did not consider the term $\mathbf{x}_0$, which can be useful for incorporating the prior knowledge beyond the input observations $\mathbf{x}_{1:T}$. For instance, dialog context as the current topic and expectations from previous utterances helps guide the head gesture recognition in a multimodal interface [30]. In addition, triangular-chain MODEL1 also has edges between the observations and z, where the hidden CRF in [23] does not.

Parallel to our work, [31] developed a joint structured model to predict sentiment on different levels of granularity for a text. The proposed sentence–document model most closely resembles the triangular-chain structured model. In our notation, a sentence-level sentiment analysis is cast as sequence labeling, and a document-level analysis as sequence classification. Therefore, triangular-chain CRFs can be naturally applied to the domain of sentiment analysis; hence, it would be interesting to compare the two models. Even though the two approaches are similar, there are two clear distinctions. First, triangular-chain CRFs are probabilistic models learned by maximum-likelihood estimation, while the sentence-document model is nonprobabilistic. Reference [31] instead used online large margin learning, a popular method in natural language processing. Although the margin-based approach makes training faster, calculating the posterior probability $p(\mathbf{y}, z|\mathbf{x})$ is important in some applications, thus making CRFs preferable. Second, our method is of interest for large-scale problems. In sentiment analysis, the state space is very small, $\mathcal{Y} = \{\text{positive, neutral, negative}\}$ and $\mathcal{Z} = \{\text{positive, negative}\}$. Although [31] has not been demonstrated to work on larger scale problems, we have considered the scalability of triangular-chain CRFs for large-scale problems.

IV. SYNTHETIC DATA EXPERIMENTS

In the previous section, we argued intuitively that triangular-chain CRFs may perform better on sequence data that are grouped by topic z. This section verifies this intuition via experiments on synthetic data.

A. Experimental Design

This experiment is designed to assess whether triangular-chain CRFs capture the dependencies between sequences $\mathbf{y}$ and topic z or not. More definitely, it addresses the problem of modeling multitopic sequences, where the data can be grouped by topic and each group of data is drawn from i.i.d. Synthetic data was generated using randomly chosen generative models $p(\mathbf{x}, \mathbf{y}, z)$. By applying chain rules, we obtain that $p(\mathbf{x}, \mathbf{y}, z) = p(\mathbf{x}|\mathbf{y}, z)p(\mathbf{y}|z)p(z)$. Unfortunately, in this generative model, $\mathbf{y}$ cannot directly affect z due to its directional property. We first randomly sample z and then z influences the sampling process for $\mathbf{y}$ and $\mathbf{x}$. It follows that $\mathbf{y}$ and $\mathbf{x}$ sequences are generated by topic z, while z is independently drawn. This experiment, therefore, aims to assess how z affects the sequential labeling problem, and only the results for sequence labeling are reported.[4]

We follow [15] for general setup. In particular, we use 26 observation values, A-Z ($|\mathcal{X}| = 26$), five sequence labels, a-e ($|\mathcal{Y}| = 5$), and five state labels, 0–4 ($|\mathcal{Z}| = 5$). We generate examples from a generating distribution with prior probability $p(z)$, transition probabilities $p(y_t|y_{t-1}, z)$, and emission probabilities $p(x_t|y_t, z)$. In order to limit the size of the Bayes error rate for the resulting models, the probability table is constrained to be regular. More concretely, $p(y_t|y_{t-1}, z)$ can have at most two large values (obviously, $0 < p \leq 1$) for each y_t, and $p(x_t|y_t, z)$ can have at most three large values for each x_t. The remaining entries of the probability table have a small positive value ($0 < p \leq 0.001$), but not zero. As a result, each hidden variable y and z mostly corresponds to a small number of observable values for x, but any path of sequence labels is possible. Since the transitions and emissions are changed by topic, capturing the dependencies between $\mathbf{y}$ and z is crucial in multitopic sequence labeling problem.

In order to balance the multitopic sequences, the transition probability is interpolated as $p_\omega(y_t|y_{t-1}, z) = \omega p(y_t|y_{t-1}, z) + (1 - \omega)p(y_t|y_{t-1})$. The observation probability is also interpolated in the same way. The prior probability is defined as $p_\omega(z) = \omega p(z) + (1 - \omega)p'(z)$, where $p'(z) = 1$ if $z = 0$, and otherwise, $p'(z) = 0$. Thus, for $\omega = 0$, the sequences $\mathbf{y}$ and $\mathbf{x}$ are generated from the same topic, and for $\omega = 1$, all sequences of $\mathbf{y}$ and $\mathbf{x}$ are perfectly related with z. A higher value of ω implies that the data, which have different topics, are balanced. For each setting of ω, we sample 20 different generating distributions. From each generating distribution, we sample 1000 training instances of length 25, and 1000 testing instances. Both linear-chain and triangular-chain CRFs are trained for 200 iterations with a Gaussian prior variance of 10. All experiments were implemented in C++ and executed in Linux with XEON 2.8-GHz dual processors and 2.0 GB of main memory.

B. Results

First, triangular-chain CRFs performed better than linear-chain CRFs. Fig. 4 shows the difference in testing

[4]For the sequence classification task, we duplicate sequence observation $\{x_t\}$ for $t = 1, \ldots, T$, as an observation $\mathbf{x}_0$. This is trivial because all $\mathbf{y}$ and z share common observations. In this setting, we observe that there is no difference in classifying z given $\mathbf{x}_0$ (by using 0-order CRFs) for arbitrary setting of the interpolation rate ω. In our result, the averaged accuracy is 99.97% and standard deviation is 0.15. Therefore, instead of evaluating the performance for prediction of z, we only consider the prediction of $\mathbf{y}$.

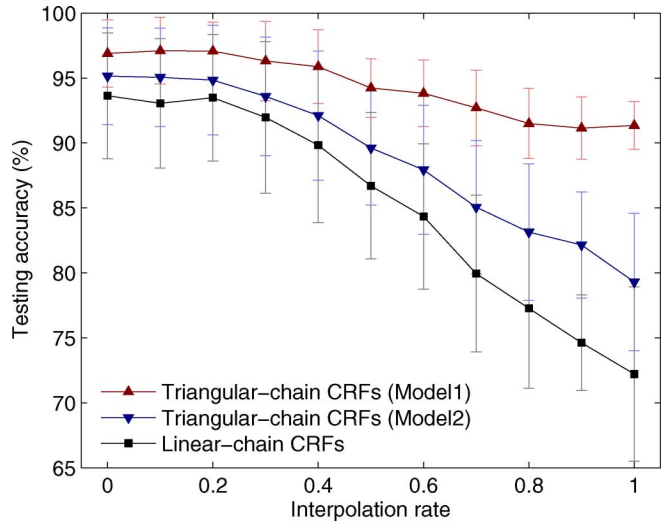


Fig. 4. Results for synthetic data. A higher value of ω makes generated sequence labels more dependent on the topic. Each point is averaged over 20 data sets and each error bar represents a standard error of 20 trials.

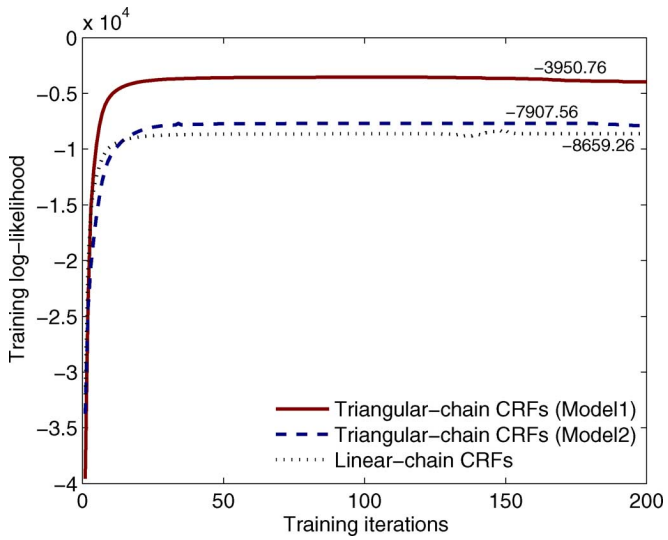


Fig. 5. Log-likelihood functions estimated by linear-chain and triangular-chain CRFs on synthetic data.

accuracy as a function of ω. We use $\omega = \{0, 0.1, \ldots, 1.0\}$ for 220 synthetic generating models. Triangular-chain CRFs have significantly higher testing accuracies (MODEL1; 94.37, MODEL2; 88.88) than the linear-chain CRFs (85.19). MODEL1 and the linear-chain CRFs had a mean difference of 9.18 and a correlation of 0.8817 for the 220 data sets, and the values for MODEL2 and the linear-chain CRFs were 3.72 and 0.9586. The accuracy of the linear-chain CRFs decreased when ω is increased because they capture only one of the multitopic distributions generated by $p(\mathbf{x}, \mathbf{y}, z)$. In contrast, triangular-chain CRFs can effectively model multitopic sequences. Increasing ω also decreased the accuracy of triangular-chain CRFs, since the distributions of multitopic sequence are partially overlapped, complicating the learning problem more.[5] This result shows that the triangular-chain CRFs are potentially appropriate for capturing the relations between sequence and topic. To verify that joint learning is better, we plot the estimated training log-likelihood functions in Fig. 5. This result shows that independently learning $p(\mathbf{y}|\mathbf{x})$ converges to a solution in few iterations. However, jointly learning $p(\mathbf{y}, z|\mathbf{x})$ converges to better solutions (in this experiment, MODEL1 is better than MODEL2).

Second, we examine how the number of topics affects the performance of models. $|\mathcal{Z}|$ was varied between 2 and 20 to generate synthetic data. A higher value of $|\mathcal{Z}|$ makes the learning problem more difficult, because the hypotheses space increases according to the number of topics. In this experiment, the parameter ω was set to 0.75, and all points were averaged over 20 data sets. Fig. 6(a) shows the relation between number of topics and testing accuracy. Here accuracies were decreased by increasing number of topics. However, the effect on triangular-chain CRFs was lower than for linear-chain CRFs. Consequently, our method could potentially be useful for multitopic problems in which distributions are diverse. While the performance of MODEL1 is significantly better than that of MODEL2, the number of parameters for MODEL2 is much smaller. The parameter vector size of MODEL2 increases linearly in proportion to the size $|\mathcal{Z}|$, which is competitive with linear-chain CRFs. Fig. 6(b) illustrates this result.

Finally, we consider the efficiency of the training algorithm. To reduce the training time, we propose two techniques: parameter initialization (INIT) and pruning (PRUNE).[6] To verify that these methods reduce training time, this experiment used $\omega = \{0, 0.25, 0.5, 0.75, 1.0\}$, for 100 synthetic generating models, each of which was trained for 500 iterations with a Gaussian prior variance of 10. We use a pseudo-likelihood parameter as an initial weight (estimated in 20 iterations) and prune the planes down to ϵ of 0.001. Fig. 7 shows the log-likelihood as a function of training time. The result indicate that seeding the parameter vector using pseudo-likelihood can be a reasonable starting point for training triangular-chain CRFs. (Also, the INIT method converged to a solution in few iterations.) Initially, pruning the low-confident planes does not improve the log-likelihood, but it does increase the convergence speed when models are close to the solution. Table I summarizes the results, where the time complexities of MODEL1 and MODEL2 are approximately the same. Accuracies for MODEL1 and MODEL2 are 93.98 ($\pm$3.27) and 88.09 ($\pm$7.26), and trained log-likelihoods are -3788.22 ($\pm$1996.80) and -8250.00 ($\pm$5054.40), respectively. Learning the models with both techniques (INIT+PRUNE) significantly re-

[5]At $\omega = 0$, although there is no effect of z for modeling $\mathbf{y}$ given $\mathbf{x}$, triangular-chain CRFs often outperform linear-chain CRFs. In this case, MODEL1 and MODEL2 are learned from data $x_{1:T}$ and $\mathbf{x}_0$, which reduces the overfitting of the parameters. In particular, the result of MODEL1 is more noteworthy than MODEL2. If we exclude the features of $\mathbf{x}_0$, then triangular-chain CRFs are identical to linear-chain CRFs at $\omega = 0$.

[6]In this synthetic experiment, the result with the partial space constraint method is not reported since joint label spaces are dense. Compared to the full state space $|\mathcal{Y}| = 5$, the averaged number of $|\mathcal{Y}_z|$ is 4.95 (± 0.42); hence, it is not expected to speed the inference.

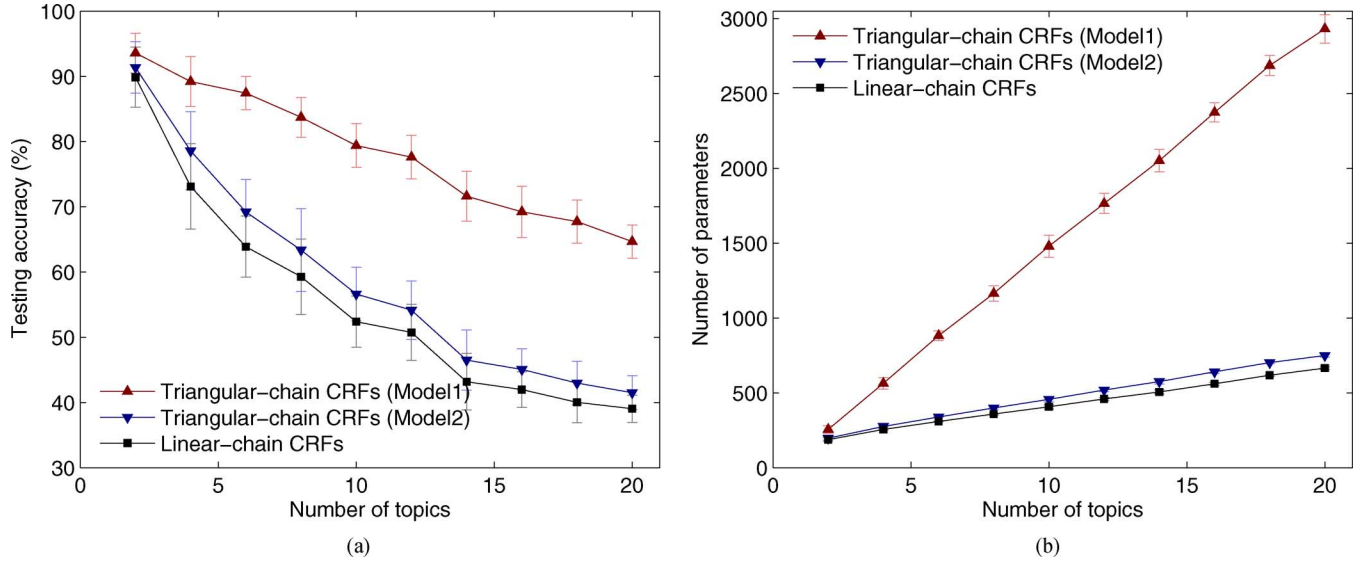


Fig. 6. Relation to number of topic and model. Each point is averaged over 20 data sets and each error bar represents a standard error of 20 trials. (a) Accuracy. (b) Parameter size.

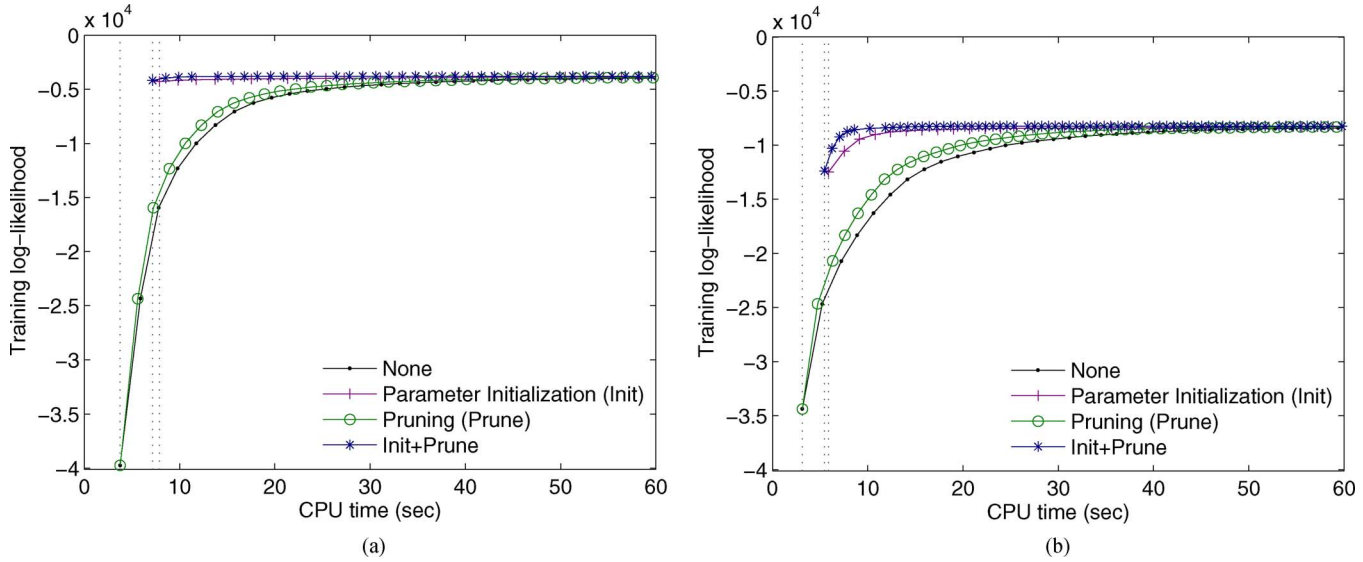


Fig. 7. Results of training complexity on synthetic data. Each point is averaged over 100 data sets. The entire training time is approximately 800 s, but the function almost converges at 60 s. (a) MODEL1. (b) MODEL2.

TABLE I
RESULTS OF TRAINING COMPLEXITY ON SYNTHETIC DATA

Method	Model1		Model2	
	Time(s)	#Iters	Time(s)	#Iters
NONE	769(±72)	391(±110)	801(±137)	476(±70)
INIT	687(±67)	340(±115)	790(±169)	464(±91)
PRUNE	656(±57)	391(±111)	597(±113)	465(±85)
INIT+PRUNE	583(±67)	338(±117)	592(±154)	439(±111)

duce the cost of training time. Compared to the baseline (NONE), using both techniques requires only three quarters of the training time.

V. SPOKEN LANGUAGE UNDERSTANDING

This section evaluates the triangular-chain CRFs on real-world dialog data. Our method is found to perform better for spoken language understanding tasks since it effectively captures the dependencies between $\mathbf{y}$ and z.

A. Task Description and Prior Work

The goal of spoken language understanding (SLU) for information-seeking dialog applications is to extract meanings from natural language speech and infer the speaker's intention to provide a natural human–computer dialog interface [10], [22], [32], [33]. To understand the user's utterance, most SLU systems define a semantic frame, i.e., a formal structure of predicted meanings consisting of slot/value pairs. A SLU frame is usually divided into two main components: dialog act (DA) and named entity (NE). A DA presents the meaning of an utterance at the discourse level, which is approximately the equivalent of *intent* [10] or *subject slot* [32] in the practical dialog

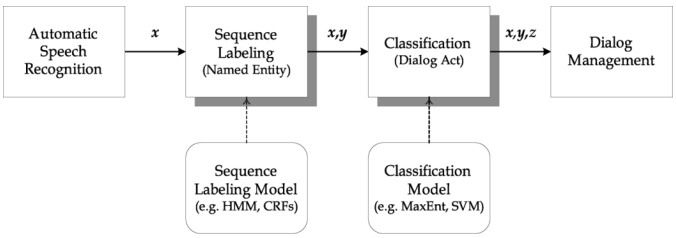


Fig. 8. Logical diagram of the language understanding process in a spoken dialog system.

system[7]. Moreover, the NE is an identifier of an entity such as a person, location, organization, or time. This paper itemizes NE as the domain-specific semantic meaning of a word. To develop a goal-oriented dialog system, we therefore attempt to understand the user's utterance based on two subproblems; DA and NE extractions.

Fig. 8 demonstrates the systematic diagram of a traditional SLU process for a dialog system. An automatic speech recognizer transcribes the speech into a string $(\mathbf{x})$ and passes it to the SLU modules. The SLU modules form a semantic frame consisting of $(\mathbf{x}, \mathbf{y}, z)$ and pass it to the dialog manager. In the statistical framework, SLU tasks can be considered to be sequence classification and sequence labeling problems. In the context of previous notations, a sequence of input words is defined as $\mathbf{x}$, an NE sequence for the state variables as $\mathbf{y}$ and a DA type for the class variable as z. The NE recognition problem requires assigning a label to a phrase rather than a word. The standard method for dealing with this segmentation problem is to use "BIO" encoding [34], in which the slot labels are drawn from a set of classes constructed by extending each label by X-B, X-I, or O. Here, X-B means "begin a phrase of slot X," X-I means "continue a phrase of slot X," and O means "not in a phrase." Fig. 9, for example, shows an utterance $(\mathbf{x})$, named entity tags $(\mathbf{y})$, and dialog act (z) for the air travel information service (Air–Travel) domain.

Current state-of-the-art SLU systems use cascaded or pipeline schemes [10], which are often accomplished by training the NE model; then they use its prediction as a feature for the DA classifier. NE plays an important role in identifying the DA, which can also improve the performance of the DA classifier in this scheme. However, this cascaded approach has a significant drawback; the NE recognition module cannot take advantage of information from the DA identification module. Our assumption here is that the problem of modeling DA and NE are significantly *correlated*, i.e., DA information influences the NE recognition task and vice versa. Thus, we concurrently optimize the DA and NE models. This problem can be solved using a complex model to reflect interdependency between DA and NE. Thus, the triangular-chain CRFs are applied to the joint SLU task.

[7]Note that our definition slightly differs from [6]. Here DA is a domain-specific intent and no utterances have multiple DAs.

B. Data Sets and Experiment Setup

We evaluate our method on two goal-oriented dialog data sets: Air–Travel (English; travel agency service—DARPA-Communicator) and Telebank (Korean; automatic response system for banking service). All data sets are collected and annotated to develop spoken dialog systems that consist of annotated DA and NE. In practice, a realistic dialog system for human–computer interface prefers a short and restrictive dialog. The average number of words per utterance is less than $5 \sim 8$ words, and the average number of classes per utterance is $1 \sim 3$ in these data sets. Both data sets are summarized in Table II. Note that the state space is larger than the synthetic data. Therefore, these data sets are appropriate to evaluate our method for large-scale problems. Fig. 10 shows the statistics for annotation. Each plot shows relative frequencies of DA and NE classes, and ranked class IDs are depicted. (The class IDs are ordered from most to least frequent.)

All data sets were evaluated with the results averaged over a tenfold cross validation (CV) with a 90/10 split of the data. As a standard, precision and recall were computed, which are evaluated on a per-entity basis and combined into a micro-averaged F_1 score ($F_1 = 2PR/(P+R)$). The significance of our results was verified with the McNemar paired test [35], which is based on individual labeling decisions to compare the correctness of two models.

We have selected feature templates such as the current word, ± 2-context words, bigram, distance-n bigram, prefix, and suffix features for NE recognition. Moreover, the bag-of-words and bag-of-bigrams are used for DA classification. Both linear-chain and triangular-chain CRFs were trained for 100 iterations with a Gaussian prior variance of 10. In addition, the methods were evaluated for both text and spoken input. For spoken inputs, the HTK-based recognition system was used [36]. The word error rates for Air–Travel and Telebank were 27.18% and 20.79%, respectively.

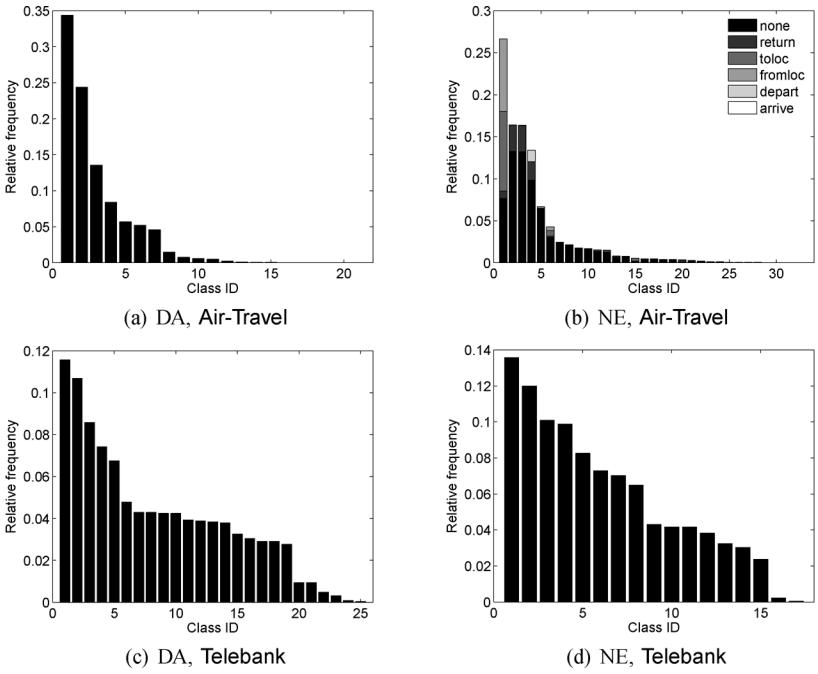
C. Results

Our method was compared to other approaches using various baseline systems.

- INDEP: This model is obtained by training DA and NE classifiers independently. This completely ignores the other classifier's outputs. We use the 0-order CRF as a DA classifier and the first-order linear-chain CRF as an NE classifier.
- CASCADE1: This model is obtained by using the NE prediction. We first extract NE and then augment a DA feature vector with it for DA classification. The result of NE prediction is the same as that of INDEP, so it is not reported.
- CASCADE2: This model is obtained using the DA prediction. This is an inverse pipeline system of CASCADE1 since we first identify DA and then augment a feature vector with it for NE classification. In this model, the result of DA prediction is the same as that of INDEP.

z	show_flight										
y	O	O	O	O	from. city-B	O	to. city-B	to. city-I	O	none. month-B	none. day_number-B
x	Show	me	flights	from	Denver	to	New	York	on	Nov.	18th

Fig. 9. Example from the Air–Travel domain.



(a) DA, Air-Travel (b) NE, Air-Travel

(c) DA, Telebank (d) NE, Telebank

Air-Travel	DA	reply_date_time, show_flight, reply_city, request_return, request_hotel, request_car, reply_airline, reply_destination, reply_city_from_to, reply_airport, reply_origin, reply_depart_time, request_air, ask_fare, request_arrive, greet, ask, repeat, reply_from, answer
	NE (1st)	city_name, day_number, month_name, period_of_day, time, state_name, airline_code, hotel_name, negative, rental_company, day_name, time_relative, year_name, car_class, country, period_mod, round_trip, today_relative, airport_name, date_relative, airport_code, cost_relative, positive, leg_num, hotel_loc, flight_mod, hotel_duration, cost_number, cost_unit, flight_number, reason_of_visit, hotel, question
	NE (2nd)	none, return, toloc, fromloc, depart, arrive
Telebank	DA	search_about, search_period, closing, search_amount, search_time, search_kind, search_state, search_service, confirm_qualified, confirm_care, confirm_service, search_how_to, search_info, confirm_limit, search_qualified, search_rate, search_count, confirm_benefit, confirm_info, search_calculate, search_benefit, search_limit, search_refund, yes, search_profit
	NE	about, doing, service, period, time, qualified, amount, info, kind, rule, limit, benefit, count, rate, via, refund, profit

(e) DA and NE lists. All classes are ordered by their class ID. In the Air-Travel data set, the NE category has a two-level hierarchy: 33 first level classes and 6 second level classes, for a total of 110 class combinations.

Fig. 10. Description of annotation for dialog data sets.

- RERANK1: This model is obtained by rescoring the n-best results of the NE classifier. We first produce n-best NE sequences, and then the DA classifier exploits this information. Finally, the best score is selected in an interpolation fashion: $p = \gamma p(z|\mathbf{x}) + (1-\gamma)p(\mathbf{y}|\mathbf{x})$. It can be viewed as a joint extraction method where learning is not jointly optimized. We empirically set the best parameters as ($n = 5$ and $\gamma = 0.2$).
- RERANK2: This model is obtained by rescoring the n-best results of the DA classifier. This is an inverse system of RERANK1 ($n = 10$ and $\gamma = 0.8$).
- JOINT1&2: This is our proposed method. It is obtained by training DA and NE jointly, which not only jointly optimizes the DA and NE tasks in the training step, but also jointly extracts the $(\hat{\mathbf{y}}, \hat{z})$ pair in the prediction step. The triangular-chain CRF MODEL1 and MODEL2 are used for the joint SLU system. During training, a pseudo-likelihood parameter is used as an initial weight (estimated in 20 iterations) and the planes are pruned down to ϵ of 0.001.

Our method performed better than the baseline systems. Tables III and IV show the experimental results for text and speech input, respectively. The results of CASCADE1&2 show that DA is useful for predicting NE and vice versa. However, the cascaded approach cannot improve both tasks. While the approach of RERANK1&2 improves both DA and NE performances, the reranking method reduces the overall performance of SLU. The reranking method based on the cascaded model cannot take advantage of interdependencies and finds a local solution, due to the limit of n-best lists. Moreover, RERANK1&2 systems often do not pass the McNemar test, compared with INDEP and CASCADE models. Thus, the reranking approach is not the best approach for joint extraction.

Our JOINT system significantly improves performance by exploiting the dependence between DA and NE. On most

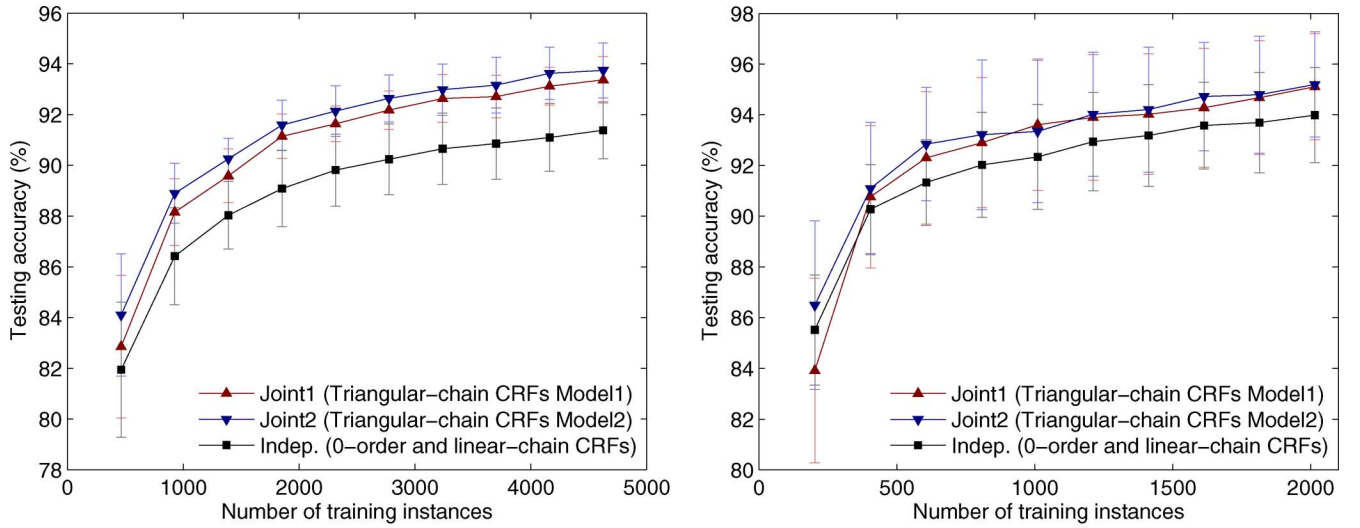


Fig. 11. Learning curves for dialog data sets, averaged over tenfold CVs. The testing accuracies for DA and NE are averaged. JOINT models perform significantly better than the INDEP model. (a) Air–Travel. (b) Telebank.

TABLE II
STATISTICS OF DIALOG DATA SETS

Data set	#Sent	#Words	#DA	#NE	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{Z}\rvert$
Air-Travel	5,138	25,024	5,138	10,237	3,632	110	21
Telebank	2,238	16,480	2,238	2,278	1,813	33	25

TABLE III
RESULT OF DIALOG DATA (TEXT). F_1 SCORES ARE AVERAGED OVER TENFOLD CV'S WITH STANDARD ERRORS

Method	Air-Travel		Telebank	
	F_1(DA)	F_1(NE)	F_1(DA)	F_1(NE)
INDEP	92.09 (±1.03)	90.67 (±0.80)	95.09 (±1.68)	92.88 (±1.49)
CASCADE1	92.70† (±1.20)	N/A	96.36† (±1.23)	N/A
CASCADE2	N/A	94.34† (±0.49)	N/A	92.97 (±1.34)
RERANK1	92.75† (±1.13)	89.75 (±0.82)	96.32† (±1.27)	92.18 (±1.81)
RERANK2	92.56† (±1.10)	94.39†‡ (±0.56)	95.42† (±1.64)	93.16 (±1.18)
JOINT1	92.96†(±0.89)	93.78† (±0.83)	96.51† (±1.49)	93.47 (±1.79)
JOINT2	**93.07**†‡ (±1.21)	**94.42**†‡ (±0.38)	**96.93**†‡ (±1.33)	**93.53**† (±1.26)

†Statistically, these results are significantly better than the INDEP model, $p < 0.05$.
‡Statistically, these results are significantly better than the CASCADE model, $p < 0.05$.

TABLE IV
RESULT OF DIALOG DATA (ASR)

Method	Air-Travel		Telebank	
	F_1(DA)	F_1(NE)	F_1(DA)	F_1(NE)
INDEP	78.44 (±1.26)	75.67 (±1.32)	90.04 (±1.33)	84.67 (±2.59)
CASCADE1	79.00 (±1.62)	N/A	90.85 (±2.04)	N/A
CASCADE2	N/A	78.88† (±1.24)	N/A	84.77 (±1.83)
RERANK1	79.04 (±1.59)	74.74 (±1.28)	91.10 (±2.01)	83.64 (±2.66)
RERANK2	78.82 (±1.43)	78.90† (±1.19)	90.02 (±1.32)	85.01 (±1.94)
JOINT1	**79.41**† (±1.25)	78.21† (±1.37)	**91.36**† (±1.27)	**85.93**†‡ (±2.43)
JOINT2	78.76 (±1.08)	**78.92**† (±1.17)	91.19† (±2.11)	85.42†‡ (±2.50)

F_1-scores, the joint model has improved over INDEP, CASCADE, and RERANK systems. Also, the differences in accuracies between JOINT and other systems are statistically significant (p-values are calculated by using the McNemar test). Note that the result of the McNemar test between JOINT and RERANK is similar to that of the JOINT and CASCADE system. This result shows that DA and NE are valuable to each other, and joint learning and inference can improve the performance of both tasks.

Although JOINT1 is more detailed, JOINT2 often outperforms JOINT1. There are two possible reasons for this. JOINT1 is too biased for small training samples and does not generalize well to the unseen data. Since our SLU task is a large-scale problem, more data is needed to learn the JOINT1 model, which has a large set of parameters. Another reason is that some words (or features) are independent of DAs given NEs; thus, enforcing the link of $\mathbf{x}$ and z acts as noise and causes prediction errors for this situation. For example, city names such as "Boston" and "Denver" are independent of "show_flight" and "request_return" given "none.city_name" and "return.city_name." Therefore, JOINT2 is more appropriate for statistical SLU problems, while JOINT1 models synthetic data well, where the distribution is definitely changed by topics.

Fig. 11 shows the difference in performance of the INDEP and JOINTmodels as a function of the training set size. In the Air–Travel data set, the INDEP model with the full training set of approximately 4600 utterances performs equivalently to the JOINT model on only 2000 training examples, a 60% reduction in the training set. Similarly, the Telebank data set has a reduction of the training set of approximately 60%.

To verify that the joint learning is better than the single learning, a learning curve of log-likelihood functions was plotted. Fig. 12 shows the estimated (conditional) log-likelihood functions for INDEP and JOINT1&2. This implies that jointly optimizing DA and NE finds a better maximum-likelihood solution compared to DA or NE only; i.e., the joint model boosts performance on the individual task when compared to classifiers that handle the tasks independently.

For practicality, we evaluated the space and time complexity for Air–Travel data. Table V summarizes the results, which are averaged on tenfold CV, but standard deviations are omitted for space reasons. The decoding time is measured per utterance. For

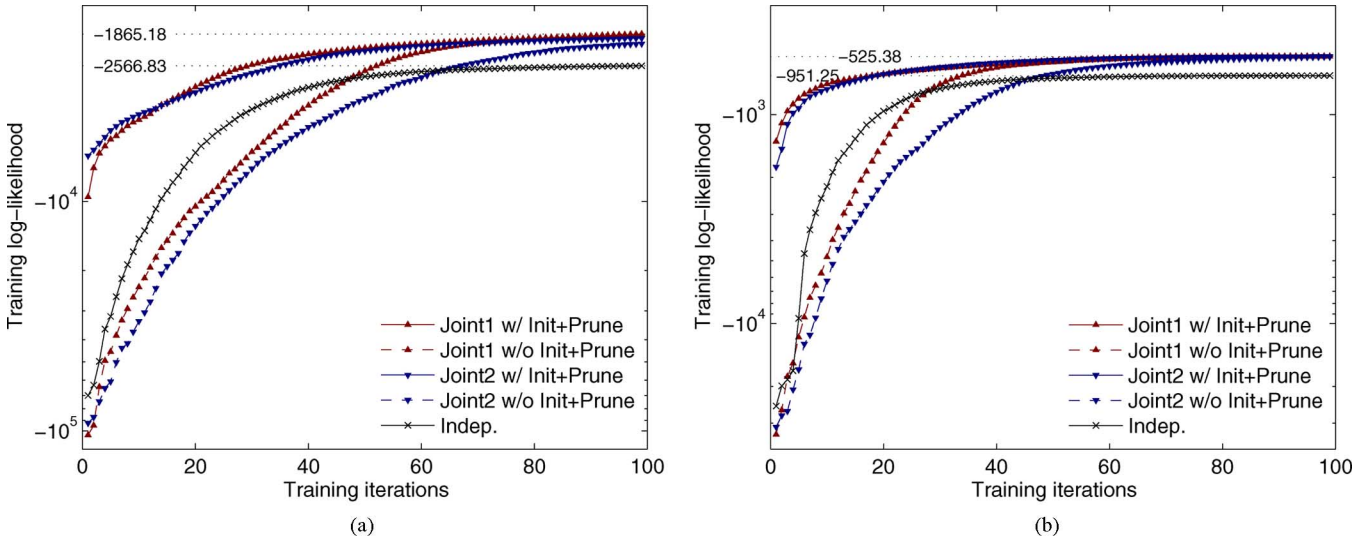


Fig. 12. Log-likelihood functions for dialog data sets, averaged over tenfold CVs. For INDEP, the training was run independently, and the log-likelihoods were aggregated. Although the speed of convergence differs between without INIT+PRUNE and with INIT+PRUNE, there is no difference in the final classification performance (i.e., the F_1 scores). (a) Air–Travel. (b) Telebank.

TABLE V
COMPARISON OF SPACE AND TIME COMPLEXITY

Method	# Parameter	Training time	Decoding time
INDEP	8,124	2,250s	5.2ms
CASCADE1	8,356	2,278s	5.3ms
CASCADE2	8,377	2,252s	5.4ms
RERANK1	8,356	2,278s	78.5ms
RERANK2	8,377	2,252s	50.1ms
JOINT1 (full)	17,826	36,464s	106.2ms
JOINT2 (full)	8,377	21,472s	46.1ms
JOINT1 (partial)	11,942	5,504s	11.1ms
JOINT2 (partial)	8,377	3,647s	8.2ms

INDEP, the 0-order CRF and linear-chain CRF were run individually, and parameter and times were aggregated. While INDEP system has no parameters for the dependencies between $\mathbf{y}$ and z, CASCADE, RERANK, and JOINT systems need more spaces to store the joint features. There are no significant differences in the training time of INDEP, CASCADE and RERANK. However, decoding in RERANK is more expensive since it has an overhead performing n-best searches and rescoring.

Our algorithm originally takes 51 936 ($\pm$3489) and 37 364 ($\pm$3418) seconds for training. To reduce the time complexity, we used INIT and PRUNE methods, so that the parameter weights are learned within 36 464 ($\pm$2812) and 21 472 ($\pm$1894) seconds, and trained log-likelihoods are -1865.18 ($\pm$15.55) and -1944.30 ($\pm$14.24) for JOINT1 (full) and JOINT2 (full), respectively. The decoding times are 106.2 ($\pm$6.85) and 46.1 ($\pm$4.06) ms for JOINT1 and JOINT2, respectively. This is a consequence of the time complexity of triangular-chain CRFs, which is linear in the state space size $|\mathcal{Z}|$. (Note that $|\mathcal{Z}| = 21$, and training and decoding times are approximately 20 times those of INDEP.) Fortunately, we reduced the time complexity using the partial space constraint described in Section III-C. The averaged number of $|\mathcal{Y}_z|$ is 12.38 ($\pm$5.49), while a full state set $|\mathcal{Y}|$ is 110. Although this method has a limitation that a novel instance not appearing in training data is impossible to decode, in practice the partial space constraint model remarkably reduced the inference times without significant loss of performance. F_1 scores for DA and NE classification (ASR) are 78.28 ($\pm$1.17) and 77.43 ($\pm$1.53) for JOINT1 (partial) and 78.97 ($\pm$1.22) and 78.56 ($\pm$1.10) for JOINT2 (partial). These results indicate that triangular-chain CRFs could be scaled-up for large-scale applications.

D. Analysis

SLU tasks have shown that the triangular-chain CRFs outperform competing models. Triangular-chain CRFs capture interdependencies between $\mathbf{y}$ and z, which is the basic cause for the improved performance. This section presents an analysis to provide more explanations for the improved performance of our model.

Let us consider two utterances; "I'd like to return on July twenty fourth in the morning" with DA "request_return," and the utterance "July twenty fourth in the early afternoon" with DA "show_flight." In this example, the user wishes to reserve a flight on a certain date and time. However, the user has a different goal aside from reserving a flight. Clearly, the first utterance asks the system to inform the flight schedule for returning, while the second one answers the system's question of when the user hopes to board the flight. For practicality, the system should divide the requests for time-related information into two cases. The word "July" in the first sentence is a "return.month," but an independent model usually classifies it incorrectly as "none.month," that is, as an annotation of "July" in the second sentence (F_1 score for "return.month"; 91.99). We can reduce the misclassification errors by exploiting DA with "request_return" and "show_flight" for two cases (F_1 score for "return.month"; 97.17). This empirical result clearly shows that DA and NE provide information to each other.

How the interdependencies between z and $\mathbf{y}$ work for triangular-chain CRFs can be inspected by considering the particular

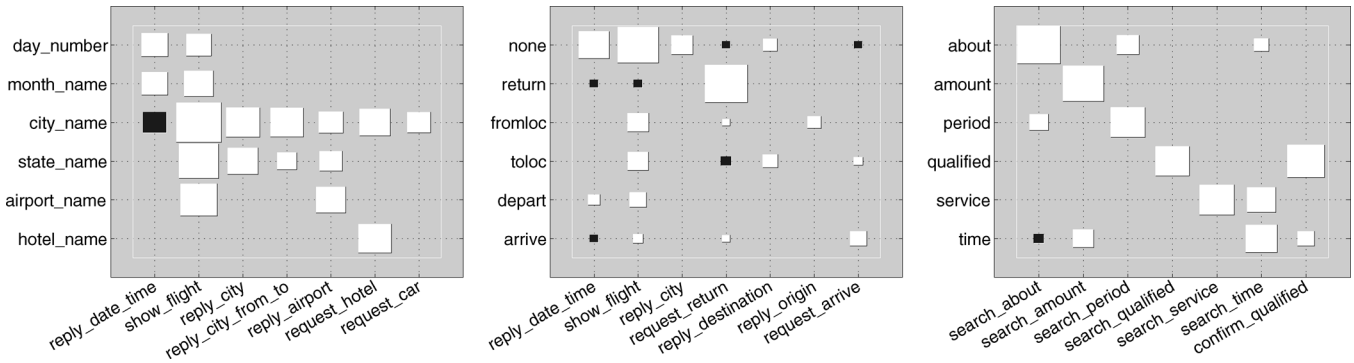


Fig. 13. Hinton diagrams for visualizing trained weights. The left diagram represents the dependencies between first level of NEs and DA, and the middle diagram indicates the dependencies between second level of NEs and DA for Air–Travel data. Equally, the right diagram shows the dependencies between NE and DA for Telebank data.

values of the parameters the model learns for function $f_k^3(y_t, z)$ on MODEL2. For example, the NE "hotel_name" is probably useless for data that are related to requests of flights, but potentially useful for hotel reservations. Fig. 13 illustrates the Hinton diagrams to visualize this feature's weights [37]. In a Hinton diagram, the white boxes present positive weights, and black boxes present negative weights. The size of the box depicts the absolute value of the weight. In this figure, each column corresponds to a DA and each row corresponds to an NE. This result only reports frequent classes, and merges the "X-B" and "X-I" weights for each entity type. As Fig. 13 shows, the DA "request_hotel" is a good indicator of entity "*.hotel_name," because its value is strongly positive. The empty areas such as the "airport_name" row and "request_hotel" column means that no feature $f_k^3(y_t, z)$ corresponding to this position is observed in training data, so these weights are set to zero. This analysis supports the hypothesis that the dependencies between NE and DA are useful to improve the performance because the co-occurrence patterns are actually sparse.[8]

VI. CONCLUSION AND FUTURE WORK

Despite the increasing interest in problems in which sequential labeling and sequence classification are jointly represented, no study to date has investigated whether CRFs can be applied to such problems. We have undertaken a problem in which a sequence $\mathbf{y}$ and a topic z are statistically correlated given $\mathbf{x}$. This paper has presented a new model, triangular-chain CRFs, to jointly learn and predict two tasks: sequential labeling and sequence classification.

Triangular-chain CRFs encode interdependence between sequence $\mathbf{y}$ and meta-sequence information z using a triangular-chain structure, where learning and inference are efficient. In synthetic data, triangular-chain CRFs were significantly better than linear-chain CRFs. The method was applied to the problem of spoken language understanding, showing its usefulness as an alternative to the existing cascaded approach. An attractive feature of our method is that it represents the two problems in a single graphical model, naturally embedding their dependence. This generic idea can be applied to any structured prediction algorithm, for example the hidden Markov model [38] and maximum-margin Markov network [39].

Despite the clear potential of triangular-chain CRFs, extending the model to large-scale problems still remains a significant challenge. To address this problem, we proposed three naive methods of parameter initialization, pruning, and partial space constraint, which reduced training times. However, the complexity of triangular-chain CRFs grows with the number of sequence labels and topics, so parameter estimation can become intractable. Thus, we hope to develop a practical method to scale triangular-chain CRFs in a principled way.

Recently, researchers have begun to study the acceleration of the training of CRFs. [40] presented the sparse forward–backward algorithm to make training of linear-chain CRFs more efficient. [25] described a tied potential method which constrains the labeling considered in each feature function, such that the functions can detect only a relatively small set of labelings. Both approximate inference techniques efficiently compute the marginals with a significantly reduced runtime, resulting in faster training and decoding.

Moreover, local training methods such as piecewise pseudo-likelihood [41] could provide a way to scale up CRFs for large-scale natural language applications. The local training method is especially attractive when collecting the joint data is expensive as it allows enough data for each task to be accessed. Finally, we can employ the stochastic gradient methods [42], which are online and scale sublinearly with a large amount of training data. Stochastic gradient methods, which make gradient steps based on subsets of the data, have recently been shown to converge significantly faster for CRFs training than batch methods, which evaluate the gradient of the entire data set before updating the parameters. In future research, we plan to apply fast inference and parameter estimation methods and explore more advanced techniques to make training and decoding of triangular-chain CRFs more efficient.

[8]The interested reader is invited to visit http://home.postech.ac.kr/~stardust/research/tccrf/ to explore more results, including per-class F_1 scores, McNemar's test, full Hinton diagrams and others.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for helpful comments.

REFERENCES

[1] T. G. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition*, T. Caelli, Ed. : Springer, 2002, vol. 2396, Lecture Notes in Computer Science, pp. 15–30.

[2] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[3] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proc. Conf. North Amer. Chap. Assoc. Comput. Linguist. Human Lang. Technol. (NAACL/HLT)*, 2003, pp. 134–144.

[4] C. Sutton and A. McCallum, "Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data," *J. Mach. Learn. Res.*, vol. 8, no. 1, pp. 693–723, Mar. 2007.

[5] F. Sha and L. K. Saul, "Large margin hidden Markov models for automatic speech recognition," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2007, pp. 1249–1256.

[6] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," *Comput. Linguist.*, vol. 26, no. 3, pp. 339–373, 2000.

[7] S. Wang, A. Quattoni, L. P. Morency, D. Demirdjian, and T. Darrell, "Hidden conditional random fields for gesture recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, 2006, vol. 2, pp. 1521–1527.

[8] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, "Hidden conditional random fields for phone classification," in *Proc. Interspeech 2005-Eurospeech*, 2005, pp. 1117–1120.

[9] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 1, pp. 72–83, Jan. 1995.

[10] N. Gupta, G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi, and M. Gilbert, "The AT&T spoken language understanding system," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 213–222, Jan. 2006.

[11] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel, "A novel use of statistical parsing to extract information from text," in *Proc. Conf. North Amer. Chap. Assoc. Comput. Linguist. (NACACL)*, 2000, pp. 226–233.

[12] K. Toutanova, A. Haghighi, and C. D. Manning, "Joint learning improves semantic role labeling," in *Proc. Conf. Assoc. Comput. Linguist. (ACL)*, Ann Arbor, MI, Jun. 2005, pp. 589–596.

[13] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Comput. Speech Lang.*, vol. 16, no. 1, pp. 69–88, 2002.

[14] M. Jeong and G. G. Lee, "Jointly predicting dialog act and named entity for spoken language understanding," in *Proc. IEEE/ACL 2006 Workshop Spoken Lang. Technol.*, Dec. 2006, pp. 66–69.

[15] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2001, pp. 282–289.

[16] 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: MIT Press, 2006.

[17] L. E. Baume, T. Peterie, G. Souled, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Statist.*, vol. 41, no. 1, pp. 164–171, 1970.

[18] A. J. Viterbi, "Error bounds for convolutional codes and asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–267, Apr. 1967.

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

[20] F. Peng and A. McCallum, "Accurate information extraction from research papers using conditional random fields," in *Proc. Conf. North Amer. Chap. Assoc. Comput. Linguist. Human Lang. Technol. (NAACL/HLT)*, 2004, pp. 329–336.

[21] R. McDonald and F. Pereira, "Identifying gene and protein mentions in text using conditional random fields," *BMC Bioinformatics*, vol. 6, no. Suppl 1, May 2005.

[22] M. Jeong and G. G. Lee, "Practical use of non-local features for statistical spoken language understanding," *Comput. Speech Lang.*, vol. 22, no. 2, pp. 148–170, Apr. 2008.

[23] A. Quattoni, S. Wang, L. P. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1848–1852, Oct. 2007.

[24] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, Feb. 2001.

[25] T. Cohn, "Efficient inference in large conditional random fields," in *Proc. Eur. Conf. Mach. Learn. (ECML)*, 2006, pp. 606–613.

[26] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 1999.

[27] J. Besag, "Statistical analysis of non-lattice data," *The Statistician*, vol. 24, pp. 179–195, 1975.

[28] H. Daumé, III and D. Marcu, "Domain adaptation for statistical classifiers," *J. Artif. Intell. Res.*, vol. 26, pp. 101–126, 2006.

[29] M. Jordan and R. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, no. 1, pp. 181–214, 1994.

[30] L.-P. Morency, C. Sidner, C. Lee, and T. Darrell, "The role of context in head gesture recognition," in *Proc. 21st National Conf. Artif. Intell. (AAAI)*, 2006, pp. 1650–1653.

[31] R. McDonald, K. Hannan, T. Neylon, M. Wells, and J. Reynar, "Structured models for fine-to-coarse sentiment analysis," in *Proc. Conf. Assoc. Comput. Linguist. (ACL)*, Prague, Czech Republic, Jun. 2007, pp. 432–439.

[32] Y. Wang, L. Deng, and A. Acero, "Spoken language understanding: An introduction to the statistical framework," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 16–31, Sep. 2005.

[33] Y. He and S. Young, "Semantic processing using the hidden vector state model," *Comput. Speech Lang.*, vol. 19, no. 1, pp. 85–106, Jan. 2005.

[34] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," in *Proc. 3rd Workshop Very Large Corpora.*, 1995, pp. 82–94.

[35] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 1989, pp. 532–535.

[36] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. D. O. Odell, D. Povey, V. Valtchev, and P. Woodland, "The HTK Book: Version 3.3," Tech. Rep. Cambridge Univ., U.K., 2005.

[37] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, "Distributed representations," in *Parallel Distributed Processing*, L. Getoor and B. Taskar, Eds. Cambridge, MA: MIT Press, 1986, vol. 1, ch. 3, pp. 77–109.

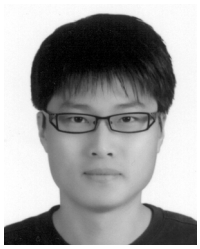
[38] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[39] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2003.

[40] C. Pal, C. Sutton, and A. McCallum, "Sparse forward–backward using minimum divergence beams for fast training of conditional random fields," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2006, pp. V-581–V-584.

[41] C. Sutton and A. McCallum, "Piecewise pseudolikelihood for efficient CRF training," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 863–870.

[42] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. Murphy, "Accelerated training of conditional random fields with stochastic meta-descent," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 969–976.



Minwoo Jeong received the B.S. degree in computer engineering from Chonbuk National University, Jeonju, Korea, in 2003 and the M.S. degree in computer science and engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 2005. He is currently pursuing the Ph.D. degree at POSTECH.

His research interests currently include language understanding, spoken dialog system, text mining, and practical machine learning methods for speech and language processing.

Gary Geunbae Lee (M'95) received the B.S. and M.S. degrees in Computer Engineering from Seoul National University, Seoul, Korea, in 1984 and 1986, respectively, and the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), in 1991

He was a Research Scientist with UCLA in 1991. He has been a Professor at in the Computer Science and Engineering Department, Pohang University of Science and Technology (POSTECH), Pohang, Korea, since 1991. He is a director of the Intelligent Software Laboratory which focuses on human language technology researches including natural language processing, speech recognition/synthesis, speech translation, question answering and web/text mining. He has authored more than 100 papers in international journals and conferences, and has served as a technical committee member and reviewer for several international conferences such as ACL, COLING, IJCAI, ACM SIGIR, AIRS, ACM IUI, Interspeech, EMNLP, and IJCNLP. He is currently leading several national and industry projects for robust spoken dialog systems, spoken dialog translation, and expressive TTS.