NAIST-IS-DD0561001

Doctoral Dissertation

A Study of Algebraic and Graph-Theoretic Frameworks for Generalized Forward-Backward Algorithms

Ai Azuma

March 15, 2013

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Ai Azuma

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Kiyohiro Shikano	(Co-supervisor)
Associate Professor Masashi Shimbo	(Co-supervisor)
Associate Professor Masayuki Asahara	(Co-supervisor, National Institute for Japanese Language and Linguistics)

A Study of Algebraic and Graph-Theoretic Frameworks for Generalized Forward-Backward Algorithms*

Ai Azuma

Abstract

Forward-backward algorithms are an important Dynamic Programming (D.P.) algorithm in structured prediction especially in sequence labeling. The algorithm enables us to avoid explicit enumeration of all the possible output candidates and provides an efficient way to accomplish computation tasks in both learning and prediction steps of sequence labeling.

In recent years, there is a series of studies in formalization of D.P. based on algebraic and graph-based frameworks. Such frameworks can subsume many important computations. In the context of sequence labeling, there are some studies on a unifying framework for Viterbi algorithms, which is vital in sequence labeling tasks. On the other hand, Forward-Backward algorithms are another crucial algorithm in sequence labeling. A key point of Forward-Backward algorithms lies in combination of D.P. in both forward and backward manners, while Viterbi algorithms are D.P. in single direction only. Such combination of D.P. is outside the scope of the unifying framework for Viterbi algorithms. As a consequence, there is no adequate framework for generalized Forward-Backward algorithms.

In this dissertation, I first introduce a unifying framework for generalized Forward-Backward algorithms based on algebraic and graph-theoretic formalization. Next, I will introduce some systematic ways to compose complex algebraic structures from simpler ones. Given this framework, I also show novel instances of generalized Forward-Backward algorithms. They show promise in various numerical computation in sequence labeling tasks. Finally, in order to demonstrate

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561001, March 15, 2013.

the actual benefit of the framework proposed in this dissertation, I propose a novel sequence labeling model which utilizes one of proposed instances of the framework in its learning step. I also show the quantitative result of the model.

Keywords:

forward-backward algorithm, dynamic programming, machine learning, structured prediction

一般化前向き後ろ向きアルゴリズムに対する代数的・グラフ理論的枠組みの研究*

東 藍

内容梗概

前向き後ろ向きアルゴリズムは構造学習のうち特に系列ラベリングにおいて重要な動的計画法によるアルゴリズムである．このアルゴリズムにより，可能なすべての出力候補を陽に列挙することなく，系列ラベリングの学習および予測の過程における計算タスクを効率的に行うことができる．

近年，動的計画法を代数的かつグラフ理論に基づく枠組みで形式化しようとする一連の研究がある．多くの重要な計算がこのような枠組みに包括される．系列ラベリングの文脈においては，系列ラベリングにおいて重要なヴィタビアルゴリズムに対する統一的枠組みに関する研究がある．一方で，前向き後ろ向きアルゴリズムもまた系列ラベリングにおいて重要なアルゴリズムである．前向き後ろ向きアルゴリズムの重要な点は，前向きの動的計画法と後ろ向きの動的計画法を組み合わせるところにある．これは一方向のみの動的計画法であるヴィタビアルゴリズムとは異なる．このような動的計画法の組み合わせはヴィタビアルゴリズムに対する統一的枠組みの中には納まらない．結果として，前向き後ろ向きアルゴリズムに対する十分な枠組みは未だにないと言える．

本稿ではまず，代数的かつグラフ理論による形式化に基づいて，一般化された前向き後ろ向きアルゴリズムに対する統一された枠組みを導入する．次に，複雑な代数構造をより簡易な代数構造から体系的に構成する方法を導入する．この枠組みを前提として，一般化された前向き後ろ向きアルゴリズムの新規の事例をいくつか導出する．これらの事例は系列ラベリングのタスクにおける様々な数値計算において有望である．最後に，本稿において提案する枠組みがもたらす実際的な恩恵を実証するために，新規の系列ラベリングモデルを提案する．このモデルの学習の過程において本稿で提案する枠組みの実例の一つが活用される．さらにこのモデルの定量的評価についても述べる．

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文，NAIST-IS-DD0561001, 2013年 3月 15日.

キーワード

前向き後ろ向きアルゴリズム, 動的計画法, 機械学習, 構造学習

Acknowledgments

主指導教官の松本裕治教授に深く感謝いたします．松本先生には，研究する上での様々な助言はもとより，研究する上でのペースやスタンスなどといった研究者としての在り方に関するアドバイスや，さらには精神面においても多くの助言と支えをいただきました．松本先生からいただいた一言一言が今日の私を支えていると言っても過言ではありません．松本研究室で研究することができたという経験は私の人生において最も幸運な出来事の一つであったと思います．

副指導教官の鹿野清宏教授にはお忙しい中審査委員をお引き受けくださり本当にありがとうございました．発表において掛けていただいた数々の言葉を励みとして博士論文の執筆を継続することができました．

副指導教官の新保仁准教授には常日頃から有益な指導の数々をしていただき，深く感謝いたします．研究に関するコメントは常に厳しいものでしたが，一方で研究の合間には雑多な話を親しくさせていただくことができ，それらを通じて研究に関わる重要な示唆をいくつもいただきました．

浅原正幸国立国語研究所特任准教授にはお忙しい中審査委員をお引き受けくださり本当にありがとうございました．松本研究室を初めて訪れた時から，様々なアドバイスをくださる先輩として，また研究に関する有益なコメントをいくつもいただいたスタッフとして，常日頃からお世話になりました．

松本研究室助教の小町守先生には大変お世話になりました．研究に関する様々な助言や示唆をいただくとともに，小町先生の広い交流範囲を通して，企業を始めとする現場における自然言語処理に携わる方々と接する機会を持つことができ，応用の側からの動機づけに触れ，自身の研究の刺激となりました．

松本研究室元秘書の大川元子さん，現秘書の北川祐子さんには研究活動における雑多の手続きを一手に引き受けてくださり，感謝の気持ちでいっぱいです．また，研究と離れたところでは常に日常の雑多な会話をしてくださり研究に憩いを与えてくださいました．

松本研究室の学生さんには，様々な刺激を与えてもらいました．特に長年スーパーバイザーを勤めてきた離散数学勉強会では，鋭い学生さんから様々な疑問を呈され，それらに真摯に答えて行こうとする中で定式化というものの意義と重要

性を再認識させられました．この体験が本博士論文の重要な部分として活かされているように思います．

最後に母，君江に感謝します．彼女は私に「知る」ということの本質的価値を教えてくださった人です．母は，知ることそれ自身が価値であるということを，教育を通して教えてくださいました．今，私が知ろうとし続ける姿勢を持てるのも母の教育があったからにほかなりません．

最後に今一度，この場を借りて，この博士論文執筆に関わってくださったすべての皆様に厚く御礼申し上げます．本当にありがとうございました．

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Many learning tasks in the real world involve complex structures in which there are multiple interrelated labels to be assigned. Structures can be classified roughly according to their complexity, and the classes include arbitrary graphs, trees and so on. A simple but notable class that involves structures is sequence labeling, in which dependencies between labels constitute a linear chain. A lot of classification or pattern recognition tasks on natural language sentences, genes and the like can be modeled as sequence labeling.

Sequence labeling involves computation over all candidate output sequences. For example, the expectation-maximization (EM) algorithm for Hidden Markov Models (HMMs) [5] requires expected frequencies of labels and adjacent pairs of labels measured on the current model. Computation of the gradient of the log-likelihood objective for Conditional Random Fields (CRFs)[16] also involves model expectations of features or adjacent pairs of features. It is also important to sum up the joint probabilities or weights over all the output sequence candidates. Such a summation results in the language model $\sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x})$ for generative models like HMMs, where $\mathbf{x}$ is an input sequence and $\mathbf{y}$ a candidate of the output sequences. For the case of CRFs, such a summation results in the normalization constant, also known as the partition function, of the input sequence. In the prediction step of sequence labeling using HMMs or CRFs, the sequence that maximizes some joint score has to be found among the output sequence candidates. These computations are originally defined as summation or maximum operation on all the possible candidates of output sequences for a given input.

However, naive computation over all the possible output sequences is not feasible for practical computing purposes because the number of the possible output sequences is typically proportional to the exponential of a natural measure of the size of the problem, e.g., the length of the input sequence. Therefore, clever and efficient procedures are needed to handle the above computations.

The aforementioned computation can be efficiently handled by Dynamic Programming (D.P.). For the sequence labeling, such D.P. is often referred to as "Viterbi algorithms" and "Forward-Backward algorithms" (henceforth abbreviated as "F.B. algorithms," or simply "F.B."). Viterbi algorithms are used to find the most likely output sequence among candidates. Viterbi algorithms also compute the language model for the case of HMMs and normalization constants for the case of CRFs. Viterbi algorithms also can be used to find the best candidate in the prediction step of HMMs and CRFs. Here, what I would like readers to keep in mind is that Viterbi algorithms are D.P. in single direction only.

F.B. algorithms are crucial to parameter estimation of many probabilistic models used in sequence labeling tasks. The original F.B. algorithm was developed to compute local maximization of the log-likelihood function for discrete-time finite-state HMMs. However, many variants of F.B. algorithms were studied in recent years. They originated out of the need to optimize the model parameter in terms of an objective other than the simple log-likelihood, to compute the value or gradient of complex regularization terms appearing in the objective, to handle probabilistic models for sequence labeling other than HMMs, and so on. Therefore, the current implication of the term "Forward-Backward algorithms" has become much broader in scope than the original one. For example, the variant of F.B. used in parameter optimization of CRFs is quite analogous to the original F.B., so it is often simply called "Forward-Backward algorithm." In addition, other variants of F.B. are sure to be developed as required. As a consequence, the scope of the term "Forward-Backward algorithms" keeps growing to include these variants.

In the past, variants of F.B. were developed and formalized in an ad hoc way that is applied only to problems at hand. As a result, similar formalization that would be unified into single generalized formalization has been carried out from problem to problem. Examples of such ad hoc derivation include Hessian-vector products of CRFs[27][26], computation of entropy gradient for CRFs [13][18].

Here, a motivation arises. Variants of F.B. have similarities, and their com-

mon parts can be factored out in a formal way. One of the main goals in this dissertation is to propose a unified formalization of generalized F.B. algorithms. It is desirable that the formalization subsumes as many known variants of F.B. algorithms as possible. Or conversely, it could define a clear scope of the term "Forward-Backward algorithms."

There are some unifying frameworks for D.P. [8][21][9][10] They all utilize algebraic structures and offer a unified view of D.P. in which recursion steps proceed in one direction only. In the context of sequence labeling, they reduce to the algebraic frameworks for Viterbi algorithms. The frameworks subsume variants of Viterbi algorithms, whose examples include the computation of the language model of HMMs, the normalization constant of CRFs, the path with the maximum score among the output candidates, and so on.

However, far too little attention has been paid to algebraic frameworks for F.B. algorithms. And I would like to argue that some kinds of limitations in the existing framework for Viterbi algorithms rise to the surface when trying to offer a unifying framework for known variants of F.B. A key feature of F.B. algorithms lies in "weighted combination" of D.P. in both forward and backward manners, while Viterbi algorithms are D.P. in one direction only. The part of "weighted combination" in F.B. cannot be fully formalized only in terms of the algebraic structures used in the framework for Viterbi. Thus, there are issues to address. In order to precisely grasp the key part of F.B. in terms of algebraic structures, some further concepts are required in addition to the ones used in the existing frameworks for Viterbi.

I would also like to argue that there is another kind of practical issues in algebraic frameworks. Discussions based on algebraic frameworks tend to be occupied with just a catalog of algebras. It is important to offer an unified point of view on different computations. However, it seems much more desirable to offer a systematic approach to construct a new algebra. Once we have such approach, we obtain a systematic approach to derive new variants of Viterbi and F.B. algorithms.

A common way of formalizing complex objects in mathematics is to provide a way to induce complex objects from simpler ones. In this dissertation, I will show some systematic ways to induce complicated semirings from simpler ones. In that way, we can induce very complex semirings one after another. And we can easily confirm that complicated variants of F.B. algorithms named above are

subsumed by semirings induced in this way.

1.2 Thesis Outline and Contributions

This dissertation addresses the issues stated in the previous section, and makes the following contributions.

1. In Chapter 2, I will first introduce algebraic and graph-theoretic concepts. And on that basis I will formalize a unifying framework of generalized Forward-Backward algorithms.

2. In Chapter 3, I will develop novel instances for the generic framework introduced in Chapter 2. The instances includes sequences of a semiring, polynomial sequences, the complex numbers, and outer products of them. It is also demonstrated how existing algorithms are subsumed by the algebraic framework. This Section also discusses further possibilities of the instances introduced in Section 3.

3. In Chapter 4, I propose a novel model for cascaded sequence labeling. The optimization procedure of the proposed model requires generalized F.B. and Viterbi algorithms for complicated calculations. Instances of the generic framework for Forward-Backward algorithm developed in Section 3 is effectively exploited in the calculations.

Chapter 2

Formalization of Forward-Backward Algorithms

In this chapter, first I will introduce some definitions in graph theory in order to formalize "trellis." Second, I will briefly review some definitions and properties of some algebraic concepts. Finally, a framework for generalized F.B. algorithms will be presented.

2.1 Formalization of Trellis

F.B. algorithms can be considered to compute "summation" of "score defined on sequences." The set of sequences has to be represented by a remarkably compact data structure. Conceptually, the set is encoded by a directed acyclic graph (DAG). A sequence is represented by a directed path in the DAG. It defines what is called *trellis*. It is a part of the basis for the framework for generalized F.B. shown in later. I will introduce some formal notations about DAGs.

Let me introduce some terminology. Consider a DAG $G = (V, E)$. DAGs in this dissertation are assumed to be linear, that is, there is no more than one arc between a pair of nodes. *The predecessor set* of node $v \in V$ is denoted by $\mathrm{N}^-(v)$, i.e., $\mathrm{N}^-(v) \stackrel{\mathrm{def}}{\equiv} \{x \in V | (x, v) \in E\}$. *The successor set* of node $v \in V$ is denoted by $\mathrm{N}^+(v)$, i.e., $\mathrm{N}^+(v) \stackrel{\mathrm{def}}{\equiv} \{x \in V | (v, x) \in E\}$. *A source node* is a node whose in-degree is zero. *A sink node* is a node whose out-degree is zero. Hereafter, it is assumed that G has only one source node and one sink node. They are denoted by $src(G)$ and $snk(G)$, respectively. They are abbreviated to src and

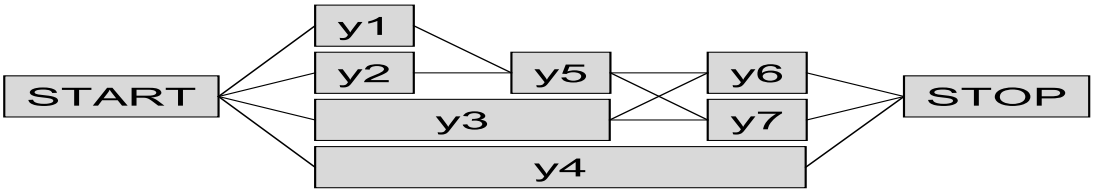

Figure 2.1: An example trellis

snk respectively if G is obvious from the context. Note that it is straightforward to extend the following discussions to DAGs with multiple source and sink nodes. Therefore, I omit the discussions for the case where there are multiple sources or sinks. Let $\Psi(u, v)$ be the set of directed paths whose starting node is $u \in V$ and end node is $v \in V$. Because a directed path in a linear DAG can be uniquely identified by the set of nodes that constitutes the path, a path is also referred to as the set of the nodes. Thus, for example, the predicate $v \in \psi$ stands for a relation whether v is in ψ, where $v \in V$ and ψ is a directed path in G. For $u, v, w \in V$, $\Psi_{\sim w}(u, v)$ is defined to denote the subset of $\Psi(u, v)$ constrained on the node w. More formally, $\Psi_{\sim w}(u, v) \stackrel{\text{def}}{\equiv} \{x \in \Psi(u, v) \mid w \in x\}$. A path ψ in G is *successful* if and only if the starting node is src and end node is snk. The set of all successful paths in G is denoted by $\Psi(G)$, that is, $\Psi(G) \stackrel{\text{def}}{\equiv} \Psi(src, snk)$.

For example, the DAG shown in Fig.2.2.1 has a source labeled "START" and a sink labeled "STOP". For that DAG, $\Psi(\text{START}, \text{STOP})$ includes seven directed paths, while $\Psi_{\sim \text{y5}}(\text{START}, \text{STOP})$ includes four paths {START, y1, y5, y6, STOP}, {START, y1, y5, y7, STOP}, {START, y2, y5, y6, STOP} and {START, y2, y5, y7, STOP}. The path {START, y3, y7, STOP} is successful, while the path {START, y3, y7} is not.

Weights, features, or some other values may be associated to nodes. They can be formally defined by functions with the domain of V. Values may also appear on arcs in addition to nodes in many practical applications. However, for the sake of simplicity, arcs are considered to just represent transitions between nodes, and no value is associated to arcs. Because there is a systematic way to transform $G = (V, E)$ into another DAG $G' = (V', E')$ such that functions with the domain of $V \cup E$ can be expressed in terms of functions with the domain of V'. The method is defining $G' = (V', E')$ with $V' \stackrel{\text{def}}{\equiv} V \cup E$ and $E' \stackrel{\text{def}}{\equiv} \{(x, y) \in V' \times V' \mid u \in V \wedge (u, v) \in E \vee (u, v) \in E \wedge v \in V\}$. Under this transformation, nodes and

arcs in G are transformed into nodes in G', and adjacency between nodes and arcs in G is replaced by arcs in G'. Henceforth, it is assumed that functions are defined only on V without loss of generality.

2.2 Algebraic Foundations

Next, I will briefly review some definitions and properties used in later sections and chapters. The notations shown in this section follow Kuich and Salomaa [15], Mohri [21], Huang [10] and Li and Eisner [17].

Before introducing algebraic structures used later in this dissertation, let me first explain why algebraic structures defined in this section is required to offer a unifying framework for generalized F.B. algorithms.

In the ordinary F.B. algorithm that is used to compute model expectation of symbol emission or state transitions in HMMs, forward and backward variables are the real numbers. The combination of forward and backward variables on each node results in the model expectation of a state. Finally, a weighted summation over the model expectation of states is carried out to compute the expectation of symbol emission, in which the weight on each node is emission probability of a symbol, that is, the real number. In short, all of forward and backward variables and weights on nodes are the real numbers in the ordinary F.B.

On the other hand, some variants of F.B. algorithms that were developed in recent years involve forward and backward variables of a pair of the real numbers. Examples include F.B. algorithms with the expectation semiring[6][17] and with the entropy semiring[4][11]. They can be viewed as generalized F.B. equipped with semirings over the two-dimensional real vectors. The addition of the semirings is the ordinary vector addition, that is, the element-wise addition. The multiplication of the semirings is defined by the convolution between two-dimensional vectors. D.P. with these semirings in single (either forward or backward) direction only is an instance of the existing frameworks for generalized Viterbi. However, a key operation of F.B. is the "weighted combination" of D.P. in both forward and backward manner. "Combination" is represented by the multiplication of semirings. But, semirings are not good enough to formalize "weighting operation" because "weights" may come from a domain other than the one of forward and backward variables. In fact, in F.B. with the expectation or entropy semirings, forward and backward variables are two-dimensional real vectors while weight

on each node is the real number. Thus, the weighting operation in F.B. with the expectation or entropy semirings is the scalar multiplication between the real numbers and two-dimensional real vectors. Therefore, in order to offer a unifying framework for generalized F.B. that subsumes F.B. with the expectation or entropy semirings, such framework should be based on algebraic structures that capture "multiplication between different domains."

In this dissertation, I will build the framework for generalized F.B. shown in Section 2.3 on a semiring and a unital associative algebra over a commutative semiring.

Definition 1. *A monoid is a system* $(S, +, 0)$ *such that;*

1. $+$ *is a closed binary operator on the set* S*, i.e.,* $+ : S \times S \to S$*.*
2. $+$ *is associative, i.e.,* $\forall a, b, c \in S,\ \ (a + b) + c = a + (b + c)$*.*
3. S *has the identity element for* $+$*, i.e.,* $\exists 0 \in S$ s.t. $s + 0 = 0 + s = s$*.*

Definition 2. *A commutative monoid is a monoid* $(S, +, 0)$ *whose operation is commutative, i.e.,* $\forall a, b \in S,\ \ a + b = b + a$*.*

Definition 3. *A semiring is a system* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ *such that;*

1. $(\mathbb{K}, \oplus, \bar{0})$ *is a commutative monoid.*
2. $(\mathbb{K}, \otimes, \bar{1})$ *is a monoid.*
3. $\otimes$ *distributes over* $\oplus$*, i.e.,* $\forall a, b, c \in \mathbb{K}$*,*
$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)\ \ ,$$
$$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)\ \ .$$
4. $\bar{0}$ *is the annihilator,*
$$\forall a \in \mathbb{K},\ \ \bar{0} \otimes a = a \otimes \bar{0} = \bar{0}\ \ . \tag{2.1}$$

For the sake of simplicity, I will often denote a semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ as simply $\mathbb{K}$ unless it is confusing.

For a semiring $\mathbb{K}$ and a natural number $n \in \mathbb{N}_0$, n-th power of $x \in \mathbb{K}$ is defined as follows,

$$x^n \stackrel{\text{def}}{\equiv} \begin{cases} \underbrace{x \otimes \cdots \otimes x}_{n\text{-times}}, & n \geq 1\ \ , \\ \bar{1}, & n = 0\ \ . \end{cases} \tag{2.2}$$

Definition 4. *A commutative semiring is a semiring* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ *such that* $\otimes$ *is commutative, i.e.,*

$$\forall a, b \in \mathbb{K}, \;\; a \otimes b = b \otimes a \;\; .$$

Definition 5. *For a given semiring* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$*, a semimodule over the semiring* $\mathbb{K}$ *consists of a commutative monoid* $(M, +, 0)$ *and an operation* $\mathbb{K} \times M \to M$ *such that* $\forall x, y \in \mathbb{K}$ *and* $m, n \in M$*,*

$$x \cdot (m + n) = x \cdot m + x \cdot n \;\; , \tag{2.3}$$

$$(x \oplus y) \cdot m = x \cdot m + y \cdot m \;\; , \tag{2.4}$$

$$(x \otimes y) \cdot m = x \cdot (y \cdot m) \;\; . \tag{2.5}$$

Note that there is a distinction between *left* and *right* semimodule, which differ in whether the operation between $\mathbb{K}$ and M is defined by $\mathbb{K} \times M \to M$ or $M \times \mathbb{K} \to M$. However, it is not significant in this dissertation, and they are always assumed to left ones, as defined above.

A semimodule is a generalization of vector spaces. Elements in M in Definition 5 correspond to "vectors," so $+$ means "vector addition." Elements in $\mathbb{K}$ correspond to "scalars," so $\oplus$ and $\otimes$ are addition and multiplication between scalars, respectively. And $\cdot : \mathbb{K} \times M \to M$ corresponds the multiplication between a scalar and a vector, that is, "scalar multiplication." In fact, a vector space over a field K is always a semimodule over K.

Semimodules are not yet enough to offer a framework for generalized F.B. Recall the discussions in the very beginning of this section. A semimodule has scalars, which represent node weights, vectors, which represents forward and backward variables, and scalar multiplication, which represents weighting operation. However, vectors do not have multiplication, so M in Definition 5 is not a semiring. Therefore, in order for semimodules to be computed in D.P., multiplication between vectors should be introduced.

Definition 6. *Let* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ *a commutative semiring. A unital associative algebra over a commutative semiring* $\mathbb{K}$ *is a semiring* $(A, +, \times, 0, 1)$ *which is also a semimodule over* $\mathbb{K}$ *such that the multiplication* $\times : A \times A \to A$ *is bilinear, that is,*

$$\forall a \in \mathbb{K}, \;\; \forall x, y \in A, \;\; a \cdot (x \times y) = (a \cdot x) \times y = x \times (a \cdot y) \;\; . \tag{2.6}$$

Roughly speaking, a unital associative algebra over a commutative semiring consists of two semirings and multiplication between elements of one semiring and the other.

Fact 1. *A semiring* $(A, +, \times, 0, 1)$ *is a unital associative algebra over* $\mathbb{N}_0$ *with the following definition.*

$$n \cdot x \overset{def}{\equiv} \underbrace{x + \cdots + x}_{n-times}, \quad n \in \mathbb{N}_0, \quad x \in A \ . \tag{2.7}$$

2.3 Formalization of Generalized Forward-Backward Algorithms

In this section, I will present a formalization of generalized F.B. algorithms. I will also provide the proof of a key property that makes F.B. algorithms beneficial in many real-world problems. In order to write down definitions and proofs, let me introduce some additional definitions.

In the following definitions, let $(\mathbb{K}, \otimes, \oplus, \bar{0}, \bar{1})$ denote a commutative semiring, $(A, +, \times, 0, 1)$ denote a unital associative algebra over $\mathbb{K}$. Note that $(A, +, \times, 0, 1)$ itself is also a semiring. A function $\phi : V \to A$ is defined on nodes of a trellis $G = (V, E)$. The value of ϕ on $v \in V$ is denoted by $\phi(v)$ and called the *weight* or *potential* of node v. In addition, the definition of weight is extended to paths in G. The weight of a path $\psi = \{v_{\psi_1}, \ldots, v_{\psi_{|\psi|}}\}$ is defined by $\phi(\psi) \overset{\text{def}}{\equiv} \prod_{v \in \psi} \phi(v) = \phi(v_{\psi_1}) \times \cdots \times \phi(v_{\psi_{|\psi|}})$. Then, consider the following form of the sum.

Definition 7. *For given* $u, v \in V$*, the shortest distance between* u, v*, denoted by* $\mathcal{D}_{A,w}(u, v)$*, is defined as follows,*

$$\mathcal{D}_{A,\phi}(u, v) \overset{def}{\equiv} \sum_{\psi \in \Psi(u,v)} \phi(\psi) \ . \tag{2.8}$$

Note that the definition of $\Psi(u, v)$ appeared in Section 2.2.1.

Some readers may feel uncomfortable with the term "shortest distance." This terminology is originally attributed to Mohri[21]. When A in this definition is substituted by $(\mathbb{R}, \min, +, 0, 1)$, which is so-called *tropical semiring*, the "sum" of (2.8) is equal to the shortest distance between the nodes u, v in the trellis G with the real-valued node weight w. Therefore, we can easily confirm that this definition is certainly a generalization of "shortest distance."

Based on the definitions described above, a formalization of generalized Viterbi algorithms is described in Algorithm 1.

Algorithm 1 Generalized Viterbi Algorithms

Input: A trellis $G = (V, E)$, a semiring $(A, +, \times, \mathbf{0}, \mathbf{1})$, node weight $\phi : V \to A$

Output: $\alpha(v) = \mathcal{D}_{A,\phi}(src, v)$, for $\forall v \in V$

1: $\alpha(src) \leftarrow \phi(src)$
2: **for all** $v \in V \setminus src$ in a topological order **do**
3: $\quad \alpha(v) \leftarrow \sum_{x \in \mathrm{N}^-(v)} \alpha(x) \times \phi(v)$
4: **end for**

The correctness of Algorithm 1 is explained as below. First of all, $\alpha(src) = \mathcal{D}_{A,\phi}(src, src)$ is obvious from the definition. In what follows, for a node $x \in V$, assuming

$$\alpha(v) = \mathcal{D}_{A,\phi}(src, v) \tag{2.9}$$

holds on $\forall v \in \mathrm{N}^-(x)$, then,

$$\begin{aligned}
\alpha(x) &= \sum_{v \in \mathrm{N}^-(x)} \alpha(v) \times \phi(x) \\
&\qquad (\because \text{line 3 in Algorithm 1}) \\
&= \sum_{v \in \mathrm{N}^-(x)} \mathcal{D}_{A,\phi}(src, v) \times \phi(x) \\
&\qquad (\because \text{the assumption (2.9)}) \\
&= \sum_{v \in \mathrm{N}^-(x)} \left(\sum_{\psi \in \Psi(src,v)} \phi(\psi) \right) \times \phi(x) \\
&\qquad (\because \text{the definition of } \mathcal{D}_{A,\phi}(src, v)) \\
&= \sum_{v \in \mathrm{N}^-(x)} \sum_{\psi \in \Psi(src,v)} (\phi(\psi) \times \phi(x)) \\
&\qquad (\because \text{distributivity}) \\
&= \sum_{\psi \in \Psi(src,x)} \phi(\psi) \\
&\quad (\because \text{collects two summations and weight}) \\
&= \mathcal{D}_{A,\phi}(src, x) \ .
\end{aligned} \tag{2.10}$$

Therefore, (2.9) holds for $\forall v \in V$ because α is recursively calculated in a topological order of G.

From Algorithm 1, a Viterbi algorithm is specified by a trellis G, a semiring $(A, +, \times, 0, 1)$ and a weight function ϕ. Hereinafter, the tuple $(G, (A, +, \times, 0, 1), \phi)$ is called the *specification* of a Viterbi algorithm. And $\alpha(snk) = \mathcal{D}_{A,\phi}(src, snk)$ is called the *result* of the Viterbi algorithm.

Note that Algorithm 1 itself is not new because it is just an adaptation of Huang[10]. However, it constitutes one half of the formalization of generalized Forward-Backward algorithms, which is a new proposal in this dissertation.

In addition to the definitions introduced in the very beginning of this section, *a feature* $f : V \to \mathbb{K}$ is defined on nodes. Given the aforementioned definitions, the following two kinds of summation are also defined.

Definition 8. *The marginalization of the weight ϕ with respect to a given node $v \in V$, denoted by $\mathcal{M}_{A,\phi}(v)$, is defined as follows,*

$$\mathcal{M}_{A,\phi}(v) \stackrel{def}{\equiv} \sum_{\psi \in \Psi_{\sim v}(src, snk)} \phi(\psi) \ . \tag{2.11}$$

Note that the definition of $\Psi_{\sim v}(src, snk)$ appeared in Section 2.2.1.

Let me explain the concrete interpretation of Definition 8 in HMMs. In finite-state discrete-time HMMs, a node $v \in V$ represents a state or state transition, and a successful path corresponds to a sequence of state transitions. Therefore, $\Psi_{\sim v}(src, snk)$ is the set of transition sequences consistent to the state or state transition represented by v. When ϕ is defined by the transition probability (and $\phi(v) = 1$ if v represents a state), then, $\mathcal{M}_{A,\phi}(v)$ is equal to the marginal probability of the state or state transition represented by v.

Definition 9. *The expectation of the feature f with respect to the weight ϕ, denoted by $\mathcal{E}_{A,\mathbb{K},\phi}(f)$, is defined as follows,*

$$\mathcal{E}_{A,\mathbb{K},\phi}(f) \stackrel{def}{\equiv} \sum_{\psi \in \Psi(G)} \left(\left(\bigoplus_{v \in \psi} f(v) \right) \cdot \phi(\psi) \right) \ . \tag{2.12}$$

Again, let me explain the concrete interpretation of Definition 9 in HMMs. When ϕ is defined by the transition probability and f by the emission probability of a symbol (and $f(v) = 0$ if v represents a state transition), then, $(\oplus_{v \in \psi} f(v)) \cdot \phi(\psi)$ is equal to the probability of the symbol emission in the transition sequence represented by the path ψ. Therefore $\mathcal{E}_{A,\mathbb{K},\phi}(f)$ is equal to the expectation of the symbol emission.

Here, I would like to address an important issue. For the example of HMMs, the product of the marginal probability and the symbol emission probability on a node $v \in V$ is equal to the probability of the symbol emission at the state represented by v. The summation of such probability over all node is equal to the symbol emission probability. That is, for the example of the symbols emission probability in HMMs, the following equation holds,

$$\sum_{v \in V} (f(v) \cdot \mathcal{M}_{A,\phi}(v)) = \mathcal{E}_{A,\mathbb{K},\phi}(f) \ . \tag{2.13}$$

F.B. actually computes the form in the left hand of (2.13). On the other hand, many values of interest are originally defined by the form in the right hand of (2.13). So, in order for F.B. to compute a value of interest, we often need to prove an equation like (2.13). For the case of HMMs, this equation can be easily proved in terms of the conditional independence among states and state transitions. However, the goal of this dissertation is to offer a unifying framework. The framework should subsume F.B. in which weights are other than probability. Therefore, additional assumptions such as conditional independence among nodes should be avoided, and the following claim has to be proved only in terms of algebraic structures.

Theorem 1 (Marginalization-Expectation)**.** *The following equation holds,*

$$\sum_{v \in V} (f(v) \cdot \mathcal{M}_{A,\phi}(v)) = \mathcal{E}_{A,\mathbb{K},\phi}(f) \ . \tag{2.14}$$

Proof of Theorem 1

First, the Kronecker delta for a semiring $(A, +, \times, \mathbf{0}, \mathbf{1})$ is defined by

$$\delta_P^A \stackrel{\text{def}}{\equiv} \begin{cases} \mathbf{1}, & \text{the predicate } P \text{ is true} \ , \\ \mathbf{0}, & \text{otherwise} \ . \end{cases} \tag{2.15}$$

Then,

$$
\begin{aligned}
\sum_{v\in V}(f(v)\cdot\mathcal{M}_{A,\phi}(v)) &= \sum_{v\in V}\left(f(v)\cdot\left(\sum_{\psi\in\Psi_{\sim v}(src,snk)}\phi(\psi)\right)\right)\\
&= \sum_{v\in V}\left(f(v)\cdot\left(\sum_{\psi\in\Psi}\delta^A_{v\in\psi}\times\phi(\psi)\right)\right)\\
&\quad\left(\text{note that}\sum_{\psi\in\Psi_{\sim v}(src,snk)}\bullet=\sum_{\psi\in\Psi}(\delta^A_{v\in\psi}\times\bullet)\right)\\
&= \sum_{v\in V}\sum_{\psi\in\Psi}\left(f(v)\cdot\left(\delta^A_{v\in\psi}\times\phi(\psi)\right)\right)\\
&\qquad\qquad(\because (2.3))\\
&= \sum_{\psi\in\Psi}\sum_{v\in V}\left(\delta^A_{v\in\psi}\times(f(v)\cdot\phi(\psi))\right) \qquad (2.16)\\
&\qquad\qquad(\because \text{ bilinearity of } \times \text{ over } \cdot)\\
&= \sum_{\psi\in\Psi}\sum_{v\in\psi}(f(v)\cdot\phi(\psi))\\
&\quad\left(\text{note that}\sum_{v\in V}\left(\delta^A_{v\in\psi}\times\bullet\right)=\sum_{v\in\psi}\bullet\right)\\
&= \sum_{\psi\in\Psi}\left(\left(\sum_{v\in\psi}f(v)\right)\cdot\phi(\psi)\right)\\
&\qquad\qquad(\because (2.4))\\
&= \mathcal{E}_{A,\mathbb{K},\phi}(f)\ . \qquad\square
\end{aligned}
$$

Based on this definition, the formalization of generalized Forward-Backward algorithms is described in Algorithm 2.

Next, I will show the correctness of Algorithm 2. First, the following equation can be verified in the same way as (2.9) and (2.10),

$$\beta(v)=\sum_{x\in\mathrm{N}^+(v)}\mathcal{D}_{A,\phi}(x,snk)\ . \qquad (2.17)$$

From the lines 5–8 in Algorithm 2, we can easily confirm

$$E=\sum_{v\in V}(f(v)\cdot(\alpha(v)\times\beta(v)))\ . \qquad (2.18)$$

Algorithm 2 Generalized Forward-Backward Algorithms

Input: A trellis $G = (V, E)$, a commutative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$, a feature $f : V \to \mathbb{K}$, a unital associative algebra over $\mathbb{K}$ $(A, +, \times, \mathbf{0}, \mathbf{1})$, a weight $\phi : V \to A$, forward variables $\alpha : V \to A$ (calculated by Algorithm 1)

Output: $E = \mathcal{E}_{A,\mathbb{K},\phi}(f)$

1: $\beta(snk) \leftarrow \mathbf{1}$
2: **for all** $v \in V \setminus snk$ in a reversed topological order **do**
3: $\quad \beta(v) \leftarrow \sum_{x \in \mathrm{N}^+(v)} \phi(x) \times \beta(x)$
4: **end for**
5: $E \leftarrow \mathbf{0}$
6: **for all** $v \in V$ **do**
7: $\quad E \leftarrow E + (f(v) \cdot (\alpha(v) \times \beta(v)))$
8: **end for**

And

$$\begin{aligned}
\alpha(v) \times \beta(v) &= \mathcal{D}_{A,\phi}(src, v) \times \left(\sum_{x \in \mathrm{N}^+(v)} \mathcal{D}_{A,\phi}(x, snk) \right) \\
&= \left(\sum_{\psi \in \Psi(src, v)} \phi(\psi) \right) \times \left(\sum_{x \in \mathrm{N}^+(v)} \sum_{\psi \in \Psi(x, snk)} \phi(\psi) \right) \\
&= \sum_{\psi \in \Psi_{\sim v}(src, snk)} \phi(\psi) \\
&= \mathcal{M}_{A,\phi}(v) \ .
\end{aligned} \tag{2.19}$$

Therefore,

$$\begin{aligned}
E &= \sum_{v \in V} (f(v) \cdot \mathcal{M}_{A,\phi}(v)) \\
&\qquad\qquad (\because (2.18), (2.19)) \\
&= \mathcal{E}_{A,\mathbb{K},\phi}(f) \qquad\qquad (\because \text{ Theorem 1}) \ .
\end{aligned} \tag{2.20}$$

Assuming that $+$, $\times$, $\oplus$, $\otimes$ and $\cdot$ all terminate in a finite length of time, it also can be shown that Algorithm 2 terminates in a finite length of time since it traverses a finite number of nodes in a topological order.

From Algorithm 2, a Forward-Backward algorithm is specified by a trellis G, a commutative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$, a feature function $f : V \to \mathbb{K}$, a unital associative algebra over $\mathbb{K}$ $(A, +, \times, \mathbf{0}, \mathbf{1})$ and a weight function $\phi : V \to A$.

Therefore, the tuple $(G, (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}), f, (A, +, \times, \mathbf{0}, \mathbf{1}), \phi)$ is called the *specification* of a Forward-Backward algorithm.

Chapter 3

Advanced Instances of the Proposed Frameworks

3.1 Introduction of This Chapter

In this chapter, I will derive an efficient algorithm applicable to much broader types of summations than ones used in the ordinary Viterbi or Forward-Backward algorithm. As a result of the generalization proposed in this chapter, for example, I will derive an efficient algorithm applicable to the following form.

$$\sum_{\psi \in \Psi(G)} \left(\prod_{v \in \psi} \phi(v) \right) \left(\sum_{v \in \psi} f_1(v) \right)^{n_1} \cdots \left(\sum_{v \in \psi} f_K(v) \right)^{n_K}, \tag{3.1}$$

where $f_1, \ldots, f_K$ are K real-valued functions, and $n_1, \ldots, n_K$ are non-negative integers.

In order to systematically derive Viterbi and F.B. algorithms to compute complex summations like (3.1), I will introduce some definitions and notations in the next section.

3.2 Preliminaries

The first advanced instance of algebraic systems that complies with the frameworks is a sequence of over a semiring. A sequence over a semiring is an ordered list of elements of a semiring.

Definition 10. *For a given semiring* $(A, +, \times, 0, 1)$*, a sequence over* A *is defined as an ordered list of elements* $(x_i)_{i\in\mathbb{N}_0}$*, where* $\forall i \in \mathbb{N}_0,\ \ x_i \in A$*.*

The definition of a sequence over a semiring is very similar to the definition of vector space. In fact, additive operation is introduced as in vector spaces. However, a sequence over a semiring differs from vector spaces in the following respects. First, a sequence over a semiring is not requested to have some axioms for vector spaces. In particular, sequences over a semiring is not required to have inverse elements of addition and scalar multiplication. Second, unlike vector spaces, a multiplicative operation have to be defined between two sequences over a semiring. This allows to sequences of semirings follow the algebraic frameworks of forward-backward algorithm.

Now, the set of sequences over a semiring A with n elements, denoted by A^n, is the main concern in the following.

Definition 11. *For a given semiring* $(A, +, \times, 0, 1)$*, the set of all the* n*-length sequences over* A *is denoted by* A^n*, that is,* $A^n \stackrel{def}{\equiv} \{(a_i)_{i=0,1,\ldots,n-1} \mid a_i \in A\}$

Definition 12. *For two elements of* A^n*, denoted by* $\mathbf{a}$ *and* $\mathbf{b}$*, addition between* $\mathbf{a}$ *and* $\mathbf{b}$*is defined as follows,*

$$\mathbf{a} + \mathbf{b} \stackrel{def}{\equiv} (a_i + b_i)_{i=0,\ldots,n-1} \ . \tag{3.2}$$

A sequence is denoted by bold-faced alphabets throughout this dissertation unless otherwise specified.

Fact 2. $\mathbf{0}_{A^n} \stackrel{def}{\equiv} (0, 0, \ldots, 0)$ *is the identity element for the addition of* A^n*, that is,*

$$\mathbf{0}_{A^n} + \mathbf{a} = \mathbf{a} + \mathbf{0}_{A^n} = \mathbf{a} \qquad (\forall \mathbf{a} \in A^n) \ . \tag{3.3}$$

For the set of the all sequences over a semiring, three multiplicative operations are introduced in this dissertation. First two multiplicative operations on sequences are relatively famous ones, and the last one is, to author's knowledge, newly introduced into the contexts of algebraic frameworks for F.B.

Definition 13. *Element-wise product on* A^n*, denoted by* $\circ$*, is defined as follows,*

$$\mathbf{a} \circ \mathbf{b} \stackrel{def}{\equiv} (a_i \times b_i)_{i=0,\ldots,n-1} \ . \tag{3.4}$$

Fact 3. $\mathbf{1}_\circ \stackrel{def}{\equiv} (1, 1, \ldots, 1)$ *is the identity element for* $\circ$*, i.e.,*

$$\forall \mathbf{a} \in A^n,\ \ \mathbf{1}_\circ \circ \mathbf{a} = \mathbf{a} \circ \mathbf{1}_\circ = \mathbf{a}\ . \tag{3.5}$$

Fact 4. $(A^n, +, \circ, \mathbf{0}_{A^n}, \mathbf{1}_\circ)$ *is a semiring.*

Definition 14. *Convolution on* A^n *is defined as follows,*

$$\mathbf{a} * \mathbf{b} \stackrel{def}{\equiv} \left(\sum_{k=0}^{i} a_k \times b_{i-k} \right)_{i=0,\ldots,n-1}\ . \tag{3.6}$$

Fact 5.

$$\mathbf{1}_* \stackrel{def}{\equiv} (1, 0, \ldots, 0) \tag{3.7}$$

is the identity element for $*$*, i.e.,*

$$\forall \mathbf{a} \in A^n,\ \ \mathbf{1}_* * \mathbf{a} = \mathbf{a} * \mathbf{1}_* = \mathbf{a}\ . \tag{3.8}$$

Fact 6. $(A^n, +, *, \mathbf{0}_{A^n}, \mathbf{1}_*)$ *is a semiring.*

Definition 15. *Binomial convolution on* A^n*, denoted by an infix operator* $\diamond$*, is defined as follows,*

$$\mathbf{a} \diamond \mathbf{b} \stackrel{def}{\equiv} \left(\sum_{k=0}^{i} \binom{i}{k} \cdot a_k \times b_{i-k} \right)_{i=0,\ldots,n-1}\ , \tag{3.9}$$

where $\binom{n}{i}$ *denotes the binomial coefficients.*

Note that multiplication between elements of a semiring A and a non-negative integer $\binom{i}{k}$ should be interpreted in terms of the definition (2.7). In addition, note the bilinearity (2.6), $\binom{i}{k} \cdot (a_k \times b_{i-k}) = \left(\binom{i}{k} \cdot a_k\right) \times b_{i-k} = a_k \times \left(\binom{i}{k} \cdot a_k\right)$, so the parentheses indicating the order of these operations can be omitted.

Fact 7. $\mathbf{1}_*$*, defined in* (3.7)*, is the identity element for the binomial convolution of* A^n*, that is,*

$$\forall \mathbf{a} \in A^n,\ \ \mathbf{1}_* \diamond \mathbf{a} = \mathbf{a} \diamond \mathbf{1}_* = \mathbf{a}\ . \tag{3.10}$$

Theorem 2. $(A^n, +, \diamond, \mathbf{0}_{A^n}, \mathbf{1}_*)$ *is a semiring.*

Proof of Theorem 2

It has been shown that $(A^n, +, \mathbf{0}_{A^n})$ is a commutative monoid, and $(A^n, \diamond, \mathbf{1}_*)$ is a commutative monoid. Thus, only associativity of binomial convolution and distributivity are required to be shown. $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in A^n$,

$$
\begin{aligned}
&((\mathbf{a} \diamond \mathbf{b}) \diamond \mathbf{c})_i \\
&= \sum_{j=0}^{i} \binom{i}{j} \cdot (\mathbf{a} \diamond \mathbf{b})_j \times c_{i-j} \\
&= \sum_{j=0}^{i} \binom{i}{j} \cdot \left(\sum_{k=0}^{j} \binom{j}{k} \cdot a_k \times b_{j-k} \right) \times c_{i-j} \\
&= \sum_{j=0}^{i} \left(\sum_{k=0}^{j} \left(\binom{i}{j} \binom{j}{k} \right) \cdot a_k \times b_{j-k} \times c_{i-j} \right) \\
&\qquad (\because \text{distributivity of } A \text{ and distributivity of } \mathbb{N}_0 \text{ over } A) \\
&= \sum_{j=0}^{i} \left(\sum_{k=0}^{i} \delta_{k \le j} \times \left(\left(\binom{i}{j} \binom{j}{k} \right) \cdot a_k \times b_{j-k} \times c_{i-j} \right) \right) \\
&\qquad \left(\because j \le i \text{ holds in the inner sum, thus } \sum_{k=0}^{j} \bullet = \sum_{k=0}^{i} \delta_{k \le j} \times \bullet \right) \\
&= \sum_{k=0}^{i} \left(\sum_{j=0}^{i} \delta_{k \le j} \times \left(\left(\binom{i}{j} \binom{j}{k} \right) \cdot a_k \times b_{j-k} \times c_{i-j} \right) \right) \\
&\qquad (\because \text{ the range of the sum } \sum_{k=0}^{i} \text{ no longer depends on the variable of} \\
&\qquad \text{the other sum } \sum_{j=0}^{i}, \text{ thus the order of these sums can be exchanged.}) \\
&= \sum_{k=0}^{i} \left(\sum_{j=k}^{i} \left(\binom{i}{j} \binom{j}{k} \right) \cdot a_k \times b_{j-k} \times c_{i-j} \right) \\
&\qquad \left(\because \sum_{j=0}^{i} \delta_{k \le j} \times \bullet = \sum_{j=k}^{i} \bullet \right)
\end{aligned}
\tag{3.11}
$$

$$
\begin{aligned}
&= \sum_{k=0}^{i} \left(\sum_{\acute{\jmath}=0}^{i-k} \left(\binom{i}{\acute{\jmath}+k} \binom{\acute{\jmath}+k}{k} \right) \cdot a_k \times b_{\acute{\jmath}} \times c_{i-k-\acute{\jmath}} \right) \\
&\left(\because \text{ linear transform of the variable of the inner sum:} j \mapsto \acute{\jmath} \stackrel{\text{def}}{\equiv} j-k \right) \\
&= \sum_{k=0}^{i} \left(\sum_{j=0}^{i-k} \left(\binom{i}{k} \binom{i-k}{j} \right) \cdot a_k \times b_j \times c_{i-k-j} \right) \\
&\left(\because {}^{\forall} i,j,k \in \mathbb{N}_0, \;\; i \geq j \geq k, \;\; \binom{i}{j}\binom{j}{k} = \binom{i}{k}\binom{i-k}{j-k} \right) \\
&= \sum_{k=0}^{i} \left(\binom{i}{k} \cdot a_k \right) \times \left(\sum_{j=0}^{i-k} \binom{i-k}{j} \cdot b_j \times c_{i-k-j} \right) \\
&= \sum_{k=0}^{i} \binom{i}{k} \cdot a_k \times (\mathbf{b} \diamond \mathbf{c})_{i-k} \\
&= (\mathbf{a} \diamond (\mathbf{b} \diamond \mathbf{c}))_i \quad .
\end{aligned}
$$

Thus, the binomial convolution is associative. And the distributivity is shown as follows,

$$
\begin{aligned}
(\mathbf{a} \diamond (\mathbf{b}+\mathbf{c}))_i &= \sum_{j=0}^{i} \binom{i}{j} \cdot \mathbf{a}_j \times (\mathbf{b}+\mathbf{c})_{i-j} \\
&= \sum_{j=0}^{i} \left(\binom{i}{j} \cdot \mathbf{a}_j \times \mathbf{b}_{i-j} + \binom{i}{j} \cdot \mathbf{a}_j \times \mathbf{c}_{i-j} \right) \\
&= \sum_{j=0}^{i} \binom{i}{j} \cdot \mathbf{a}_j \times \mathbf{b}_{i-j} + \sum_{j=0}^{i} \binom{i}{j} \cdot \mathbf{a}_j \times \mathbf{c}_{i-j} \\
&= (\mathbf{a} \diamond \mathbf{b})_i + (\mathbf{a} \diamond \mathbf{b})_i \quad .
\end{aligned}
\tag{3.12}
$$

Thus, $(A^n, +, \diamond, \mathbf{0}, \mathbf{1}_*)$ is a semiring. □

Definition 16. *For an element x of a given semiring $(A, +, \times, 0, 1)$, a n-degree monomial sequence of x, denoted by $P_n(x)$, is defined by*

$$
P_n(x) \stackrel{def}{\equiv} (1, x, x^2, \cdots, x^n) \in A^{n+1} \quad . \tag{3.13}
$$

Note that, for an element of semiring x, the n-th power of x is defined in (2.2).

Theorem 3. *The following equation holds,*

$$
P_n(x) \diamond P_n(y) = P_n(x+y) \quad . \tag{3.14}
$$

Semiring	$+$	$\times$	0	1	isomorphic semiring
Element-wise product	$+$	$\circ$	$0_{\mathbb{K}^n}$	$1_\circ$	
Convolution	$+$	$*$	$0_{\mathbb{K}^n}$	1_*	element-wise product under the Fourier transform
Binomial convolution	$+$	$\diamond$	$0_{\mathbb{K}^n}$	1_*	convolution under binomial transform

Table 3.1: Table of semirings for the set of sequences over a semiring

Proof. Trivial because this is the binomial theorem in A. □

Hereinafter, the sequence of the binomial convolution is denoted by $\diamond_i x_i$, that is, for example, $\diamond_{i=0,1,2,\ldots} x_i = x_0 \diamond x_1 \diamond x_2 \diamond \cdots$.

Lemma 1.

$$\diamond_{i\in\mathcal{I}} P_n(x_i) = P_n\left(\sum_{i\in\mathcal{I}} x_i\right) \tag{3.15}$$

Proof. Trivial from Theorem 3. □

Lemma 2. *The result of the Viterbi algorithm specified by* $(G, (A^n, +, \diamond, \mathbf{0}, \mathbf{1}_*), \phi \circ P_n)$ *is* $\sum_{\psi\in\Psi} \left(\sum_{v\in\psi} \phi(v)\right)^i \ (i = 0, \ldots, n)$.

Proof. Trivial because, from Lemma 1,

$$\sum_{\psi\in\Psi} (\diamond_{v\in\psi} P_n(\phi(v))) = \sum_{\psi\in\Psi} P_n\left(\sum_{v\in\psi} \phi(v)\right) \quad . \tag{3.16}$$

□

Definition 17. *Let* $\mathbb{K}$ *be a commutative semiring,* A *be a semimodule over* $\mathbb{K}$ *and* $(A^n, +, \times, 0, 1)$ *be a semiring, then,* $\cdot : \mathbb{K} \otimes A^n \to A^n$ *with the following definition is called scalar multiplication,*

$$(r \cdot \mathbf{a})_i = r \cdot (\mathbf{a})_i \ \ (r \in \mathbb{K}, \mathbf{a} \in A^n) \ .$$

Theorem 4. *Let* $\mathbb{K}$ *be a commutative semiring,* A *be a unital associative algebra over* $\mathbb{K}$ *and* $(A^n, +, \times, 0, 1)$ *be a semiring then,* $(A^n, +, \times, 0, 1)$ *equipped with the scalar multiplication defined in Definition 17 is a unital associative algebra over* $\mathbb{K}$.

Proof. Trivial. □

Lemma 3. *The result of the Forward-Backward algorithm specified by* $(G, (\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}), f, (A^n, +, \diamond, \mathbf{0}, \mathbf{1}_*), \phi \circ P_n)$ *is*

$$\sum_{\psi \in \Psi} \left(\left(\bigoplus_{v \in \psi} f(v) \right) \cdot \left(\sum_{v \in \psi} \phi(v) \right)^i \right), \quad (i = 0, \ldots, n) \ .$$

Definition 18. *Let* $\mathbb{K}$ *be a commutative semiring and* A *be a unital associative algebra over* $\mathbb{K}$*, a tensor over* A *with rank* $r \in \mathbb{N}_0$ *is a* r*-dimensional array over elements of* A*. A tensor over* A *with rank* 0 *is defined by an element of* A*.*

Here, the term "rank" is used in its tensor meaning. Do not confuse the usage with the matrix meaning.

For example, the sequence over A, introduced by Definition 10, is a tensor over A with rank 1.

For a r-rank tensor over A, each component in the tensor $\mathbf{T}$ is uniquely specified by the notation $\mathbf{T}_{i_1, i_2, \cdots, i_r}$, which means the i_1-th component in the first dimension of the i_2-th component in the second dimension of ... of the i_r-th component in the r-th dimension.

Definition 19. *Given two tensors over* A*,* $\mathbf{T}$ *with rank* r *and* $\mathbf{T}'$ *with rank* r'*, the tensor product of* $\mathbf{T}$ *and* $\mathbf{T}'$ *is defined by*

$$(\mathbf{T} \otimes \mathbf{T}')_{i_1, i_2, \ldots, i_{r+r'}} \overset{def}{\equiv} \mathbf{T}_{i_1, i_2, \cdots, i_r} \times \mathbf{T}_{i_{r+1}, i_{r+2}, \cdots, i_{r+r'}} \ .$$

Definition 20. *Let* $\mathbb{K}$ *be a commutative semiring,* $T = (A^{n_1, \ldots, n_r}, +_T, \times_T, 0_T, 1_T)$ *be a unital associative algebra over* $\mathbb{K}$*, and* $V = (A^n, +_V, \times_V, 0_V, 1_V)$ *be a vector space over* $\mathbb{K}$*, then,* $T' = (A^{n_1, \cdots, n_r, n_{r+1}}, +_{T'}, \times_{T'}, 0_{T'}, 1_{T'})$ *with the following definitions is called the tensor product of algebras* T *and* V*.*

1. *If* $r = 0$*, then,* $T' = (A^n, +_V, \times_V, 0_V, 1_V)$*.*
2. *If* $r \geq 1$*, given* $\mathbf{T}, \mathbf{T}' \in A^{n_1, \ldots, n_r}$ *and* $\mathbf{a}, \mathbf{a}' \in A^n$*, then* $\mathbf{T} \otimes \mathbf{a}, \mathbf{T}' \otimes \mathbf{a}' \in A^{n_1, \ldots, n_r, n_{r+1}}$*, and*
   (a) *addition in* $A^{n_1, \ldots, n_r, n_{r+1}}$ *is defined by*

$$+_{T'} : (\mathbf{T} \otimes \mathbf{a}) \otimes (\mathbf{T}' \otimes \mathbf{a}') \mapsto (\mathbf{T} +_T \mathbf{T}') \otimes (\mathbf{a} +_V \mathbf{a}') \quad , \tag{3.17}$$

(b) production in $A^{n_1,\ldots,n_r,n_{r+1}}$ *is defined by*

$$\times_{T'} : (\mathbf{T} \otimes \mathbf{a}) \otimes (\mathbf{T}' \otimes \mathbf{a}') \mapsto (\mathbf{T} \times_T \mathbf{T}') \otimes (\mathbf{a} \times_V \mathbf{a}') \ , \tag{3.18}$$

(c) $0_{T'} \stackrel{def}{\equiv} 0_T \otimes 0_V$,

(d) $1_{T'} \stackrel{def}{\equiv} 1_T \otimes 1_V$,

(e) scalar multiplication between $\mathbb{K}$ *and* $A^{n_1,\ldots,n_r,n_{r+1}}$ *is defined by*

$$\cdot : r \otimes (\mathbf{T} \otimes \mathbf{a}) \mapsto \mathbf{T} \otimes (r \cdot \mathbf{a}) \ . \tag{3.19}$$

Theorem 5. *The tensor product of* $(A^{n_1,\ldots,n_r}, +, \times, 0, 1)$ *and* $(A^n, +, \times, 0, 1)$ *is a unital associative algebra over* $\mathbb{K}$.

Proof. Trivial. □

From Definition 20 and Theorem 5, we can construct a set of tensors equipped with the algebraic structure of a unital associative algebra over a commutative semiring from a set of sequences over a semiring. A tensor product of algebras $T = (A^{n_1,\ldots,n_r}, +_T, \times_T, 0_T, 1_T)$ constructed by $A_1 = (A^{n_1}, +_{A_1}, \times_{A_1}, 0_{A_1}, 1_{A_1})$, $A_2 = (A^{n_2}, +_{A_2}, \times_{A_2}, 0_{A_2}, 1_{A_2})$, $\ldots$, $A_r = (A^{n_r}, +_{A_r}, \times_{A_r}, 0_{A_r}, 1_{A_r})$ is denoted by $A_1 \otimes \cdots \otimes A_r$.

Combining tensor products of algebras and the fact described in Lemma 1, we obtain the specification of the Viterbi or Forward-Backward algorithms that result in very complicated summations.

Example 1. *The result of the Viterbi algorithm specified by*

$$(G, A_1 \otimes \cdots \otimes A_r, P_{\phi,\phi_1,\ldots,\phi_n}) \ ,$$

where

$$\begin{aligned} A_i &\stackrel{def}{\equiv} (A^{n_i}, +, \diamond, 0_{A^{n_i}}, 1_*) \ \ (i = 1, \ldots, r) \ , \\ P_{\phi,\phi_1,\ldots,\phi_n} : v \mapsto \phi(v) \cdot (1, \phi_1(v), &(\phi_1(v))^2, \ldots, (\phi_1(v))^{n_1}) \\ &\otimes (1, \phi_2(v), (\phi_2(v))^2, \ldots, (\phi_2(v))^{n_2}) \\ &\otimes \cdots \\ &\otimes (1, \phi_r(v), (\phi_r(v))^2, \ldots, (\phi_r(v))^{n_r}) \end{aligned}$$

is

$$\sum_{\psi \in \Psi(G)} \left(\prod_{v \in \psi} \phi(v) \right) \times \left(\sum_{v \in \psi} \phi_1(v) \right)^{i_1} \cdots \times \left(\sum_{v \in \psi} \phi_r(v) \right)^{i_r} ,$$
$$(i_1 = 0, \ldots, n_1, \ldots, i_r = 0, \ldots, n_r) \ .$$

Example 2. *The result of the Forward-Backward algorithm specified by*

$$(G, \mathbb{K}, f, A_1 \otimes \cdots \otimes A_r, P_{\phi,\phi_1,\ldots,\phi_n})$$

is

$$\sum_{\psi \in \Psi(G)} \left(\bigotimes_{v \in \psi} f(v) \right) \cdot \left(\prod_{v \in \psi} \phi(v) \right) \times \left(\sum_{v \in \psi} \phi_1(v) \right)^{i_1} \cdots \times \left(\sum_{v \in \psi} \phi_r(v) \right)^{i_r}, \\ (i_1 = 0, \ldots, n_1, \ldots, i_r = 0, \ldots, n_r) \ .$$

3.3 Applications

In this section, I assume that the distribution over output sequences is modeled by CRFs, which is defined as

$$P(\pi \mid \mathbf{x}; \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{\lambda}; \mathbf{x})} \left(\prod_{v \in \pi} \phi(\mathbf{x}, v) \right)$$

with $Z(\boldsymbol{\lambda}; \mathbf{x}) \stackrel{\text{def}}{\equiv} \sum_{\pi \in \Psi(G(\mathbf{x}))} \left(\prod_{v \in \pi} \phi(\mathbf{x}, v) \right)$ and $\phi(\mathbf{x}, v) \stackrel{\text{def}}{\equiv} \exp\left(\sum_k \lambda_k f_k(\mathbf{x}, v) \right)$, where $\mathbf{x}$ is the input sequence, and $G(\mathbf{x})$ is the DAG that encodes all the output paths conditioned on the input $\mathbf{x}$.

For CRFs, it is sometimes necessary to calculate the covariance of feature functions. For feature functions $F_1(\pi), F_2(\pi)$ $(\pi \in \Psi(G(\mathbf{x})))$, the covariance is

$$\begin{aligned} \mathrm{Cov}\left[F_1(\pi), F_2(\pi)\right] &= E\left[F_1(\pi) F_2(\pi)\right] \\ &\quad - E\left[F_1(\pi)\right] E\left[F_2(\pi)\right] \ . \end{aligned} \tag{3.20}$$

If both F_1 and F_2 are of the form $\sum_{v \in \pi} f_k(v)$, the expectations in the second term of (3.20) can be calculated by the ordinary forward-backward algorithm. The first term of (3.20) can be calculated by the generalized forward-backward algorithm described in Example 2, where $\phi(v) = \phi(\mathbf{x}, v)$ and $\phi_1(v) = f_k(v)$.

The covariance of feature functions is useful for optimization of CRFs. Because it has rich information of the curvature of the log-likelihood objective function, i.e., $\frac{\partial^2}{\partial \lambda_i \partial \lambda_j} \mathcal{L}(\boldsymbol{\lambda}) = -\mathrm{Cov}[F_i(\mathbf{x}, \pi), F_j(\mathbf{x}, \pi)]$ where $\mathcal{L}$ is the log-likelihood objective function for $P(\pi \mid \mathbf{x}; \lambda)$. For example, [27] used stochastic gradient methods to accelerate the optimization of the log-likelihood objective for CRFs. In stochastic gradient methods, it is required to calculate the Hessian-vector product. Although they use automatic differentiation to calculate the Hessian-vector product,

I can derive an alternative algorithm for it as a specialization of the generalized forward-backward algorithm. That algorithm is equivalent to the calculation by the automatic differentiation as far as the computational cost is concerned.

As another practical application of the covariance, I can take differentiation of more complicated objective functions than the log-likelihood objective. For example, the objective function used in [13] has the conditional entropy of CRFs, i.e., $\mathcal{H}(\boldsymbol{\lambda}) = E[\log P(\pi|\mathbf{x};\boldsymbol{\lambda})]$. A specialization of our generalized algorithm offers an efficient way to calculate the parameter gradient of it.

[14] proposed to use Hamming loss objective function in parameter estimations involved in sequence labeling. For a correctly annotated input-output sequence pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, the objective to be optimized is $C_1(\boldsymbol{\lambda}; \hat{\mathbf{x}}, \hat{\mathbf{y}}) := \frac{1}{T}\sum_t \log P(\hat{y}_t|\hat{\mathbf{x}};\boldsymbol{\lambda})$. To calculate the gradient of this objective function, [14] derived a sort of the forward-backward algorithm, which is as efficient as the ordinary forward-backward algorithm. I can show that a specialization of the generalization proposed in this chapter also leads to an equivalent algorithm.

[19] proposed a semi-supervised training method called generalized expectation criteria to improve accuracy with unlabeled data. The objective used in [19] has an additional term

$$-\theta D(\hat{P}||\tilde{P}) \ , \tag{3.21}$$

where θ is a given hyper-parameter, D is the KL divergence, $\hat{P}$ is a given target distribution and $\tilde{P}$ is the conditional distribution of labels given a feature $f_m(\mathbf{x}, t)$ at time t, i.e., [1]

$$\tilde{P} := \tilde{P}(y_t|f_m(\mathbf{x}, t) = 1; \lambda) \ . \tag{3.22}$$

I now focus on the parameter gradient. Instead of the derivation shown in [19], I first rephrase $\tilde{P}(y|f_m(x,t) = 1;\lambda)$ as

$$\begin{aligned} &\tilde{P}(\ell(v)|f_m(v) = 1;\lambda) \\ &\quad = \frac{1}{U_m}\sum_{\mathbf{x}\in\mathbf{U}_m}\sum_{\pi\in\Psi(G)} P(\pi|\mathbf{x};\lambda)\Delta_{f_m,y}(\pi) \ , \end{aligned} \tag{3.23}$$

where $\ell(v)$ is the label assigned to v, $\mathbf{U}_m$ is the set of sequences where f_m is present for some nodes, and

$$\Delta_{f_m,y}(\pi) = \sum_{v\in\pi} f_m(v)\delta_{\ell(v),y} \ . \tag{3.24}$$

[1] In the same way as their paper, I assume that all output sequences are same in length and f_m is binary.

Noting the following relation

$$\begin{aligned}\Delta_{f_m}(\pi) &\stackrel{\text{def}}{\equiv} \sum_l \frac{\hat{P}}{\tilde{P}} \Delta_{f_m,l}(\pi) \\ &= \sum_{v\in\pi} \frac{\hat{P}(\ell(v)|f_m(v)=1)}{\tilde{P}(\ell(v)|f_m(v)=1;\lambda)} f_m(v) \ ,\end{aligned}$$

I get

$$\frac{\partial}{\partial\lambda_k} D(\hat{P}||\tilde{P}) = -\frac{1}{U_m} \sum_{\mathbf{x}\in\mathbf{U}_m} \sum_l \frac{\hat{P}}{\tilde{P}} \frac{\partial}{\partial\lambda_k} E\left[\Delta_{f_m,l}(\pi)\right] \ , \tag{3.25}$$

and

$$\begin{aligned}&\sum_l \frac{\hat{P}}{\tilde{P}} \frac{\partial}{\partial\lambda_k} E\left[\Delta_{f_m,l}(\pi)\right] \\ &\quad = -E[\Delta_{f_m}(\pi) F_k(\mathbf{x},\pi)] + E[\Delta_{f_m}(\pi)] E[F_k(\mathbf{x},\pi)] \ .\end{aligned}$$

Here, the first term can be calculated by the generalized order forward-backward algorithm, the second term by the ordinary forward-backward algorithm.

Note that this calculation can drastically reduce the computational cost compared with the algorithm proposed in [19]. Computational cost needed by our formulation is scaled by up to a constant factor compared with the ordinary forward-backward algorithm, whereas [19] takes the forward-backward algorithm for each label so computational cost is proportional to the number of labels.

3.4 Further Possibilities

In this section, I will show that I can calculate summations that involve quite complex functions by series expansions.

Consider the following summation

$$\sum_{\psi\in\Psi(G)} (F_1(\psi))^{i_1} \times \cdots \times (F_r(\psi))^{i_r} \times \phi(F(\psi)) \ , \tag{3.26}$$

where $F_i = \sum_{v\in\psi} f_i(v)$ and $\phi(x)$ is some complicated function. Even for such a case, I can approximate the summation by a series expansion of ϕ.

If $\phi(x)$ has the Taylor series expansion around $x_0 \in \mathbb{C}$ with the convergence radius $r > 0$, and $|x - x_0| \leq r$ for ${}^{\forall}x \in \left[\min_{\psi \in \Psi(G)} F(\psi), \max_{\psi \in \Psi(G)} F(\psi)\right]$, then

$$(3.26) = \sum_{\psi \in \Psi(G)} (F_1(\psi))^{i_1} \times \cdots \times (F_r(\psi))^{i_r} \times \sum_{n=0}^{\infty} a_n (x - x_0)^n \ ,$$

where $a_n \stackrel{\text{def}}{\equiv} \frac{1}{n!} \cdot \frac{d^n \phi(x)}{dx^n}\Big|_{x=x_0}$. By truncating higher terms of the expansion, I obtain the truncated Taylor series expansion for (3.26).

$$(3.26) \approx \sum_{n=0}^{N} a_n \sum_{\psi \in \Psi(G)} (F_1(\psi))^{i_1} \times \cdots \times (F_r(\psi))^{i_r} \times (F'(\psi))^{n-1} \ ,$$

where $F'(\psi) \stackrel{\text{def}}{\equiv} \sum_{v \in \psi} f'(v)$, $f'(v) \stackrel{\text{def}}{\equiv} f(v) - x_0 \delta_{v=src}$. All the terms appearing in the truncated Taylor series expansion can be calculated by the generalized Viterbi algorithm described in Example 1.

Another possible series expansion is the Fourier series expansion. If ϕ in (3.26) is a $2L$-periodic function that is integrable on $[-L, L]$ $(L > 0)$, then $\phi(x)$ has the Fourier series expansion, i.e.,

$$\phi(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right) \ ,$$

where

$$a_n \stackrel{\text{def}}{\equiv} \frac{1}{L} \int_{-L}^{L} \phi(x) \cos\left(\frac{n\pi x}{L}\right) dx \quad (n = 0, 1, 2, \dots) \ ,$$
$$b_n \stackrel{\text{def}}{\equiv} \frac{1}{L} \int_{-L}^{L} \phi(x) \sin\left(\frac{n\pi x}{L}\right) dx \quad (n = 1, 2, 3, \dots) \ .$$

By truncating higher terms of the expansion, I obtain

$$\phi(x) \approx \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right) \ ,$$

so

$$
\begin{aligned}
(3.26) \approx \frac{a_0}{2} & \sum_{\pi \in \Psi(G)} h_{1,i_1}(F_1(\pi)) \cdots h_{M,i_M}(F_M(\pi)) \\
& + \sum_{n=1}^{N} a_n \sum_{\pi \in \Psi(G)} h_{1,i_1}(F_1(\pi)) \cdots h_{M,i_M}(F_M(\pi)) \\
& \qquad \times \cos\left(\frac{n\pi F_{M+1}(\pi)}{L}\right) \qquad (3.27) \\
& + \sum_{n=1}^{N} b_n \sum_{\pi \in \Psi(G)} h_{1,i_1}(F_1(\pi)) \cdots h_{M,i_M}(F_M(\pi)) \\
& \qquad \times \sin\left(\frac{n\pi F_{M+1}(\pi)}{L}\right)
\end{aligned}
$$

I can calculate all the terms appearing in the truncated Fourier series expansion above by the generalized forward algorithm proposed in this chapter.

To show a sample application of series expansions, I will describe the expected F-measure optimization.

[12] utilized the expected f-measure as an objective function in optimization of logistic regression models. He has just described on single label, non-structured case. Here, let this criteria be sequentially-extended. I define

$$
\mathcal{F}_\gamma(\lambda; \hat{\mathbf{x}}) := E[\mathcal{F}_\gamma(\pi)] \ , \qquad (3.28)
$$

where the expectation is taken under CRFs, and $\mathcal{F}_\gamma(\pi)$ is the label-wise f-measure for a given output sequence π, i.e., $\mathcal{F}_\gamma(\pi) := \frac{(1+\gamma^2)l(\pi)}{|\pi|+\gamma^2 L}$. L is the number of labels in the correctly annotated answer sequence, and $l(\pi)$ is the number of labels in the output sequence π that coincide with the correct labels. $|\pi|$ is the total number of labels appearing in π. Hereafter, I refer only to the case $\gamma = 1$. Let $\mathcal{F}(\pi) := \mathcal{F}_1(\pi)$. For the case where all the possible output sequences have the same number of labels (length), optimization concerning (3.28) – calculations of value and gradient – requires the forward-backward algorithm for the form $h_{1,i_1}(F_1(\pi))h_{2,i_2}(F_2(\pi))$, where $n_1 = 1,\ h_{1,1}(x) \stackrel{\text{def}}{\equiv} \exp(x)$ and $n_2 = 2,\ h_{2,i_2}(x) \stackrel{\text{def}}{\equiv} x^{i_2-1}$. On the other hand, for the case where the lengths of the output sequences are different as in [23], because the denominator in (3.28) is variable in the summation, calculations of the value and the gradient of (3.28) are not trivial at all.

However, with the Taylor expansion of the reciprocal function in (3.28), I get

$$E[\mathcal{F}(\pi)] = \frac{2}{Z(\lambda;\mathbf{x})(L+x_0)} \times \sum_{n=0}^{\infty} \sum_{\pi\in\Psi(G)} \Phi(\pi) l(\pi) \left(\frac{|\pi| - x_0}{-L - x_0}\right)^n \quad , \tag{3.29}$$

where x_0 is a complex value that satisfies the condition $\left|\frac{|\pi|-x_0}{-L-x_0}\right| < 1 \ (^\forall\pi \in \Psi(G))$ (such a value really exists at any time). By truncating higher terms in the Taylor series, I can approximate the value as follows,

$$E[\mathcal{F}(\pi)] \approx \frac{2}{Z(\lambda;\mathbf{x})(L+x_0)} \times \sum_{n=0}^{N} \sum_{\pi\in\Psi(G)} \Phi(\pi) l(\pi) \left(\frac{|\pi| - x_0}{-L - x_0}\right)^n \quad . \tag{3.30}$$

The summation $\sum_{\pi\in\Psi(G)}$ in (3.30) is of the form to which the generalized forward algorithm is applicable. I can also derive an approximation of the gradient of (3.30).

$$\frac{\partial}{\partial\lambda_k} E[\mathcal{F}(\pi)] = E[\mathcal{F}(\pi) F_k(\mathbf{x},\pi)] - E[\mathcal{F}(\pi)] E[F_k(\mathbf{x},\pi)] \tag{3.31}$$

The first and second terms in (3.31) can be approximated by the generalized forward-backward algorithm in a similar manner to (3.30). Therefore, I can optimize the objective with numerical optimization routines.

For a given constant N, these calculations are as efficient as the ordinary forward-backward algorithm, scaled by up to a constant factor. For trellises in which the length of a node is up to 5, I have experimentally confirmed that $N \sim 15$ proves to be sufficient to approximate the objective and the gradient with enough precision under the double-precision floating-point arithmetic.

3.5 Conclusions of this Chapter

In this chapter, I proposed a generalization of the forward-backward algorithm, which is applicable to much broader types of summations over all possible sequences than the ordinary forward-backward algorithm. I also show that our

theorems in this chapter can offer efficient algorithms for some calculations required in past studies, and even for summations that involve quite complex functions with series expansions.

I have just confirmed that our generalization is runnable on existing calculations. I am now investigating other possibilities of application of our generalization, including optimization of much more complex probabilistic models than log-linear models and boosting algorithms on sequence labeling tasks.

While I proposed the generalization only for linear chains, it is straightforward to extend this generalization to trees. However, it will be much more challenging to find a counterpart in the case of structures that have loops. I am going to study such a possibility.

Chapter 4

Application to a Real World Task - Multilayer Sequence Labeling

4.1 Introduction

Machine learning approach is widely used to classify instances into discrete categories. In many tasks, however, some set of interrelated labels should be decided simultaneously. Such tasks are called *structured prediction*. *Sequence labeling* is the simplest subclass of structured prediction problems. In sequence labeling, the most likely one among all the possible label sequences is predicted for a given input. Although sequence labeling is the simplest subclass, a lot of real-world tasks are modeled as problems of this simplest subclass. In addition, it might offer valuable insight and a toehold for more general and complex structured prediction problems. Many models have been proposed for sequence labeling tasks, such as Hidden Markov Models (HMM), Conditional Random Fields (CRF) [16], Max-Margin Markov Networks [25] and others. These models have been applied to lots of practical tasks in natural language processing (NLP), bioinformatics, speech recognition, and so on. And they have shown great success in recent years.

In real-world tasks, it is often needed to cascade multiple predictions. A *cascade of predictions* here means the situation in which some of predictions are made based upon the results of other predictions. Sequence labeling is not an exception. For example, in NLP, I perform named entity recognition or base-phrase chunking for given sentences based on part-of-speech (POS) labels predicted by another sequence labeler. Natural languages are especially interpreted to have a hierarchy of sequential structures on different levels of abstraction. Therefore, many tasks

in NLP are modeled as a cascade of sequence predictions.

If a prediction is based upon the result of another prediction, I call the former *upper* stage and the latter *lower* stage.

Methods pursued for a cascade of predictions –including sequence predictions, of course–, are desired to perform certain types of capability. The types of capability is two-folded. One is *rich forward information propagation*, that is, the learning and estimation on each stage of predictions should utilize rich information of the results of lower stages whenever possible. "Rich information" here includes next bests and confidence information of the results of lower stages. The other is *backward information propagation*, that is, the rich annotated data on an upper stage should affect the models on lower stages retroactively.

Many current systems for a cascade of sequence predictions adopt a simple 1-best feed-forward approach. They simply take the most likely output at each prediction stage and transfer it to the next upper stage. Such a framework can maximize reusability of existing sequence labeling systems. On the other hand, it exhibits a strong tendency to propagate errors to upper labelers.

Typical improvement on the 1-best approach is to keep k-best results in the cascade of predictions. However, the more k becomes, the more difficult it is to enumerate and maintain the k-best results. It is particularly prominent in sequence labeling.

The essence of this orientation is that the labeler on an upper stage utilizes the information of all the possible output candidates on lower stages. However, the size of the output space can become quite large in sequence labeling. It effectively forbids explicit enumeration of all possible outputs, so it is require to represent all the labeling possibilities compactly or employ some approximation schemes. Some studies look in this way. In the method proposed in Finkel et al. [7], a cascades of sequence predictions is viewed as a Bayesian network, and sample sequences are drawn at each stage according to the output distribution. The samples are then used to estimate the entire distribution of the cascade. In the method proposed in Bunescu [2], an upper labeler uses the probabilities marginalized on the parts of the output sequences on lower stages as weights for the features. The weighted features are integrated in the model of the labeler on the upper stage. A k-best approach and the methods mentioned above are effective to improve the forward information propagation. However, they can never contribute on backward information propagation.

To improve the both directions of information propagation, Some studies propose the joint learning of multiple sequence labelings. Sutton et al. [24] proposes the joint learning method in case where multiple labels are assigned to each time slice of the input sequences. It enables simultaneous learning and estimation of multiple sequence labelings on the same input sequences, where time slices of the outputs of all the out sequences are regularly aligned. However, it puts the distribution of states into Bayesian networks with cyclic dependencies, and exact inference is not tractable in such a model in general. Therefore, it requires some approximative inference algorithms in learning or predictions. Moreover, it only considers the cases where labels of an input sequence and all output sequences are regularly aligned. It is not clear how to build a joint labeling model which handles irregular output label sequences like semi-Markov models.

In this chapter, I propose a middle ground for a cascade of sequence predictions. The proposed method adopts the basic idea of Bunescu [2]. I first assume that the model on all the sequence labeling stages is probabilistic one. In modeling of an upper stage, a feature is weighted by the marginal probability of the fragment of the outputs from a lower stage. However, this is not novel itself because it is just a paraphrase of Bunescu's core idea. Our intuition behind the proposed method is about as follows. Features integrated in the model on each stage are weighted by the marginal probabilities of the fragments of the outputs on lower stages. So, if the output distributions on lower stages are changed, the marginal probability of any fragment is also changed, and this in turn can changes the value of the features on the upper stage. In other words, the features on an upper stage indirectly depend on the models on the lower stages. Based on this intuition, the learning procedure of the model on an upper stage can affect not only direct model parameters, but also the weights of the features by changing the model on the lower stages. Supervised learning based on annotated data on an upper stage may affect the model or model parameters on the lower stages. It could be said that the information of annotation data on an upper stage is propagated back to the model on lower stages.

In the next section, I describe the formal notation of our model. In Section 4.3, I propose an optimization procedure according to the intuition noted above. In Section 4.4, I report an experimental result of our method. The proposed method shows some improvements on a real-world task in comparison with ordinary methods.

4.2 Formalization

In this section, I introduce the formal notation of our model. Hereafter, for the sake of simplicity, I only describe the simplest case in which there are just two stages, one lower stage of sequence labeling named L_1 and one upper stage of sequence labeling named L_2. In L_1, the most likely one among a set of possible sequences is predicted for a given input $\mathbf{x}$. L_2 is also a sequence labeling stage for the same input $\mathbf{x}$ and the output of L_1. No assumption is made on the structure of $\mathbf{x}$. The information of $\mathbf{x}$ is totally encoded in feature functions. It is only assumed that the output spaces of both L_1 and L_2 are conditioned on the initial input $\mathbf{x}$.

First of all, I describe the formalization of the probabilistic model for L_1. The model for L_1 per se is the same as ordinary ones for sequence labeling. For a given input $\mathbf{x}$, consider a directed acyclic graph (DAG) $G_1 = (V_1, E_1)$. A *source* of a DAG G is a node whose in-degree is equal to zero. A *sink* of a DAG G is nodes whose out-degree is equal to zero. Let $src(G)$, $snk(G)$ denote the set of source and sink nodes in G, respectively. A *successful path* of a DAG G is defined as a directed path on G whose starting node is a source and end node is a sink. If $\mathbf{y}$ denotes a path on a DAG, let $\mathbf{y}$ also denote the set of all the arcs appearing on $\mathbf{y}$ for the sake of shorthand. I denote the set of all the possible successful paths on G_1 by $\mathbf{Y}_1$. The space of the output candidates for L_1 is exactly equal to $\mathbf{Y}_1$. For the modeling of L_1, it is assumed that features of the form $f_{\langle 1,k_1,e_1,\mathbf{x}\rangle} \in \mathbb{R} \quad (k_1 \in \mathcal{K}_1, e_1 \in E_1)$ are allowed to be used. Here, $\mathcal{K}_1$ is the index set of the feature types for L_1. Such a feature can capture an aspect of the correlation between adjacent nodes. I call this kind of features *input features* for L_1. This naming is used to distinguish them from another kind of features defined on L_1, which comes later. Although features on V_1 can be also defined, they are totally omitted in this chapter for brevity. Hereafter, if a symbol has subscripts, then let the same but boldfaced symbol without some of the subscripts denote the set over which the omitted subscripts range. For example, $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle} \stackrel{\text{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{k_1\in\mathcal{K}_1}$, $\mathbf{f}_{\langle 1,k_1,\mathbf{x}\rangle} \stackrel{\text{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{e_1\in E_1}$, $\mathbf{f}_{\langle 1,\mathbf{x}\rangle} \stackrel{\text{def}}{\equiv} \left\{f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}\right\}_{k_1\in\mathcal{K}_1,e_1\in E_1}$, and so on. The probabilistic model on L_1 forms the log-linear model, that is,

$$P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \stackrel{\text{def}}{\equiv} \frac{1}{Z_1(\mathbf{x};\boldsymbol{\theta}_1)} \exp\bigl(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\bigr) \qquad (\mathbf{y}_1 \in \mathbf{Y}_1) \quad , \tag{4.1}$$

where $\theta_{\langle 1,k_1\rangle} \in \mathbb{R}$ $(k_1 \in \mathcal{K}_1)$ is the *parameter* or *weight* for the feature of the same index k_1, and $F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle} \stackrel{\text{def}}{\equiv} \sum_{e_1 \in \mathbf{y}_1} f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}$. Dot operator $(\cdot)$ denotes the inner product with respect to the subscripts commonly missing in both operands. Z_1 is the *partition function* for P_1, defined as

$$Z_1(\mathbf{x};\boldsymbol{\theta}_1) \stackrel{\text{def}}{\equiv} \sum_{\mathbf{y}_1 \in \mathbf{Y}_1} \exp\bigl(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\bigr) \quad . \tag{4.2}$$

It is worth noting that this formalization subsumes both directed and undirected linear-chain graphical models, which are the most typical models for sequence labeling, including HMM and CRF. That is, if the elements of V_1 are aligned into regular time slices, and the nodes in each time slice are associated with possible assignments of labels for that time, I obtain the representation equivalent to the ordinary linear-chain graphical models, in which all possible label assignments for each state are expanded. In such configuration, all the possible successful paths defined in our notation have strict one-to-one correspondence to all the possible joint assignments of labels in linear-chain graphical models. I purposely employ this DAG-based notation because; it is quite convenient to describe the models and algorithms for our purpose, it allows for labels to stay in arbitrary time as in semi-Markov models, and it is easily extended to models for a set of trees instead of sequences by replacing the graph-based notation with hypergraph-based notation.

Next, I formalize the probabilistic model on the upper stage L_2. The same as L_1, consider a DAG $G_2 = (V_2, E_2)$ conditioned on the input $\mathbf{x}$, and the set of all the possible successful paths on G_2, denoted $\mathbf{Y}_2$. The space of the output candidates for L_2 becomes $\mathbf{Y}_2$.

The form of the features available in designing the probabilistic model for L_2, denoted by P_2, is the key of the method proposed in this chapter. A feature on an arc $e_2 \in E_2$ can access local characteristics of the confidence-rated superposition of the L_1's outputs, in addition to the information of the input $\mathbf{x}$. To formulate local characteristics of the superposition of the L_1's outputs, I first define *output features* of L_1, denoted by $h_{\langle 1,k'_1,e_1\rangle} \in \mathbb{R}$ $(k'_1 \in \mathcal{K}'_1,\, e_1 \in E_1)$. Here, $\mathcal{K}'_1$ is the index set of the output feature types of L_1. Before the output features are integrated into the model for L_2, they all are confidence-rated with respect to P_1, that is, each output feature $h_{\langle 1,k'_1,e_1\rangle}$ is numerically rated by the estimated probabilities summed over the sequences emitting that feature. More formally,

all the L_1's output features are integrated in features for P_2 in the form of the *marginalized output features*, which are defined as follows;

$$\begin{aligned}\bar{h}_{\langle 1,k_1',e_1\rangle}(\boldsymbol{\theta}_1) &\stackrel{\text{def}}{\equiv} h_{\langle 1,k_1',e_1\rangle} P_1(e_1|\mathbf{x};\boldsymbol{\theta}_1) \\ &\qquad (k_1' \in \mathcal{K}_1',\ e_1 \in E_1) \quad ,\end{aligned} \tag{4.3}$$

where

$$\begin{aligned}P_1(e_1|\mathbf{x};\boldsymbol{\theta}_1) &\stackrel{\text{def}}{\equiv} \sum_{\mathbf{y}_1 \sim e_1} P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \\ &= \sum_{\mathbf{y}_1 \in \mathbf{Y}_1} \delta_{e_1 \in \mathbf{y}_1} P_1(\mathbf{y}_1|\mathbf{x};\boldsymbol{\theta}_1) \\ &\qquad (e_1 \in E_1) \quad .\end{aligned} \tag{4.4}$$

Here, the notation $\sum_{\mathbf{y}_1 \sim e_1}$ represents the summation over sequences consistent with an arc $e_1 \in E_1$, that is, the summation over the set $\{\mathbf{y}_1 \in \mathbf{Y}_1 \mid e_1 \in \mathbf{y}_1\}$. $\delta_{\mathcal{P}}$ denotes the indicator function for a predicate $\mathcal{P}$. The input features for P_2 on an arc $e_2 \in E_2$ are permitted to arbitrarily combine the information of $\mathbf{x}$ and the L_1's marginalized output features $\bar{\mathbf{h}}_1$, in addition to the local characteristics of the arc at hand e_2. In summary, an input feature for L_2 on an arc $e_2 \in E_2$ is of the form

$$f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big) \in \mathbb{R} \qquad (k_2 \in \mathcal{K}_2) \quad , \tag{4.5}$$

where $\mathcal{K}_2$ is the index set of the input feature types for L_2. To make the optimization procedure feasible, smoothness condition on any L_2's input feature is assumed with respect to all the L_1's output features, that is, $\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial h_{\langle 1,k_1',e_1\rangle}}$ is always guaranteed to exist for ${}^{\forall}k_1'$, e_1, k_2, e_2. For given input features $\mathbf{f}_{\langle 2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)$ and parameters $\theta_{\langle 2,k_2\rangle} \in \mathbb{R} \quad (k_2 \in \mathcal{K}_2)$, the probabilistic model for L_2 is defined as follows;

$$\begin{aligned}&P_2(\mathbf{y}_2|\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \\ &\quad \stackrel{\text{def}}{\equiv} \frac{1}{Z_2(\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)} \exp\big(\boldsymbol{\theta}_2 \cdot \mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)\big) \\ &\qquad\qquad (\mathbf{y}_2 \in \mathbf{Y}_2) \quad ,\end{aligned} \tag{4.6}$$

where $F_{\langle 2,k_2,\mathbf{y}_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big) \stackrel{\text{def}}{\equiv} \sum_{e_2 \in \mathbf{y}_2} f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)$ and Z_2 is the partition function of P_2, defined by

$$\begin{aligned}&Z_2(\mathbf{x};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2) \\ &\quad \stackrel{\text{def}}{\equiv} \sum_{\mathbf{y}_2 \in \mathbf{Y}_2} \exp\big(\boldsymbol{\theta}_2 \cdot \mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\big(\bar{\mathbf{h}}_1(\boldsymbol{\theta}_1)\big)\big) \quad .\end{aligned} \tag{4.7}$$

The definition of P_2 (4.6) reveals one of the most important points in this method. P_2 is viewed not only as the function of the ordinary direct parameters $\boldsymbol{\theta}_2$ but also as the function of $\boldsymbol{\theta}_1$, which represents the parameters for the L_1's model, through the intermediate variables $\bar{\mathbf{h}}_1$. So optimization procedure on P_2 may affect the determination of the values not only of the direct parameters $\boldsymbol{\theta}_2$ but also of the indirect ones $\boldsymbol{\theta}_1$.

If the result of L_1 is reduced to the single golden output $\tilde{\mathbf{y}}_1$, i.e. $P_1(\mathbf{y}_1|\mathbf{x}) = \delta_{\mathbf{y}_1=\tilde{\mathbf{y}}_1}$, the definitions above boil down to the formulation of the simple 1-best feed forward architecture.

4.3 Optimization Algorithm

In this section, I describe optimization procedure for the model formulated in the previous section. Let $\mathcal{D} = \{\langle \hat{\mathbf{x}}, \langle G_1, \hat{\mathbf{y}}_1 \rangle, \langle G_2, \hat{\mathbf{y}}_2 \rangle \rangle_m\}_{m=1,2,\cdots,M}$ denote annotated data for the supervised learning of the model. Here, $\langle G_1, \hat{\mathbf{y}}_1 \rangle$ is a pair of a DAG and correctly annotated successful sequence for L_1. The same holds for $\langle G_2, \hat{\mathbf{y}}_2 \rangle$. For given $\mathcal{D}$, I can define the conditional log-likelihood function on L_1 and L_2 respectively, that is,

$$\begin{aligned} &\mathcal{L}_1(\boldsymbol{\theta}_1; \mathcal{D}) \\ &\quad \stackrel{\text{def}}{\equiv} \sum_{\langle \hat{\mathbf{x}}, \hat{\mathbf{y}}_1 \rangle \in \mathcal{D}} \log\left(P_1(\hat{\mathbf{y}}_1|\hat{\mathbf{x}}; \boldsymbol{\theta}_1)\right) - \frac{|\boldsymbol{\theta}_1|}{2{\sigma_1}^2} \end{aligned} \tag{4.8}$$

and

$$\begin{aligned} &\mathcal{L}_2(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}) \\ &\quad \stackrel{\text{def}}{\equiv} \sum_{\langle \hat{\mathbf{x}}, \hat{\mathbf{y}}_2 \rangle \in \mathcal{D}} \log\left(P_2(\hat{\mathbf{y}}_2|\hat{\mathbf{x}}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\right) - \frac{|\boldsymbol{\theta}_2|}{2{\sigma_2}^2} \ . \end{aligned} \tag{4.9}$$

Here, ${\sigma_1}^2$, ${\sigma_2}^2$ are the variances of the prior distributions of the parameters. For the sake of simplicity, I set the prior distribution as the zero-mean uni-variance Gaussian. To optimize the both probabilistic models P_1 and P_2 jointly, I also define the joint conditional log-likelihood function

$$\mathcal{L}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}) \stackrel{\text{def}}{\equiv} \mathcal{L}_1 + \mathcal{L}_2 \ . \tag{4.10}$$

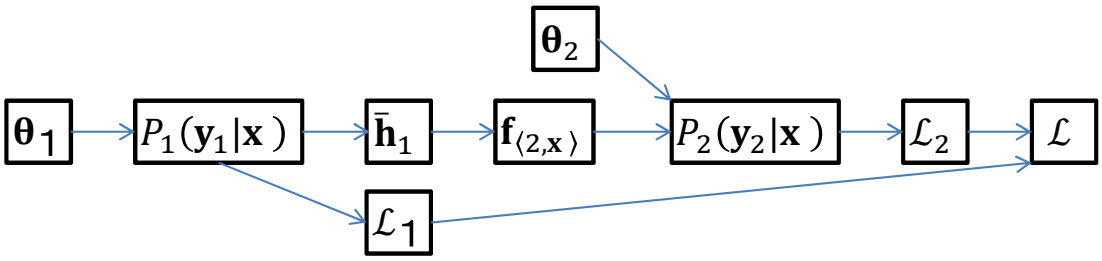

Figure 4.1: Computation Graph of the Proposed Model

The parameter values to be learned are the ones that (possibly locally) maximize this objective function, i.e.,

$$\langle \hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2 \rangle \stackrel{\text{def}}{\equiv} \underset{\langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \rangle}{\text{argmax}} \mathcal{L}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; \mathcal{D}) \quad . \tag{4.11}$$

"argmax" stands for the argument of a *local* maximum.

I employ gradient-based parameter optimization here. Optimization procedure repeatedly searches a direction in the parameter space which is ascendent with respect to the objective function, and updates the parameter values into that direction by small advances. Many existing optimization routines like steepest descent or conjugation gradient do that job only by giving the objective value and gradients on parameter values to be updated. So, the optimization problem here boils down to the calculation of the objective value and gradients on given parameter values.

Before entering the detailed description of the algorithm for calculating the objective function and gradients, I note the functional relations among the objective function and previously defined variables. A diagram is shown in Fig.4.1. This diagram illustrates the functional relations among the parameters, input and output feature functions, models, and objective function. The variables at the head of a directed arrow in the figure is directly defined in terms of the ones at the tail of the same arrow. The value of the objective function on given parameter values can be calculated in order of the arrows shown in the diagram. On the other hand, the parameter gradients are calculated step-by-step in reverse order of the arrows. The functional relations illustrated in the Fig.4.1 ensure some forms of the chain rule of differentiation among the variables. The chain rule is iteratively used to decompose the calculation of the gradients into a divide-and-conquer fashion. These two directions of stepwise computation are analogous to the *forward*

Algorithm 3 Gradient-based optimization of the model parameters

Input: $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$

Output: $\operatorname{argmax}_{\langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \rangle} \mathcal{L}$

1: **while** $\boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$ changes significantly **do**
2: calculate Z_1 by (4.2), $\bar{\mathbf{h}}_1$ by (4.3) with the F-B on G_1, and then $\mathcal{L}_1$ by (4.8)
3: calculate $\mathbf{f}_{\langle 2,\mathbf{x} \rangle}$ according to their definitions
4: calculate Z_2 by (4.7) with the F-B on G_2, and then $\mathcal{L}_2$ by (4.9) and $\mathcal{L}$ by (4.10)
5: calculate $\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1}$ and $\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2}$ by (4.13) with the F-B on G_1 and G_2, respectively
6: calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}}$ by (4.17) with the F-B on G_2, $\frac{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$, and them $\frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} = \frac{\partial \mathcal{L}}{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 1,\mathbf{x} \rangle}}{\partial \bar{\mathbf{h}}_1}$
7: calculate $\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1}$ by (4.19) with Algorithm 4
8: $\langle \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \rangle \leftarrow$ **update-parameters**$\Big(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \mathcal{L}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_1}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_2}\Big)$
9: **end while**

and *back propagation* for multilayer feedforward neural networks, respectively.

Algorithm 3 shows the whole picture of the gradient-based optimization procedure for our model. I first describe the flow to calculate the objective value for a given parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, which is shown from line 2 through 4 in Algorithm 3. The values of marginalized output features $\bar{\mathbf{h}}_{\langle 1,\mathbf{x} \rangle}$ can be calculated by (4.3). Because they are the simple marginals of features, the ordinary forward-backward algorithm (hereafter, abbreviated as "F-B") on G_1 offers an efficient way to calculate their values. Although nothing definite about the forms of the input features for L_2 is presented in this chapter, $\mathbf{f}_{\langle 2,\mathbf{x} \rangle}$ can be calculated once the values of $\bar{\mathbf{h}}_{\langle 1,\mathbf{x} \rangle}$ have been obtained. Finally, $\mathcal{L}_1$, $\mathcal{L}_2$ and then $\mathcal{L}$ are easy to calculate because they are no different from the ordinary log-likelihood computation.

Now I describe the algorithm to calculate the parameter gradients,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1} + \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1}, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_2} = \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2} \ . \tag{4.12}$$

Line 5 through line 7 in Algorithm 3 describe the gradient computation. The terms $\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1}$ and $\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2}$ in (4.12) become the same forms that appear in the ordinary CRF optimization, i.e., the difference between the empirical frequencies of the

features and the model expectations of them,

$$\begin{aligned}\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\theta}_1} &= \tilde{E}\big[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\big] - E_{P_1}\big[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\big] - \frac{|\boldsymbol{\theta}_1|}{{\sigma_1}^2} \ , \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_2} &= \tilde{E}\big[\mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\big] - E_{P_2}\big[\mathbf{F}_{\langle 2,\mathbf{y}_2,\mathbf{x}\rangle}\big] - \frac{|\boldsymbol{\theta}_2|}{{\sigma_2}^2} \ .\end{aligned} \tag{4.13}$$

These calculations are performed by the ordinary F-B on G_1 and G_2, respectively. Using the chain rule of differentiation derived from the functional relations illustrated in Fig.4.1, the remaining term $\frac{\partial L_2}{\partial \boldsymbol{\theta}_1}$ in (4.12) can be decomposed as follows;

$$\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial \bar{\mathbf{h}}_1} \cdot \frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1} \ . \tag{4.14}$$

Note that Leibniz's notation here denotes a Jacobian with the index sets omitted in the numerator and the denominator, for example,

$$\frac{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial \bar{\mathbf{h}}_1} \stackrel{\text{def}}{\equiv} \left\{ \frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial h_{\langle 1,k'_1,e_1\rangle}} \right\}_{k_2 \in \mathcal{K}_2, e_2 \in E_2, k'_1 \in \mathcal{K}'_1, e_1 \in E_1} \ . \tag{4.15}$$

And also recall that dot operators here stand for the inner product with respect to the index sets commonly omitted in both operands, for example,

$$\begin{aligned}&\frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_2} \cdot \frac{\partial \mathbf{f}_2}{\partial \bar{\mathbf{h}}_1} \\ &\qquad = \sum_{k_2 \in \mathcal{K}_2, e_2 \in E_2} \frac{\partial \mathcal{L}_2}{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}} \cdot \frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial \bar{\mathbf{h}}_1} \ .\end{aligned} \tag{4.16}$$

I describe the manipulation of each factor in the right side of (4.14) in turn. Noting $\frac{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}}{\partial f_{\langle 2,\grave{k}_2,\grave{e}_2,\mathbf{x}\rangle}} = \delta_{k_2=\grave{k}_2}\delta_{e_2=\grave{e}_2}$, each element of the first factor of (4.14) $\frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}$ can be transformed as follows;

$$\frac{\partial \mathcal{L}_2}{\partial f_{\langle 2,k_2,e_2,\mathbf{x}\rangle}} = \theta_{\langle 2,k_2\rangle} \sum_{\langle \hat{\mathbf{x}},\hat{\mathbf{y}}_2\rangle \in \mathcal{D}} (\delta_{e_2 \in \hat{\mathbf{y}}_2} - P_2(e_2|\hat{\mathbf{x}};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)) \ . \tag{4.17}$$

$P_2(e_2|\hat{\mathbf{x}};\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$, the marginal probability on e_2, can be obtained as a by-product of the F-B for (4.13).

As described in the previous section, it is assumed that the values of the second factor $\frac{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial \bar{\mathbf{h}}_1}$ is guaranteed to exists for any given $\boldsymbol{\theta}_1$, and the procedure for calculating them is fixed in advance. The procedure for some of concrete features is exemplified in the previous section.

From the definition of $\bar{\mathbf{h}}_1$ (4.3), each element of the third factor of (4.14) $\frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1}$ becomes

$$\frac{\partial \bar{h}_{\langle 1,k_1',e_1\rangle}}{\partial \theta_{\langle 1,k_1\rangle}} = h_{\langle 1,k_1',e_1\rangle} \mathrm{Cov}_{P_1(\mathbf{y}_1|\mathbf{x})}\left[\delta_{e_1\in\mathbf{y}_1},\, F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \quad . \tag{4.18}$$

There exists efficient D.P. to calculate the covariance value (4.18) (without going into that detail because it is equivalent to the one shown later in this chapter), and of course I can run such D.P. for $^{\forall}k_1' \in \mathcal{K}_1'$, $e_1 \in E_1$. However, the size of the Jacobian $\frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1}$ is equal to $|\mathcal{K}_1'| \times |E_1| \times |\mathcal{K}_1|$. Since it is too large in many tasks likely to arise in practice, I should avoid to calculate all the elements of this Jacobian in a straightforward way. Instead of such naive computation, if the values of $\frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}$ and $\frac{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial \bar{\mathbf{h}}_1}$ are obtained, then I can compute $\frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} = \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}} \cdot \frac{\partial \mathbf{f}_{\langle 2,\mathbf{x}\rangle}}{\partial \bar{\mathbf{h}}_1}$, and from (4.14) and (4.18),

$$\begin{aligned}
\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\theta}_1} &= \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_1} \cdot \frac{\partial \bar{\mathbf{h}}_1}{\partial \boldsymbol{\theta}_1} \\
&= E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle}\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \\
&\quad - E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle}\right] E_{P_1(\mathbf{y}_1|\mathbf{x})}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \quad ,
\end{aligned} \tag{4.19}$$

where $H'_{\langle 1,\mathbf{y}_1\rangle} \overset{\text{def}}{\equiv} \sum_{e_1\in\mathbf{y}_1} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$. In other words, $\frac{\partial \mathcal{L}_2}{\partial \theta_{\langle 1,k_1\rangle}}$ becomes the covariance between the k_1-th input feature for L_1 and the hypothetical feature $h'_{\langle 1,e_1\rangle} \overset{\text{def}}{\equiv} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$.

The final problem is to derive an efficient way to compute the first term of (4.19). The second term of (4.19) can be calculated by the ordinary F-B because it consists of the marginals of arc features. There are two derivations of the algorithm for calculating the first term. I describe briefly the both derivations.

One is a variant of the F-B on the expectation semiring proposed in Li et al.[17] First, the F-B is generalized to the expectation semiring with respect to the hypothetical feature $h'_{\langle 1,e_1\rangle}$, and by summing up the marginals of the feature vectors $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}$ on all the arcs under the distribution of the semiring, then I obtain the expectation of the feature vector $\mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}$ on the semiring potential. This expectation is equal to the first term of (4.19). [1]

Another derivation is to apply the automatic differentiation (AD)[28, 3] on the F-B calculating $E_{P_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]$. It exploits the fact that $\frac{\partial}{\partial \lambda} E_{P_1'}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]\Big|_{\lambda=0}$ is equal to the first term of (4.19), where $\lambda \in \mathbb{R}$ is a dummy parameter, and

[1] For the detailed description, see Li et al.[17] and its references.

$P'_1(\mathbf{y}_1|\mathbf{x}) \stackrel{\text{def}}{\equiv} \frac{1}{Z_1}\exp\Big(\boldsymbol{\theta}_1 \cdot \mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle} + \lambda H'_{\langle 1,\mathbf{y}_1\rangle}\Big)$. It is easy to derive the F-B for calculating the value $E_{P'_1}\left[\mathbf{F}_{\langle 1,\mathbf{y}_1,\mathbf{x}\rangle}\right]\Big|_{\lambda=0}$. AD transforms this F-B into another algorithm for calculating the differentiation w.r.t. λ evaluated at the point $\lambda = 0$. This transformation is achieved in an automatic manner, by replacing all appearances of λ in the F-B with a dual number $\lambda + \varepsilon$. The *dual number* is a variant of the complex number, with a kind of the imaginary unit ε with the property $\varepsilon^2 = 0$. Like the usual complex numbers, the arithmetic operations and the exponential function are generalized to the dual numbers, and the ordinary F-B is also generalized to the dual numbers. The imaginary part of the resulting values is equal to the needed differentiation. Anyway, these two derivations lead to the same algorithm, and the resulting algorithm is shown as Algorithm 4. [2]

The final line in the loop of Algorithm 3 can be implemented by various optimization routines and line search algorithms.

4.4 Experiment

I examined effectiveness of the method proposed in this chapter on a real task. The task is to annotate the POS tags and to perform base-phrase chunking on English sentences.

Base-phrase chunking is a task to classify continuous subsequences of words into syntactic categories. This task is performed by annotating a chunking label on each word [22]. The types of chunking label consist of "Begin-*Category*", which represents the beginning of a chunk, "Inside-*Category*", which represents the inside of a chunk, and "Other." Usually, POS labeling runs first before base-phrase chunking is performed. Therefore, this task is a typical interesting case where a sequence labeling depends on the output from other sequence labelers.

The data used for our experiment consist of English sentences from the Wall Street Journal. They are annotated by humans in the Penn Treebank project [20] and consist of 10948 sentences and 259104 words. I divided them into two groups, training data consisting of 8936 sentences and 211727 words and test data consisting of 2012 sentences and 47377 words. The number of the POS label

[2] For example, Berz[1] gives a detailed description of the reason why the dual number is used for this purpose.

Algorithm 4 Forward-backward Algorithm for Calculating Feature Covariances

Input: $\mathbf{f}_{\langle 1,\mathbf{x}\rangle}$, $\phi_{e_1} \stackrel{\text{def}}{\equiv} \exp\left(\boldsymbol{\theta}_1 \cdot \mathbf{f}_{\langle 1,e_1,\mathbf{x}\rangle}\right)$, $h'_{e_1} \stackrel{\text{def}}{\equiv} \frac{\partial \mathcal{L}_2}{\partial \bar{\mathbf{h}}_{\langle 1,e_1\rangle}} \cdot \mathbf{h}_{\langle 1,e_1\rangle}$

Output: $q_{k_1} = \mathrm{Cov}_{P(\mathbf{y}_1|\mathbf{x})}\left[H'_{\langle 1,\mathbf{y}_1\rangle},\, F_{\langle 1,k_1,\mathbf{y}_1,\mathbf{x}\rangle}\right] \qquad \left(^{\forall}k_1 \in \mathcal{K}_1\right)$

for $^{\forall}v_1 \in src(G_1)$, $\alpha_{v_1} \leftarrow 1$, $\alpha'_{v_1} \leftarrow 1$
for all $v_1 \in V_1$ in a topological order **do**
  prev $\leftarrow \{x \in V_1 \mid (x, v_1) \in E_1\}$
  $\alpha_{v_1} \leftarrow \sum_{x \in \mathrm{N}^-} \phi_{(x,v_1)} \alpha_x$, $\alpha'_{v_1} \leftarrow \sum_{x \in \mathrm{N}^-} \phi_{(x,v_1)} \left(h'_{(x,v_1)} \alpha_x + \alpha'_x\right)$
end for
$Z_1 \leftarrow \sum_{x \in snk(G_1)} \alpha_x$
for $^{\forall}v_1 \in snk(G_1)$, $\beta_{v_1} \leftarrow 1$, $\beta'_{v_1} \leftarrow 1$
for all $v_1 \in V_1$ in a reverse topological order **do**
  next $\leftarrow \{x \in V_1 \mid (v_1, x) \in E_1\}$
  $\beta_{v_1} \leftarrow \sum_{x \in \mathrm{N}^+} \phi_{(v_1,x)} \beta_x$, $\beta'_{v_1} \leftarrow \sum_{x \in \mathrm{N}^+} \phi_{(v_1,x)} \left(h'_{(v_1,x)} \beta_x + \beta'_x\right)$
end for
for $^{\forall}k_1 \in \mathcal{K}_1$, $q_{k_1} \leftarrow 0$
for all $(u_1, v_1) \in E_1$ **do**
  $p \leftarrow \phi_{(u_1,v_1)} \left(\alpha_{u_1} \beta'_{v_1} + \alpha'_{u_1} \beta_{v_1}\right) / Z_1$
  for $^{\forall}k_1 \in \mathcal{K}_1$, $q_{k_1} \leftarrow q_{k_1} + p f_{\langle 1,k_1,e_1,\mathbf{x}\rangle}$
end for

types is equal to 45. The number of the label types used in base-phrase chunking is equal to 23.

I compare the proposed method to two existing sequence labeling methods as baselines. The POS labeler is the same in all the three methods used in this experiment. This labeler is a simple CRF and learned by ordinary optimization procedure. One baseline method is the 1-best pipeline method. A simple CRF model is learned for the chunking labeling, on the input sentences and the most likely POS label sequences predicted by the already learned POS labeler. I call this method "CRF + CRF." The other baseline method has a CRF model for the chunking labeling, which uses the marginalized features offered by the POS labeler. However, the parameters of the POS labeler are fixed in the training of the chunking model. This method corresponds to the method proposed in Bunescu [2]. I call this baseline "CRF + CRF-MF" ("MF" for "marginalized

=== Node feature templates ===
Node is source
Node is sink
Input word on the same time slice
Suffix of input word on the same time slice, n characters $(n \in [1, 2, 3])$
Initial word character is capitalized†
All word characters are capitalized†
Input word included in the vocabulary of POS $T^{\dagger}$ $(T \in \{(\text{All possible POS labels})\})$
Input word contains numbers†
POS label‡
=== Arc feature templates ===
Tail node is source
Head node is sink
Corresponding ordered pair of POS labels‡

Table 4.1: List of feature templates.

features"). The proposed method is the same as "CRF + CRF-MF", except that the both labelers are jointly trained by the procedure described in Section 4.3. I call this proposed method "CRF + CRF-BP" ("BP" for "back propagation").

In "CRF + CRF-BP," the objective function for joint learning (4.10) is not guaranteed to be convex, so optimization procedure is sensible to the initial configuration of the model parameters. In this experiment, I set the parameter values learned by "CRF + CRF-MF" as the initial values for the training of the "CRF + CRF-BP" method. Feature templates used in this experiment are listed in Table 4.4. Note that, all node features are combined with the corresponding node label (POS or chunking label) feature, all arc features are combined with the feature of the corresponding arc label pair. Features marked by † in Table 4.4 are instantiated on each time slice in five character window. Features marked by ‡ are not used in POS labeler, and marginalized as output features for "CRF + CRF-MF" and "CRF + CRF-BP."

Although I only described the formalization and optimization procedure of the models with arc features, I use node features in the experiment.

	CRF + CRF	CRF + CRF-MF	CRF +CRF-BP
POS labeling	0.956	(0.956)	0.958
Base-phrase chunking	0.921	0.927	0.931

Table 4.2: Experimental result (F-measure)

Table 2 shows the result of the methods I mentioned. From Table 2, the proposed method outperforms two baseline methods on both POS labeling and chunking performance. Particularly, the improvement in POS labeling performance by the proposed method "CRF + CRF-BP" shows that optimization procedure provides some form of backward information propagation in comparison to "CRF + CRF-MF."

4.5 Conclusions for this Chapter

In this chapter, I adopt the method to weight features on an upper sequence labeling stage by the marginalized probabilities estimated by the model on lower stages. I also point out that the model on an upper stage is considered to depend on the model on lower stages indirectly. In addition, I propose optimization procedure that enables the joint optimization of the multiple models on the different level of stages. I perform an experiment on a real-world task, and our method outperforms existing methods.

I examined the effectiveness of the proposed method only on one task in comparison to just a few existing methods. In the future, I hope to compare our method to other competing methods like joint learning approaches in terms of both accuracy and computational efficiency, and perform extensive experiments on various tasks.

Chapter 5

Conclusions

In this dissertation, I presented a graph-theoretic formalization of Forward-Backward algorithm. Based on the graph-theoretic formalization, I also presented an algebraic framework for Forward-Backward algorithm. The combination of these frameworks constitutes the theoretical foundation for many generalizations of Forward-Backward algorithm, which is illustrated by examples. And as one of the main contributions of this dissertation, I showed a new kinds of generalization of Forward-Backward algorithm, which is introduced by the binomial convolution semiring. As shown in this dissertation, this generalization of F.B. subsumes many kinds of existing variants of F.B., and it not only offers the unified point of view of these variants of F.B., but also is insightful in induction of new variants of F.B.

I also presented a new application of F.B. algorithm on sequence labeling task, the multilayer sequence labeling. The multilayer sequence labeling is a very useful model for the case where a sequence labeling task depends on the output of another kind of sequence labeling task. Such cascaded tasks appear very frequently especially in NLP. I verified that this model is quantitatively useful in a particular NLP task. Due to the formalization and generalization of F.B. algorithm presented in this dissertation, the rigorous formalization and algorithm used for the parameter optimization of this model is immediately available.

As its name indicates, the generalization of F.B. algorithm shown in this dissertation focuses on sequential structures that can be encoded in trellises. There are many classes of data structures that have dynamic programming very similar to F.B. algorithm, including Inside-Outside algorithm on trees, message passing algorithm on cycle-free factor graphs, etc. It is a remaining question how to

adapt the formalization of F.B. presented in this dissertation to such wider range of data structures. Once we have had such formalization, the algebraic framework for F.B. presented in this dissertation immediately offers the methodology for computing much more complicated numerical values on data structures other than sequential ones.

It also remains an open question whether there are any other algebras useful to induce a variant of F.B. that is applied to problems in the real world. Recently, the amount and complexity of structured data handled in real world applications is drastically growing, and the algorithms utilized in that application also needs to shift with that. I would like to formalize such variants of F.B. by a not ad-hoc but systematic and unified way. For that purpose, we still need to specify the unseen nature of F.B. algorithm.

I keep looking for further sophisticated formalization and generalization of Forward-Backward algorithm.

Bibliography

[1] M. Berz. Automatic differentiation as nonarchimedean analysis. In *Computer Arithmetic and Enclosure*, pages 439–450, 1992.

[2] R.C. Bunescu. Learning with probabilistic features for improved pipeline models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 670–679, 2008.

[3] G.F. Corliss, C. Faure, and A. Griewank. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Verlag, 2002.

[4] C. Cortes, M. Mohri, A. Rastogi, and M. Riley. On the computation of the relative entropy of probabilistic automata. *International Journal of Foundations of Computer Science*, 19(01):219–242, 2008.

[5] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[6] J. Eisner. Expectation semirings: Flexible em for learning finite-state transducers. In *Proceedings of the ESSLLI workshop on finite-state methods in NLP*, 2001.

[7] J.R. Finkel, C.D. Manning, and A.Y. Ng. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, 2006.

[8] J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999.

[9] L. Huang. Dynamic programming algorithms in semiring and hypergraph frameworks. *Qualification Exam Report*, pages 1–19, 2006.

[10] L. Huang. Advanced dynamic programming in semiring and hypergraph frameworks. *COLING, Manchester, UK*, 2008.

[11] V.M. Ilic, M.S. Stankovic, and B.T. Todorovic. Entropy message passing. *Information Theory, IEEE Transactions on*, 57(1):375–380, 2011.

[12] Martin Jansche. Maximum expected f-measure training of logistic regression models. In *Proc. of HLT/EMNLP 2005*, pages 692–699, October 2005.

[13] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proc. of COLING-ACL 2006*, pages 209–216, July 2006.

[14] S. Kakade, Y. W. Teh, and S. Roweis. An alternate objective function for Markovian fields. In *Proc. of the ICML 2002*, volume 19, 2002.

[15] W. Kuich and A. Salomaa. Semirings, Automata, Languages. Number 5 in EATCS Monographs on Theoretical Computer Science, 1986.

[16] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML 2001*, pages 282–289, 2001.

[17] Zhifei. Li and Jason. Eisner. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 40–51. Association for Computational Linguistics, 2009.

[18] Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *HLT-NAACL 2007; Companion Volume, Short Papers*, pages 109–112, April 2007.

[19] Gideon S. Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proc. of ACL-08: HLT*, pages 870–878, June 2008.

[20] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):330, 1993.

[21] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[22] L.A. Ramshaw and M.P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge MA, USA, 1995.

[23] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS 17*, pages 1185–1192, 2004.

[24] C. Sutton, A. McCallum, and K. Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723, 2007.

[25] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*, 2003.

[26] Y. Tsuboi, Y. Unno, H. Kashima, and N. Okazaki. Fast newton-cg method for batch learning of conditional random fields. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[27] S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. of ICML 2006*, pages 969–976, 2006.

[28] RE Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):464, 1964.