EXPERIMENTS ON PHRASAL CHUNKING IN NLP USING EXPONENTIATED GRADIENT FOR STRUCTURED PREDICTION

by

Porus Patell

Bachelor of Engineering, University of Mumbai, 2009

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2011

APPROVAL

Name: Porus Patell

Degree: Master of Science

Title of Thesis: Experiments on Phrasal Chunking in NLP using Exponentiated Gradient for Structured Prediction

Examining Committee: Dr. Binay Bhattacharya
Chair

Dr. Anoop Sarkar,
Associate Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Greg Mori,
Associate Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Fred Popowich,
Professor, Computing Science
Simon Fraser University
Examiner

Date Approved: _____

Abstract

Exponentiated Gradient (EG) updates were originally introduced in (Kivinen and Warmuth, 1997) in the context of online learning algorithms. EG updates were shown by (Collins et al., 2008) to provide fast batch and online algorithms for learning a max-margin classifier. They show that EG can converge quickly due to multiplicative updates, and that EG updates can be factored into tractable components for structured prediction tasks where the number of output labels is exponential in the size of the input. In this project, we implement EG for a Natural Language Processing structured prediction task of phrasal chunking (finding noun phrases, and other phrases in text) and we compare the performance of EG with other discriminative learning algorithms that have state of the art results on this task.

Keywords: Natural Language Processing, Computational Linguistics, Machine Learning, Sequence Learning for NLP

To the Almighty Lord, to my wonderful family, and to my girl Deepa

"There is always light at the end of a tunnel"
- VRRP Spiritual Learning

Acknowledgments

This defense would not have been possible without the guidance and dedication shown by Dr. Anoop Sarkar, who not only has been a senior supervisor to me, but also a friend. I have thoroughly enjoyed working with him, enjoyed his witty humor and appreciated his significant knowledge on the research subject. Thank you Anoop for your guidance and effective supervision.

My special thanks goes to Dr. Greg Mori, my supervisor, who has always been a friend and mentor to me during my time at SFU. I was fortunate to have taken 2 courses with him, including the machine learning course in my first semester, which first got me interested in this wonderful area. A special thanks to Dr. Fred Popowich, my examiner, who not only is a very distinguished professor, but a very warm and humble person. Thank you Fred for providing many insightful comments for my defense report.

My time at SFU would not have been enjoyable without the company of some amazing individuals, friends that I shall keep with me for life. A special mention to you Rajib for being a great friend and for all those crazy ping-pong rivalries we had. My heartfelt thanks to every member of the Natlang group, who have always been so helpful in every possible way. A special thanks to Jenny and Harry, who have treated me like family throughout my year and a half stay at their wonderful home. Thank you everybody.

This report is dedicated to you mom and dad for loving me and raising me to be a strong, hard-working and caring individual. My love and thanks to you, Shahvir, Pouruchisti, Grandpa, Granny, Behram mama, Firoz mama, Sweetie and to all those wonderful people that have helped shape the person I am today. Finally, my love and thanks to you Deepa, your deep love and care towards me is far more than what I might have ever expected from any lady.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our project explores the use of a max-margin discriminative learning algorithm called Exponentiated Gradient(EG) in an online weight update setting for the structured prediction task of finding non-recursive phrasal chunks in text, where each word´s position within a sentence is associated with 1 out of 23 discrete phrasal tags that represent open and close brackets for each type of phrasal chunk. Thus, non-recursive chunking is treated as a tagging task (Skut and Brants, 1998); like for example part of speech tagging of words. We present the implementation details and experimental results associated with an online max-margin EG implementation used with the Viterbi decoding scheme, and provide a comparison with another state of the art discriminative learning algorithm for the phrasal chunking task, namely the mistake-driven, additive update based perceptron algorithm (Rosenblatt, 1958). The training and test data are taken from the CoNLL 2000 chunking shared task (Sang and Buchholz, 2000), created from the Penn Treebank Wall Street Journal corpus (WSJ) by keeping the non-recursive chunking structure of the Treebank. We evaluate our learning performance on an independent test set, where Viterbi decoding is used to decode the best hidden-state structure (sequence of labels), given a discriminatively learnt global weight vector defined over local structural parts identified in the training sentences. We then use a standard evaluation script to determine the overall accuracy (percentage of sentence tokens that receive the correct chunk tag), precision, recall and F-score measures, and per-phrase precision and recall accuracies on the independent test set.

1.1 Parts-of-Speech(POS) Phrasal Chunking

Phrasal chunking (Abney, 1991) consists of dividing a text into *syntactically correlated* non-recursive parts of words. Chunking helps provide a contextual structure to a sentence and is often used as an intermediate step towards full parsing. An example sentence taken from (Collins, 2003) to illustrate non-recursive phrasal chunking as a tagging task is "The burglar robbed the apartment.", where by using leftmost derivations of context free grammar(CFG) rules, we get "The burglar" as a bracketed noun phrase chunk (`NP`) in the sentence, and "robbed the apartment" as a bracketed verb phrase chunk (`VP`). Applying a per-word tagging structure to the above bracketed chunks, we get "The" as beginning noun phrase (`B-NP`) and "burglar" as an intermediate noun phrase(`I-NP`), "robbed" as beginning verb phrase (`B-VP`) and so on. Figure 1.1 shows an example sentence representation of how phrasal chunking can be represented as a tagging task.

The	burglar	robbed	the	apartment	.
[	NP]	[	VP]		[O]
B-NP	I-NP	B-VP	B-NP	I-NP	O

Figure 1.1: Phrasal chunking as a tagging task (Example taken from (Collins, 2003)).

POS chunking was first introduced as a shared task for CoNLL-2000. Training and test data for this task were taken from sections of the Wall Street Journal corpus (WSJ) with sections 15-18 as training data (211727 tokens) and section 20 as test data (47377 tokens). Figure 1.2 shows a gold standard chunking data representation for a particular training sentence taken from CoNLL-2000. The first column contains the current word, the second its part-of-speech tag and the third contains the truth chunk tag as derived from the WSJ corpus. A chunk tag includes the part of speech for the phrasal chunk (`NP` or `VP` etc.) and the type of chunk tag, for example `B-NP` for the first word of a `NP` and `I-NP` for a non-beginning word in a `NP`.

Using the gold standard chunking information across all training sentences, supervised learning techniques can be used to learn a weight vector, which after the training phase can recognize POS-chunk segmentations of the test data as well as possible.

Confidence	NN	B-NP
in	IN	B-PP
the	DT	B-NP
pound	NN	I-NP
is	VBZ	B-VP
.	.	.
.	.	.

Figure 1.2: Phrasal chunking gold standard data representation (Example taken from CoNLL-2000 data).

1.2 Motivation

Large-margin classifiers like Support Vector Machines have consistently out performed traditional classifiers in many computing applications. By enumerating over all possible output candidates and maximizing the decision boundary between the truth and non-truth output candidates, large-margin classifiers have provided a successful alternative framework to traditional probability based classifiers. Traditional max-margin classification involves identifying a decision boundary between a finite set of classes. This is a tractable learning inference as it involves iterating over discrete output labels associated with the training data.

Most Natural Language classification tasks can be classified into variants of tagging or parsing tasks on structured data, where the primary aim is to learn a *hidden contextual structure* that best describes the observed sequence. Natural language classification is synonymous with structured prediction (Collins, 2002), where a given input structure like a sentence will not have a single output label, but rather a structured collection of finite labels that jointly make up an output generation. The structured collection of labels may vary in their generation order, thus leading to a large number of possible candidate outputs for a given input sequence. Figure 1.3 shows how multiple structured outputs can get generated from a single input sentence.

The number of possible output structures that can be formed depends on the size of the input sequence. For larger input sequences, the number of structured outputs might be exponential. Intractability of possible exponential number of candidate outputs for many structured prediction Natural Language tasks makes working with margin-maximization

Sentence	**Truth output**	**Candidate output 1**	**Candidate output 2**	. .
Confidence	B-NP	B-NP	B-VP	. .
in	B-PP	I-NP	I-VP	. .
the	B-NP	B-VP	B-PP	. .
pound	I-NP	B-ADJP	B-NP	. .
is	B-VP	I-VP	B-VP	. .
.	.	.	.	. .
.	.	.	.	. .

Figure 1.3: Multiple structured outputs for a sentence.

techniques for these tasks infeasible, as margin-maximization requires enumeration over all output structures to determine the max-margin candidate. (Taskar et al., 2004a) showed how to decompose the max-margin updates based on parts of the structured prediction. The EG algorithm expands on this idea further using multiplicative updates to obtain a max-margin classifier.

Our project expands on the previous work of (Bartlett et al., 2004), (Globerson et al., 2007), (Collins et al., 2008), who demonstrated how max-margin classifiers like Exponentiated Gradient can be practically used for structured prediction by working with tractable *local parts* that can be identified in a sequence-candidate pair, rather than enumerating over all structured outputs. For the Phrasal Chunking task, the local structural parts include observed unigram parts and contextual bigram parts defined at each sentence index.

The online weight update process in EG is shown to have a *guaranteed convergence* (Bartlett et al., 2004), and *bounded convergence times* were theoretically proven and experimentally demonstrated for both the batch and randomized online weight update procedures for the structured prediction task of dependency parsing (Globerson et al., 2007), (Collins et al., 2008). Our project aims to provide an implementation of the above work for tagging based Natural Language tasks, which include chunking and word segmentation. We hope that our work with the EG algorithm for the phrasal chunking task, implemented with a tractable part-based learning inference, would serve as a prototype for tagging based learning applications across the Natural Language spectrum. An example includes finding the best word segmentation for unsegmented text in languages like Chinese and Vietnamese, where meaningful segmentation of text is needed before it can be translated into another language like English.

Chapter 2

Background

2.1 Supervised Learning

Supervised learning involves prediction of a meaningful label for unknown data. It includes generative and discriminative learning techniques. In the next section, we introduce the concept of generative models for supervised learning. Discriminative learning will be discussed later in section 2.3.

2.1.1 Generative Models

For sequence data learning, probabilistic generative models like Hidden Markov Models can be effectively used. A first order HMM defines a joint distribution $P(\mathbf{x}, \mathbf{y})$, given a sequence input $\mathbf{x} = x_1 \dots x_n$ and the candidate sequence $\mathbf{y} = y_1 \dots y_n$ and makes a prediction at each sequence index i by calculating $P(y_i|y_{i-1}, x_i)$, and then selecting the most likely generated label y_i at i. The sequence y is modeled using conditional independence assumption at each step as $P(\mathbf{y}) = P(y_1 \dots y_n) = P(y_1) * P(y_2|y_1) \dots P(y_n|y_{n-1})$.

Sequence decoding in generative models involves finding the most likely sequence of tags $y_1 \dots y_n$ where likelihood is computed using transition and emission probabilities. An example of generative decoding is Viterbi decoding which will be discussed in section 2.4, which uses the conditional independence assumption at each step to decode the most likely sequence of labels. A disadvantage of using generative models is that the piecewise learning of the feature weights cannot adequately model all complex dependencies that might exist in data. It has been observed that discriminative models perform better than generative

models in most machine learning tasks.

We illustrate the concept of learning using generative models with the following simple example of a Hidden Markov Model as shown below.
Consider a word segmentation example $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is the input sequence `ABCDE` and $\mathbf{y}$ is the output labeling `01011`:

```
A B C D E
0 1 0 1 1
```

In this example, the tagging `01011` means that the output is a sequence of three words: `AB`, `CD` and `E`.

We will write down the equations and pseudo-code for an HMM representation, which is a simplified form of the full global linear model which contains several features rather than a single emission feature. In an HMM, each part is either a transition tuple $\langle a_{0,1}, 1\rangle$ for character location 1 representing a transition from state 0 to 1, or an emission tuple $\langle b_1(B), 1\rangle$ for character location 1 representing the emission symbol `B` from state 1. We denote the transition weight for $a_{i,j}$ as $w_{i,j}$ for transition from index i to j and similarly emission weight for $b_j(o)$ is $w_{j,o}$ for symbol o at state j.

Using HMM's, the joint probability P(`ABCDE`, `01011`) is modeled such that at each sequence index, a tag's decoding is dependent on the previous index's decoded tag and the observed symbol sequence.

2.2 Feature Vector Representation

The simplified assumptions of a generative framework do not provide enough scope to model detailed relationships amongst the features. For the phrasal chunking task, we adopt a feature vector representation first proposed by (Ratnaparkhi, 1996), (Berger et al., 1996) where a log-linear framework is used to model the features. Log-linear models compute a conditional dependence of a tag $t_\imath$ at a sequence index $\imath$, based on its previously observed sequence history h. This sequence history might include the previously decoded tag sequence $[t_1 \dots t_{\imath-1}]$, the entire observed word sequence $[o_1 \dots o_n]$ for a sequence of length n, and the current index $\imath$. Using log-linear models, the conditional $(t_\imath|h)$ relation at all indices $\imath$ of a

sequence can be formulated as:

$$P(t_1 \dots t_n | o_{[1:n]}) = \prod_{\imath=1}^{n} P(t_\imath | t_1 \dots t_{\imath-1}, o_{[1:n]}, \imath)$$

To model local structural features conditionally independent of each other, the tagged-sequence needs to be modeled as a Markov network. For a Markov based structured prediction framework, each bigram part dependency can be represented as a 2-clique graph $(\mathtt{a}, \mathtt{b}, \mathtt{u}, \mathtt{v})$, where `a` and `b` represent the nodes of the graph (observed words) and `u` and `v` represent the discrete tags or labels for `a` and `b` respectively. Hence, bigram (contextual) and unigram (observed) feature-parts can be independently identified in windowed regions defined at each training sequence index (Collins, 2002).

Another important consideration while modeling the feature parts for structured prediction is that they are not locally normalized to form probabilities, but rather occurrence counts of the parts in a given structured sequence are maintained. The part occurrences are globally normalized only during the final weight update phase. The idea of global normalization was first introduced through Conditional Random Fields (CRF) by (Lafferty et al., 2001), where individual probabilities were normalized using a globally computed marginal across all local part counts. This was done because locally normalized sequence models suffered from the label bias problem during sequence decoding. (Collins, 2002), (Taskar et al., 2004a) showed how structured prediction learning could effectively use globally normalized Markov networks to adequately model complex dependencies amongst local feature parts r that exist in the training sequences.

Thus, any structured sequence x_i can be efficiently represented in terms of its tractable local feature-part occurrences $\phi(x_i, r)$. For an input sequence x_i with its corresponding structured output y, we can write the global feature-vector for the sequence-candidate pair (x_i, y) as a collection over its tractable local part counts.

$$\mathbf{\Phi}(x_i, y) = \sum_{r \in R(x_i, y)} \phi_r(x_i, y)$$

where R can be thought of as a d-dimensional representation over all local feature parts r and $R(x_i, y)$ can be thought of as a representation over all r that lie in (x_i, y). The vector $\mathbf{\Phi}(x_i, y)$ can be thought of as indexed by feature parts $r \in R(x_i, y)$ and whose value for each index is the count of occurrences of r across all indices of the sequence-output pair, represented as $\phi_r(x_i, y)$.

For example, for the HMM example previously introduced, considering a start sequence marker S_{-1}, we can write:

$$\begin{aligned}\mathbf{\Phi}(x, y) = \quad & \phi(\texttt{ABCDE}, \langle a_{S_{-1},0}, 1\rangle) + \phi(\texttt{ABCDE}, \langle b_0(A), 1\rangle) \\ & + \phi(\texttt{ABCDE}, \langle a_{0,1}, 2\rangle) + \phi(\texttt{ABCDE}, \langle b_1(B), 2\rangle) \\ & + \phi(\texttt{ABCDE}, \langle a_{1,0}, 3\rangle) + \phi(\texttt{ABCDE}, \langle b_0(C), 3\rangle) \\ & + \phi(\texttt{ABCDE}, \langle a_{0,1}, 4\rangle) + \phi(\texttt{ABCDE}, \langle b_1(D), 4\rangle) \\ & + \phi(\texttt{ABCDE}, \langle a_{1,1}, 5\rangle) + \phi(\texttt{ABCDE}, \langle b_1(E), 5\rangle)\end{aligned}$$

2.2.1 Loss in structured prediction

The concept of a structured sequence consisting of many globally normalized local parts can be extended to loss computations for structured sequences (Taskar et al., 2004a). For a input sequence x_i, the cost of proposing a part-rule r is called the loss:

$$l_{i,r} = \begin{cases} 0 & \text{if } r \text{ is not in truth} \\ 1 & \text{otherwise} \end{cases}$$

For each example x_i in the training data and for each candidate output y, the global loss for a sequence-candidate pair can be defined as a sum over its localized losses.

$$L(x_i, y) = \sum_{r \in R(x_i, y)} l_{i,r}$$

For the previously introduced HMM example, If we consider the labeling `01011` to be the truth labeling for `ABCDE`, the tagged bigram feature-parts identified in the truth output labeling are:

$$[\langle B : S_{-1}, 0\rangle,\ \langle B : 1, 0\rangle,\ \langle B : 0, 1\rangle,\ \langle B : 1, 1\rangle]$$

Their respective part counts are:

$$\phi(\texttt{ABCDE}, \langle B : S_{-1}, 0\rangle) = 1,\ \phi(\texttt{ABCDE}, \langle B : 0, 1\rangle) = 2,\ \phi(\texttt{ABCDE}, \langle B : 1, 0\rangle) = 1,\ \phi(\texttt{ABCDE}, \langle B : 1, 1\rangle) = 1$$

For all such truth bigram parts, $l_{i,r} = 0$. The remaining bigram parts $[\langle B : S_{-1}, 1\rangle,\ \langle B : 0, 0\rangle]$ that may get proposed in the sequence, but do not lie in the truth, have $l_{i,r} = 1$.

2.3 Linear Discrimination

In discriminative supervised classification, the task is to learn a parametric function $f : X \rightarrow Y$ from a set of N independent and identically distributed instances $\mathtt{S} = \{(x_i, y_i)\}_{\mathtt{i=1}}^{\mathtt{N}}$, drawn

from a fixed distribution $D_{X \times Y}$ and $y_i = truth(x_i)$. The classification mapping function `f` is typically selected from some parametric family which might be linear, polynomial or exponential. In linear discriminative classification we attempt to find a linear discriminative boundary between two classes as $f(x) = Sign(\mathbf{\Phi}(x) \cdot \mathbf{w})$, where $\mathbf{\Phi} : X \rightarrow R^d$, and parameter vector $\mathbf{w} \in R^d$.

The inference problem in flat or single-label discriminative classification involves assignment of one of $Y = \{Y_1, \ldots, Y_k\}$ discrete labels to a given input instance x_i. In joint-label classification such as structured prediction tasks like sequence labeling and parsing, each candidate label y for a input instance x_i has a rich internal structure consisting of small structural parts that are identified at local regions of the sequence.

2.3.1 Phrasal Chunking as a Discriminative Sequence Learning Task

Phrasal chunking can be modeled as a sequence learning problem where for an input sentence x_i, each generated candidate y has a rich internal structure consisting of a hidden sequence of states (non-recursive phrasal chunks). The internal structure may be a text segmentation, a state sequence labeling or parse tree structure.

We use a notational framework given in (Collins, 2002) to describe the structured prediction learning inference. If X represents a set over all training input sentences and Y represents a set over the structured output candidates, then a structured prediction discriminative classification task can be modeled as a joint sequence-candidate learning task that uses a feature vector $\mathbf{\Phi} : (X \times\ Y) \rightarrow R^d$, and parameter vector $\mathbf{w} \in R^d$.

In sequence learning tasks, for a given sequence-candidate pair (x_i, y), calculating an inference over all candidate output generations $y = \text{GEN}(x_i)$ might not be possible due to the exponential number of outputs that a sequence might generate. Hence, the learning inference problem is often modeled over the local feature-parts r that can be identified in a given sequence. Though the number of parts are large, they are tractable and this avoids learning over a possible exponential number of outputs.

Given an observed sequence instance x_i and a candidate output y, a global feature vector $\mathbf{\Phi}(x_i, y)$ is defined as a structure that maintains counts $\phi_r(x_i, y)$ over local part r occurrences in (x_i, y) (Collins, 2002). The linear discriminative score assigned to a structured candidate

$y \in \text{GEN}(x_i)$, based on enumeration of its local feature parts is defined as:

$$\text{Score}(x_i, y) = \mathbf{w} \cdot \mathbf{\Phi}(x_i, y) \tag{2.1}$$

$$= \mathbf{w} \cdot \sum_{r \in R(x_i, y)} \phi_r(x_i, y) \tag{2.2}$$

$$= \sum_{r \in R(x_i, y)} w_r * \phi_r(x_i, y) \tag{2.3}$$

The best sequence y is the one with the highest score among all $y \in \text{GEN}(x_i)$.

2.4 Viterbi Decoding

Decoding refers to the process of finding the best candidate output given an input instance x_i. In structured prediction tasks, the number of possible candidate labeled outputs y for an input sequence x_i can be exponential because of the combinatorial explosion in the possible joint-labels that can be formed. Thus identifying the best candidate $\arg\max y$ from the exponential number of generated outputs $\text{GEN}(x_i)$ becomes infeasible.

A solution to the above inference problem is found by using a first order generative Hidden Markov Model inference (Rabiner, 1989) (Viterbi decoding) which conditions the current state sequence on previously decoded state sequence and observed input sequence. The Viterbi decoding scheme for structured data (Collins, 2002) uses the HMM conditional independence assumption for hidden states at each sequence index to determine a max-likelihood path labeling up to that symbol. At any stage, the Viterbi algorithm examines all possible paths leading to a state and only the most likely path per state is kept and used in exploring the most likely path toward the next state. The best decoded state sequence is obtained when the entire input sequence is parsed. The feature weights in $\mathbf{w}$ represent probabilistic values that guide the decoding process. At the end of the algorithm, by traversing backward along the most likely path, the corresponding best state sequence can be found.

The structured prediction hypothesis to determine the highest scoring candidate $f_{\mathbf{w}}(x_i)$

is defined as:

$$f_{\mathbf{w}}(x_i) = \underset{y \in GEN(x_i)}{\arg\max}\ \text{Score}(x_i, y) \tag{2.4}$$

$$= \underset{y \in GEN(x_i)}{\arg\max}\ \mathbf{w} \cdot \mathbf{\Phi}(x_i, y) \tag{2.5}$$

$$= \underset{y \in GEN(x_i)}{\arg\max} \sum_{r \in R(x_i, y)} w_r * \phi_r\,(x_i, y) \tag{2.6}$$

Equation 2.6 shows that the best predicted candidate structure $f_{\mathbf{w}}(x_i)$ is determined by selecting the best score across scores generated by all candidate sequences GEN(x_i). Each discriminative score is calculated as a inner product between the global feature vector $\mathbf{\Phi}(x_i, y)$ and the global parametric weight vector $\mathbf{w}$, which helps guide the decoding process to determine the best candidate. Viterbi decoding ensures that the best predicted candidate is determined in polynomial time without the need to iterate over all candidate outputs, which might lead to a possible exponential inference.

2.5 Perceptron Learning Algorithm

The perceptron algorithm (Rosenblatt, 1958) is a linear discriminative, online, mistake-driven additive update algorithm that generates an output for an input based on a weighted set of features. The perceptron algorithm updates the feature weights after each parse through a training sentence (online weight updates). The perceptron and its variants have achieved state of the art accuracies in many Natural Language learning tasks. It is an intuitive, easy to update learning algorithm that tends to reach convergence fairly quickly due to its online weight update procedure.

Online learning is an important component of the perceptron algorithm. Online learning implies that the weight vector gets updated after iterating over a single training sentence, and that after each such update, the loss function is guaranteed to monotonically descend towards the global optimal solution (if it exists). Online algorithms are very useful in machine learning applications as they tend to reach convergence fast. This is because online algorithms tend to take short update steps towards the global optimal solution, and hence tend not to overshoot the global minimum.

In the batch update procedure, however, a guaranteed step towards convergence is possible only when all training examples have been iterated over, and then the weight vector

update is performed. Hence, batch update algorithms are slower to reach convergence than online algorithms. Unlike online algorithms, batch update algorithms tend to overshoot the global minimum due to the large step descents that are enforced by values collected over all training data. An example of a batch implementation of the EG algorithm with slow convergence was outlined by (Song and Sarkar, 2009) for their Chinese Word segmentation task.

Algorithm 1 shows a standard perceptron algorithm used in Natural Language applications. A discriminative score $\mathbf{w} \cdot \mathbf{\Phi}(x_i, y)$ is assigned to a possible output candidate y of x_i. The adaptation of the perceptron algorithm in a discriminative setting to Natural Language structured learning (Collins, 2002) is based on whether parts that are identified in the best predicted candidate $\arg\max y \in \text{GEN}(x_i)$ (obtained using Viterbi decoding) lie in the truth or not. For each training iteration t and each training sentence x_i, the above check is performed and feature weight values are incremented in the weight vector if they lie in the truth or decremented from the weight vector if they do not lie in the truth (line 8 of the algorithm). Accuracy of a prediction system depends on its capability to effectively learn this weight vector through an iterative learning process.

Algorithm 1 Perceptron algorithm

1: **Input**: Training set (x_i, y_i) for $i = 1, \ldots N$
2: **Initialize**: $\mathbf{w} = [0, 0, \ldots, 0]$
3: **for** $t = 1, 2, \ldots, (T + 1)$ **do**
4: **for** $i = 1, 2, \ldots, N$ **do**
5: Calculate: $y'_i = \arg\max_{y \in GEN(x_i)} \mathbf{w} \cdot \mathbf{\Phi}(x_i, y)$
6: **if** $y_i \neq y'_i$ **then**
7: Set: $\mathbf{w} = \mathbf{w} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y'_i)$
8: **end if**
9: **end for**
10: **end for**
11: **Output**: $\mathbf{w}$

A possible drawback of the perceptron algorithm is that since the feature-part weights are additively updated, they are not normalized to give a probability distribution. Also, in certain learning scenarios, the perceptron can take longer to converge due to the ordering of the data.

2.6 Max-Margin Classification

Large-margin discriminative classifiers do not adopt a probabilistic framework to learn a weight vector from the data. Instead, they try to maximize a margin between the score of the truth segmentation and the score of the best decoded segmentation. To determine the best candidate, they enumerate all possible candidate generations and calculate a margin score over each, which can lead to an exponentially large inference. Large margin classifiers are well suited to handle data defined with a large number of features since they do not operate in the traditional data-space, instead they transform the data to a dual feature-space per example formulation.

The distance from the discriminative decision surface to the closest data point determines the margin of the classifier. The classifer can be a linear classifer (as in a linear SVM) or the classifier can use an exponential function as in EG. The separating decision surface can be fully specified by a subset of instances which lie closest to the boundary. These boundary supporting points are referred to as the support vectors, leading to the generic max-margin discriminative classifer called Support Vector Machine (SVM).

For a given sequence-candidate pair (x_i, y), the margin $\mathrm{M}(x_i, y)$ is defined as the difference between the discriminative scores (as defined in equation 2.3) of the candidate ($\mathrm{S}(x_i, y)$) with that of the truth ($\mathrm{S}(x_i, y_i)$). The size of the margin provides a confidence measure for rejecting incorrect candidates, where a larger margin between the truth generation and other candidate generations indicates a greater confidence in rejecting incorrect generations. The margin formulation for sequence learning tasks is shown in equation 2.9.

$$\begin{aligned} \mathrm{M}(x_i, y) &= \mathrm{S}(x_i, y_i) - \mathrm{S}(x_i, y) && (2.7)\\ &= (\mathbf{w}\cdot \mathbf{\Phi}(x_i, y_i)) - (\mathbf{w}\cdot \mathbf{\Phi}(x_i, y)) && (2.8)\\ &= \mathbf{w}\cdot (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)) && (2.9) \end{aligned}$$

For max-margin models, learning can be framed as minimization of a convex regularized loss function (Rosset et al., 2004). To account for outliers in max-margin classification, the addition of non-negative slack variables ξ_i per example x_i allows for an increase in the global margin by paying a local penalty on some outlying examples. The large margin regularization constant $C > 0$ dictates the capability of an update to affect a change to the decision margin. A larger C will provide a tighter control on the margin's variability. As $C \to \infty$, it approaches a large-margin solution.

The loss based optimization problem for margin maximization as formulated in (Bartlett et al., 2004) is:

$$\mathbf{w}^* = \arg\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + \mathrm{C}\sum_i \xi_i$$

subject to the constraints

$\forall i, \forall y \in \mathrm{GEN}(x_i)$, $\mathbf{w}\cdot\boldsymbol{\Phi}(x_i, y) \geq \mathrm{L}(x_i, y) - \xi_i$

and $\forall i$, $\xi_i \geq 0$

where $\mathrm{L}(x_i, y)$ is the total loss and $\mathrm{L}(x_i, y) = 0$ for $y = y_i$.

If $M(x_i, y)$ is the dual margin and $\boldsymbol{\alpha}^*$ is the corresponding arg max of the dual problem, then the relationship between the primal $\mathbf{w}^*$ and the dual $\boldsymbol{\alpha}^*$ is (Bartlett et al., 2004):

$$\mathbf{w}^* = C\sum_{i,y} \boldsymbol{\alpha}^* \left(\mathrm{L}(x_i, y) - \mathrm{M}(x_i, y)\right)$$

2.7 Chapter Summary

Chapter 2 introduced us to the idea of inference learning on Natural Language based structured prediction tasks such as the sequence learning task of finding non-recursive phrasal chunks in text. Linear discrimination techniques and a globally normalized Markov network based feature generation framework were also discussed.

We then introduced a discriminative additive update based learning algorithm called the perceptron, used with the Viterbi decoding scheme. The concept of large-margin classification and its advantages was also introduced. A more detailed look at max-margin structured learning would be seen in the next chapter on the Exponentiated Gradient learning algorithm.

Chapter 3

Exponentiated Gradient

The Exponentiated Gradient algorithm (Kivinen and Warmuth, 1997) is a linear discriminative algorithm that uses margin-maximization along with multiplicative updates for loss re-estimation. The updates are performed on a dual-space variable and that variable is then used to update the weight values associated with the discriminative features. Both batch and online versions of EG have a guaranteed theoretical convergence (Bartlett et al., 2004). Also, the weight update process in EG is well suited for online learning and bounded learning times have been demonstrated for both batch and randomized online weight update procedures for the structured prediction task of dependency parsing (Globerson et al., 2007), (Collins et al., 2008).

An important aspect of the EG algorithm is that the dual-space parameters are multiplicatively updated using exponential penalty values. For the candidate generation based EG (section 3.2), the multiplicative update procedure ensures that the discriminative strength of each generated candidate y for each x_i is represented as a probabilistic distribution over the dual variables. Hence, the discriminative-strength distribution for generated candidates of each training sentence x_i, is maintained by a dual parameter $\boldsymbol{\alpha}_i$ for that sentence. An advantage of maintaining a probability distribution is that candidates with poor discriminating capabilities are driven to near-zero values and can be ignored in successive margin boosting iterations for a given sequence x_i. In the part enumeration based EG (section 3.3), a globally normalized dual-space marginal is computed for all structural feature parts r identified in x_i. The normalized marginals help in identifying irrelevant features in the training run so that they can be driven down towards a zero value. The EG algorithm uses a probability distribution based multiplicative update procedure on the dual variables, and

is in contrast to the perceptron based additive update mechanism, which is done directly on the values of **w**. Due to additive update mechanism in perceptron, a distribution cannot be maintained over the features and hence even highly irrelevant features continue to be accounted for throughout the entire training phase, making convergence slower.

3.1 EG Loss

The EG formulation as proposed in (Collins et al., 2008) for structured prediction tasks, uses an exponential hinge loss on a combination of 2 distinct loss generations to optimize the global parameters: a margin based loss $\nabla_{i,y}$ that uses margin-maximization for candidate discrimination and a Hamming loss $L(x_i, y_i, y)$ over all local parts of a sequence-candidate pair (x_i, y). The combined loss value for a candidate y helps to provide a rejection confidence measure for incorrect candidate generations.

The margin based hinge loss is $\nabla_{i,y} = (1 - M(x_i, y))_+$ such that

$$(1 - M(x_i, y))_+ = \begin{cases} (1 - M(x_i, y)) & \text{if } y \neq y_i \\ 0 & \text{if } y{=}y_i \end{cases}$$

where

$$\mathrm{M}(x_i, y) = \mathbf{w} \cdot (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)$$

The exponential loss formulation ensures that significantly larger penalties can be applied to non-truth candidates, thus ensuring a faster rate of descent towards the global minimum solution. However, one thing to be noted with exponential updates is that they might force the algorithm to descend very quickly, and as a result it might not provide sufficient learning for the global parameters. This happens due to the large multiplicative exponential penalties they impose. Hence, choosing the right exponent for a particular problem is very important.

Unlike the max-margin loss which iterates directly over the candidates, the Hamming loss is defined over local parts that make up a particular candidate. As previously outlined in section 2.2.1, a local loss $l_{i,r}$ of value 1 is assigned to a part r that is proposed for a particular candidate y, but which does not lie in the truth output y_i. If the part lies in the truth output y_i, then it gets a loss of 0. We define $L(x_i, y)$ to be the total Hamming loss for a candidate y and this loss can be computed as a sum over all local losses for each structural

part r that gets proposed for that particular candidate.

$$L(x_i, y) = \sum_{r \in R(x_i, y)} l(i, r)$$

.

Margin based maximization ensures that candidates with generation scores $\mathbf{\Phi} \cdot \mathbf{w}$ greater than or closest to the truth's score get pushed away from the truth (decision margin maximization). These candidates represent the best decoded candidate for a sequence, and can be found using Viterbi decoding by decoding with the trained weight vector $\mathbf{w}$. The remaining candidates represent weakly discriminative candidates which can be discarded from further consideration during the learning phase.

Hence, a candidate y is considered to be a support vector if it gives the highest value for the margin and least Hamming distance from the truth, or the least overall loss on $\eta C\left(L(x_i, y) - M(x_i, y)\right)$, where η is the learning rate and C is the margin regularization constant. Combining the Hamming and margin based losses, the total exponential loss for each generated candidate is $\exp\{\eta C\left(L(x_i, y) - M(x_i, y)\right)\}$. The candidate with the best loss is defined as:

$$\underset{y \in GEN(x_i)}{\arg\max} \exp\{\eta C\left(L(x_i, y) - M(x_i, y)\right)\}$$

If we write the above loss as $\nabla_{i,y} = C(L(x_i, y) - M(x_i, y))$, then the update for each dual parameter $\alpha_{i,y}$ for a training sentence x_i and its candidate generation y, is as given in equation 3.1.

$$\alpha'_{i,y} = \frac{\alpha_{i,y} \exp\{\eta \nabla_{i,y}\}}{\sum_{y'} \alpha_{i,y'} \exp\{\eta \nabla_{i,y'}\}} \tag{3.1}$$

We can express the above loss $\nabla_{i,y}$ purely in terms of the local parts r that exist in a given candidate as shown in equation 3.3 (Taskar et al., 2004a):

$$L(x_i, y) - M(x_i, y) = \left(\sum_{r \in R(x_i, y)} l_{i,r} \right) - \left(\mathbf{w} \cdot \mathbf{\Phi}(x_i, y_i) - \sum_{r \in R(x_i, y)} w_r * \phi_r(x_i, y) \right) \tag{3.2}$$

$$= \left(\sum_{r \in R(x_i, y)} (l_{i,r} + w_r * \phi_r(x_i, y)) \right) - (\mathbf{w} \cdot \mathbf{\Phi}(x_i, y_i)) \tag{3.3}$$

The term in the first bracket of equation 3.3 represents the empirical score while the term in the second bracket represents the truth score. The difference between the two represents the prediction loss that needs to be minimized, which is how the weight vector $\mathbf{w}$ gets updated so that the loss reaches a minimum.

Extending the above local loss $l_{i,r}$ formulation to a dual part-based loss score formulation $\theta_{i,r}$, the updates for $\alpha_{i,y}$ can also be written as:

$$\alpha'_{i,y} = \frac{\exp \sum_{r \in R(x_i,y)} \theta_{i,r}}{\sum_{y'} \exp \left(\sum_{r \in R(x_i,y')} \theta_{i,r} \right)}$$

where $\theta_{i,r}$ represents a dual loss-score assigned to each part r in proposed (x_i, y).

The part based EG loss formulation will be seen later in section 3.3. The above relation however, does provide us with an idea as to how the candidate EG and part based EG loss formulations are related to each other. The dual in the candidate enumeration formulation can be expressed in terms of the dual in the part based formulation and vice versa.

Hence, the overall max-margin regularized EG loss formulation $L_{\mathbf{MM}}(\mathbf{w}, x_i, y_i)$ as defined in (Collins et al., 2008) is shown in equation 3.4.

$$L_{\mathbf{MM}}(\mathbf{w}, x_i, y_i) = \arg\min_{\mathbf{w}} \sum_i \max_y \left[\mathrm{L}(x_i, y_i, y) - \mathrm{M}(x_i, y_i, y) \right] + \frac{C}{2} ||\mathbf{w}||^2 \qquad (3.4)$$

3.2 Candidate Generation based EG

The max-margin candidate enumeration EG algorithm has its exponentiated loss descent inversely proportional to the margin width of each discriminative candidate output y for a given x_i. An incorrect candidate with a small margin width to the truth output, needs to have its global discriminative score explicitly reduced so that it can be pushed away from the truth, so as to maximize its margin from the truth. As a result, it's local features are given a larger loss penalty (dual-loss) as compared to candidates that can be easily discriminated from the truth (candidates with larger margin values).

For the candidate generation EG algorithm, $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_N]$ is a dual-space loss parameter that needs to be optimized. A probability distribution $\boldsymbol{\alpha}_i$ is defined over all possible candidate generations for each input sequence x_i. Each $\boldsymbol{\alpha}_i$ represents a dual parametric vector probability distribution containing probability scores $\alpha_{i,y}$ assigned to each generated candidate $y \in \mathrm{GEN}(x_i)$, such that such that $\forall i : \sum_y \alpha_{i,y} = 1$. An update for a single instance $\alpha_{i,y}$ of the dual in the EG algorithm with a learning rate of η requires computation of an exponential hinge loss $(1 - M_{x_i,y})_+$ along with a Hamming loss $L_{i,y}$ over all local feature parts that can be identified in that candidate. The optimization function $Q(\boldsymbol{\alpha})$ is written as:

$$\begin{aligned} Q(\boldsymbol{\alpha}) &= Q(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_N) \\ &= -b^T \boldsymbol{\alpha} + \boldsymbol{\alpha}^T A \boldsymbol{\alpha} \end{aligned}$$

where A is a matrix over all $N.|Y| \times N.|Y|$ possible candidate generations and $|Y|$ represents the number of candidate generations for an input x_i. The matrix A can be indexed by any (x_i, y) pair, such that:

$$A_{(x_i,y),(x_j,z)} = \frac{1}{C} \Psi_{(x_i,y)} \Psi_{(x_j,z)}$$

and candidate potentials $\Psi_{(x_i,y)} = [\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)]$ and $\Psi_{(x_j,z)} = [\mathbf{\Phi}(x_j, z_j) - \mathbf{\Phi}(x_j, z)]$. The set of distributions over each $|\mathtt{Y}|$ can be denoted as a $|\mathtt{Y}|$ dimensional probability simplex $\triangle$, and for N training instances with distributions over $|\mathtt{Y}|$, the simplex can be represented as $\triangle^N$. The optimization function for EG in terms of the margin-maximization based dual formulation (Taskar et al., 2004b), (Bartlett et al., 2004) can be written as:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha} \in \triangle^N} Q(\boldsymbol{\alpha})$$

Algorithm 2 (Bartlett et al., 2004), (Song, 2008) shows a candidate enumeration based online EG algorithm, which requires enumeration of all possible candidate outputs for a particular train sequence x_i to compute their margin scores. The inputs to the algorithm include a learning rate η and a margin regularization constant C. The multiplicative updates for the dual-space variables are seen in line 13 of algorithm 2. Line 14 shows the weight updates performed in an online manner after the dual-space variables have been updated.

Algorithm 2 Online EG Training by enumeration of all generated candidates

Input: Training set (x_i, y_i) for $i = 1, \ldots N$
Input: learning rate η, margin parameter C, number of training iterations T
Primal Parameters: Global weight vector $\mathbf{w}$ that stores the local feature-part weights, $\forall r \in R$
Dual Parameters: Dual global vector $\boldsymbol{\alpha} \equiv [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \ldots \boldsymbol{\alpha}_N]$ over N training sequences. Each sequence-candidate pair (x_i, y) has a dual score $\alpha_{i,y}$, $\forall y \in \text{GEN}(x_i)$
Define: Global feature-vector $\boldsymbol{\Phi}(x_i, y)$, defined $\forall r \in R(x_i, y)$
Define: Candidate discriminative score: $S(x_i, y) = \mathbf{w} \cdot \boldsymbol{\Phi}(x_i, y)$
Initialize: $\boldsymbol{\alpha}_i$ as a vector with initial probabilities such that $\sum_y \alpha_{i,y} = 1$, $\forall y \in \text{GEN}(x_i)$
Initialize: $\mathbf{w} = [0, 0, \ldots, 0]$
for $t = 1, 2, \ldots, (T+1)$ **do**
  for $i = 1, 2, \ldots, N$ **do**
    if $y_i \neq \arg\max y \in GEN(x_i)$ **then**
      Calculate the margins: $M_{i,y}$, $\forall y \in GEN(x_i) : M_{i,y} = S(x_i, y_i) - S(x_i, y)$
      Update the duals: $\alpha_{i,y}^{t+1} = \frac{\alpha_{i,y}^{t} \exp\{\eta C(L_{i,y} - M_{i,y})\}}{\sum_{y'} \alpha_{i,y'}^{t} \exp\{\eta C(L_{i,y} - M_{i,y})\}}$
      Set the weight vector: $\mathbf{w} = \mathbf{w} + \sum_y \alpha_{i,y}^{'} [\boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, y)]$
    end if
  end for
end for
Output: $\frac{1}{C} \cdot \mathbf{w}$

We will explain the concept of candidate-enumeration based online EG algorithm using an example. Consider 2 word segmentation examples of the form (x_i, y_i) where x_i represents the input unsegmented text and y_i is the truth structured output representing points of segmentation. Suppose input x_1 is $\mathtt{w_0 w_1 w_1}$ with its corresponding truth segmentation $\mathtt{t_0 t_1 t_1}$, and input sequence x_2 is $\mathtt{w_1}$ with its corresponding truth segmentation $\mathtt{t_1}$, where tag $\mathtt{t_0}$ represents unsegmented text and tag $\mathtt{t_1}$ represents a text segmentation point.

The possible structured labels for $x_1 \equiv \mathtt{w_0 w_1 w_1}$ are:

$$y_1 \equiv \mathtt{t_0 t_1 t_1}, y_2 \equiv \mathtt{t_0 t_0 t_0}, y_3 \equiv \mathtt{t_0 t_0 t_1}, y_4 \equiv \mathtt{t_0 t_1 t_0}, y_5 \equiv \mathtt{t_1 t_0 t_0}, y_6 \equiv \mathtt{t_1 t_0 t_1}, y_7 \equiv \mathtt{t_1 t_1 t_0}, y_8 \equiv \mathtt{t_1 t_1 t_1}$$

The possible structured labels for $x_2 \equiv \mathtt{w_1}$ are:

$$y_1 \equiv \mathtt{t_1}, y_2 \equiv \mathtt{t_0}$$

where (x_1, y_1) and (x_2, y_1) represent the gold segmented outputs for sequences 1 and 2 respectively.

3.2.1 Hamming Loss

Before we look at the actual EG algorithm, we shall look into the concept of Hamming Loss over local feature parts, using the example sequence previously mentioned. For the input $x_1 \equiv \mathtt{w_0 w_1 w_1}$, let us consider a candidate segmentation $y \equiv \mathtt{t_1 t_0 t_1}$ and the corresponding truth segmentation is $y_1 \equiv \mathtt{t_0 t_1 t_1}$. The local structural parts at each index for the truth segmentation are shown in table 3.1. Tag $\mathtt{S_{-1}}$ is the beginning of sentence marker indicating a start state for the sequence, and tag $\mathtt{S_{+1}}$ is the end of sentence marker indicating a end state for the sequence. The end state features are not required for the candidate EG technique, however they are needed for the forward-backward computations of the part based EG technique as seen in section 3.3.

Input:		$\mathtt{w_0}$	$\mathtt{w_1}$	$\mathtt{w_1}$	
Truth segmentation:	$\mathtt{S_{-1}}$	$\mathtt{t_0}$	$\mathtt{t_1}$	$\mathtt{t_1}$	$\mathtt{S_{+1}}$
Truth segmentation parts		$\langle \mathtt{U} : \mathtt{w_0}, \mathtt{t_0} \rangle$ $\langle \mathtt{B} : \mathtt{S_{-1}}, \mathtt{t_0} \rangle$	$\langle \mathtt{U} : \mathtt{w_1}, \mathtt{t_1} \rangle$ $\langle \mathtt{B} : \mathtt{t_0}, \mathtt{t_1} \rangle$	$\langle \mathtt{U} : \mathtt{w_1}, \mathtt{t_1} \rangle$ $\langle \mathtt{B} : \mathtt{t_1}, \mathtt{t_1} \rangle$	$\langle \mathtt{U} : \mathtt{w_{S_{+1}}}, \mathtt{S_{+1}} \rangle$ $\langle \mathtt{B} : \mathtt{t_1}, \mathtt{S_{+1}} \rangle$
Candidate segmentation:	$\mathtt{S_{-1}}$	$\mathtt{t_1}$	$\mathtt{t_0}$	$\mathtt{t_1}$	$\mathtt{S_{+1}}$
Candidate segmentation parts		$\langle \mathtt{U} : \mathtt{w_0}, \mathtt{t_1} \rangle$ $\langle \mathtt{B} : \mathtt{S_{-1}}, \mathtt{t_1} \rangle$	$\langle \mathtt{U} : \mathtt{w_1}, \mathtt{t_0} \rangle$ $\langle \mathtt{B} : \mathtt{t_1}, \mathtt{t_0} \rangle$	$\langle \mathtt{U} : \mathtt{w_1}, \mathtt{t_1} \rangle$ $\langle \mathtt{B} : \mathtt{t_0}, \mathtt{t_1} \rangle$	$\langle \mathtt{U} : \mathtt{w_{S_{+1}}}, \mathtt{S_{+1}} \rangle$ $\langle \mathtt{B} : \mathtt{t_1}, \mathtt{S_{+1}} \rangle$

Table 3.1: Local Part illustration for Hamming Loss.

If we consider the unigram feature $\langle \mathtt{U} : \mathtt{w_0}, \mathtt{t_1} \rangle$ proposed by the candidate at its first index for the observed word $\mathtt{w_0}$, then we see that the proposed feature does not lie in the truth and hence we provide it with a loss value of 1. A similar local-loss of 1 is given to the bigram feature $\langle \mathtt{B} : \mathtt{t_1}, \mathtt{t_0} \rangle$ for being wrongly proposed in the candidate segmentation. If a non-truth part appears multiple times in the candidate segmentation, then a loss count is maintained. Thus for the above candidate segmentation, the hamming loss $\mathtt{L}$ is computed

as shown below.

$$L(x_i \equiv \mathtt{w_0 w_1 w_1}, y_i \equiv \mathtt{t_0 t_1 t_1}, y \equiv \mathtt{t_1 t_0 t_1}) = l_{\langle \mathtt{U:w_0,t_1} \rangle} + l_{\langle \mathtt{B:S_{-1},t_1} \rangle} + l_{\langle \mathtt{U:w_1,t_0} \rangle} + l_{\langle \mathtt{U:t_0,t_1} \rangle} + l_{\langle \mathtt{B:t_1,t_0} \rangle}$$
$$= 5$$

We now show an example to illustrate a working of the candidate enumeration based EG algorithm for a simple batch update on the HMM word segmentation example that was previously introduced. For the batch EG algorithm, given the dual parametric values $\alpha_{i,y}$, the primal global weight vector $\mathbf{w}$ is updated as shown in equation 3.5.

$$\mathbf{w} = \sum_{i,y} \alpha_{i,y} \cdot [\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)] \tag{3.5}$$

where (x_i, y_i) represents the truth structured output and (x_i, y) represents the candidate segmented output. The global feature vector $\mathbf{\Phi}$ defined over all local feature parts r is:

$$\mathbf{\Phi} \equiv [\phi_{\langle \mathtt{U:w_0,t_0} \rangle} \phi_{\langle \mathtt{U:w_0,t_1} \rangle} \phi_{\langle \mathtt{U:w_1,t_0} \rangle} \phi_{\langle \mathtt{U:w_1,t_1} \rangle} \phi_{\langle \mathtt{B:S_{-1},t_0} \rangle} \phi_{\langle \mathtt{B:S_{-1},t_1} \rangle} \phi_{\langle \mathtt{B:t_0,t_0} \rangle} \phi_{\langle \mathtt{B:t_0,t_1} \rangle} \phi_{\langle \mathtt{B:t_1,t_0} \rangle} \phi_{\langle \mathtt{B:t_1,t_1} \rangle}]$$

The global weight vector $\mathbf{w}$ defined over all local feature parts r is:

$$\mathbf{w} \equiv [w_{\langle \mathtt{U:w_0,t_0} \rangle} w_{\langle \mathtt{U:w_0,t_1} \rangle} w_{\langle \mathtt{U:w_1,t_0} \rangle} w_{\langle \mathtt{U:w_1,t_1} \rangle} w_{\langle \mathtt{B:S_{-1},t_0} \rangle} w_{\langle \mathtt{B:S_{-1},t_1} \rangle} w_{\langle \mathtt{B:t_0,t_0} \rangle} w_{\langle \mathtt{B:t_0,t_1} \rangle} w_{\langle \mathtt{B:t_1,t_0} \rangle} w_{\langle \mathtt{B:t_1,t_1} \rangle}]$$

We initialize the candidate duals with equi-probable values. The initializations for the EG algorithm's batch update simulation are:
$(\alpha_{1,1} \ldots \alpha_{1,8}) = 0.125$ and $(\alpha_{2,1}, \alpha_{2,2}) = 0.5$.
$\mathbf{w} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, $\eta = 1$ and $C = 1$.

Table 3.2 demonstrates a simulation of the initial weight vector update step for the EG algorithm.

The updated weight vector $\mathbf{w}$ obtained from Table 3.2, using the default initial values $\alpha_{i,y}$, is $\mathbf{w} = [0.5, -0.5, -1.5, 1.5, 0, 0, -0.5, 0.5, -0.5, 0.5]$. Table 3.3 demonstrates a simulation of the EG algorithm's loss computations and per-candidate dual variable updates, for all possible candidate generations of input sequences x_1 and x_2, after the initial weight vector update step. The Hamming loss $L_{i,y}$ is computed over all local losses $l_{i,r}$ for parts that are proposed in the candidate, but which do not lie in the truth output. Also, as previously seen in section 2.6, the per-candidate margin is computed as $M_{i,y} = \mathbf{w} \cdot (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y))$.

From table 3.3, it can be observed that the updated $\boldsymbol{\alpha}_i^{'}$ for each training sentence, obey the probabilistic distribution constraints as $\sum_{y=1\ldots8} \alpha_{1,y}^{'} = 1$ and $\sum_{y=1\ldots2} \alpha_{2,y}^{'} = 1$. Also,

Candidate Generations	$\mathbf{\Phi}(x_i, y)$	$\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)$	$\alpha_{i,y}$
(x_1, y_1)	$(1,0,0,2,1,0,0,1,0,1)$	$(0,0,0,0,0,0,0,0,0,0)$	0.125
(x_1, y_2)	$(1,0,2,0,1,0,2,0,0,0)$	$(0,0,-2,2,0,0,-2,1,0,1)$	0.125
(x_1, y_3)	$(1,0,1,1,1,0,1,1,0,0)$	$(0,0,-1,1,0,0,-1,0,0,1)$	0.125
(x_1, y_4)	$(1,0,1,1,1,0,0,1,1,0)$	$(0,0,-1,1,0,0,0,0,-1,1)$	0.125
(x_1, y_5)	$(0,1,2,0,0,1,1,0,1,0)$	$(1,-1,-2,2,1,-1,-1,1,-1,1)$	0.125
(x_1, y_6)	$(0,1,1,1,0,1,0,1,1,0)$	$(1,-1,-1,1,1,-1,0,0,-1,1)$	0.125
(x_1, y_7)	$(0,1,1,1,0,1,0,0,1,1)$	$(1,-1,-1,1,1,-1,0,1,-1,0)$	0.125
(x_1, y_8)	$(0,1,0,2,0,1,0,0,0,2)$	$(1,-1,0,0,1,-1,0,1,0,-1)$	0.125
(x_2, y_1)	$(0,0,0,1,0,1,0,0,0,0)$	$(0,0,0,0,0,0,0,0,0,0)$	0.5
(x_2, y_2)	$(0,0,1,0,1,0,0,0,0,0)$	$(0,0,-1,1,-1,1,0,0,0,0)$	0.5

Table 3.2: Table simulating EG calculations for initialization step of candidate generation method.

due to the $[0, 1]$ probability constraints, some $\alpha_{i,y}$ are driven down to near zero probabilistic values. For example, $\alpha_{1,2}$ and $\alpha_{1,5}$ in table 3.3 are driven down to near zero probabilities.

In table 3.3, for the generated non-truth candidate ($x_1 \equiv \mathtt{w_0 w_1 w_1}, y_8 \equiv \mathtt{t_1 t_1 t_1}$), the dual loss score $\alpha_{1,8}$ is the highest as compared to the dual loss scores for all other output candidates of x_1. This is because this candidate has the least margin distance (discriminative-width) from the truth, and hence it represents a best decoded candidate for the distribution $\boldsymbol{\alpha}_1$. The best decoded candidate needs to be heavily penalized so that its discriminative score can be significantly reduced from the truth's score (its margin can be maximized). The high $\alpha_{i,y}$ probabilistic dual-loss value assigned to the best decoded candidate will be used to enforce that penalty, by imparting a greater loss to the weights of the feature-parts that lie in this candidate, but not in the truth output. Equation 3.5 demonstrates the multiplicative weight updates for the feature-parts using the dual scores. Hence, the EG weight update step and its hinge loss property ensure that all non-truth candidates get penalized at each parse through the sentence, thus maximizing the discriminative margin. The amount of penalty is proportional to the dual loss values.

The updated weight vector $\mathbf{w}$ obtained from Table 3.3, after the $\alpha_{i,y}$ values were updated, is as shown below. The updated values in the global weight vector $\mathbf{w}$ show that, generally, feature-parts that were identified in the truth output had a higher updated weight than those which did not lie in the truth. An exception to that is the updated weight for the part

Candidate Generations	$L_{i,y}$	$M_{i,y}$	$\nabla_{i,y}$	$e^{(\eta \times \nabla_{i,y})}$	$\alpha_{i,y} e^{\eta \nabla_{i,y}}$	$\alpha'_{i,y}$
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_1 \equiv \mathtt{t_0 t_1 t_1})$	0	0	0	1	0.125	0.091
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_2 \equiv \mathtt{t_0 t_0 t_0})$	4	8	-4	0.018	0.002	0.001
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_3 \equiv \mathtt{t_0 t_0 t_1})$	3	4	-1	0.368	0.046	0.033
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_4 \equiv \mathtt{t_0 t_1 t_0})$	2	4	-2	0.135	0.016	0.013
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_5 \equiv \mathtt{t_1 t_0 t_0})$	6	9	-3	0.049	0.006	0.004
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_6 \equiv \mathtt{t_1 t_0 t_1})$	5	5	0	1	0.125	0.091
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_7 \equiv \mathtt{t_1 t_1 t_0})$	5	5	0	1	0.125	0.091
$(x_1 \equiv \mathtt{w_0 w_1 w_1}, y_8 \equiv \mathtt{t_1 t_1 t_1})$	3	1	2	7.39	0.923	0.674
$(x_2 \equiv \mathtt{w_1}, y_1 \equiv \mathtt{t_1})$	0	0	0	1	0.5	0.732
$(x_2 \equiv \mathtt{w_1}, y_2 \equiv \mathtt{t_0})$	2	3	-1	0.367	0.183	0.267

Table 3.3: Table simulating the dual $\alpha_{i,y}$ update for the candidate generation based EG algorithm.

$\langle \mathtt{B} : \mathtt{t_1}, \mathtt{t_1} \rangle$, which lies in the truth (x_1, y_1) but not in the Viterbi best candidate (x_1, y_8).

$$\begin{aligned}\mathbf{w} &\equiv [w_{\langle \mathtt{U:w_0,t_0} \rangle} w_{\langle \mathtt{U:w_0,t_1} \rangle} w_{\langle \mathtt{U:w_1,t_0} \rangle} w_{\langle \mathtt{U:w_1,t_1} \rangle} w_{\langle \mathtt{B:S_{-1},t_0} \rangle} w_{\langle \mathtt{B:S_{-1},t_1} \rangle} w_{\langle \mathtt{B:t_0,t_0} \rangle} w_{\langle \mathtt{B:t_0,t_1} \rangle} w_{\langle \mathtt{B:t_1,t_0} \rangle} w_{\langle \mathtt{B:t_1,t_1} \rangle}] \\ &= [0.863, -0.863, -0.495, 0.495, 0.593, -0.593, -0.038, 0.770, -0.2, -0.532]\end{aligned}$$

3.3 Part Enumeration based EG

We now consider a part-based EG technique, the motivation for which was provided earlier in section 2.3.1 and in various other sections that outlined the infeasibility of enumerating over a possible exponential number of generated candidates. The key to working with a part-based formulation is to identify the Viterbi best decoded candidates for each x_i at each iterative training phase. Instead of trying to maximize a margin for all output candidates $y = \text{GEN}(x_i)$, we identify candidates that lie closest to the truth (least margin width from the truth). The best decoded candidates can be identified by taking an $\arg\max y$ using Viterbi decoding over the part based EG loss formulation as shown in equations 3.3 and 3.4. Hence, non-truth candidates that were incorrectly decoded as being the best ones for a given x_i, have their discriminative scores explicitly reduced.

A generated candidate $y \in \text{GEN}(x_i)$ is considered to be made up of local parts r proposed at each sequence index, and any computations performed on the generated candidate can be modeled as a sum of computations on its local parts. The part based EG represents a practical way of using the EG algorithm in structured prediction tasks, thus avoiding

an inference over a possible exponential number of candidate possibilities. An important aspect of the part based EG algorithm is that it is modeled as a globally normalized Markov framework over all parts that can possibly get proposed for a training sentence x_i.

The computation of $\alpha_{i,y}$ as shown in equation 3.1 can be represented in terms of local structural parts as shown in equation 3.7

$$\alpha'_{i,y} = \frac{\exp\{\eta \nabla_{i,y}\}}{\sum_{y' \in GEN(x_i)} \exp\{\eta \nabla_{i,y'}\}} \text{ where } \nabla_{i,y} = C\left(L(x_i, y) - M(x_i, y)\right) \tag{3.6}$$

$$\sigma_{i,y}(\boldsymbol{\theta}) = \frac{\exp\{\eta \sum_{r \in R(x_i,y)} \theta_{i,r}\}}{\sum_{y' \in GEN(x_i)} \exp\{\eta \sum_{r' \in R(x_i,y')} \theta_{i,r'}\}} \tag{3.7}$$

$$= \frac{\exp\{\eta \sum_{r \in R(x_i,y)} \theta_{i,r}\}}{Z_i} \tag{3.8}$$

where $\theta_{i,r}$ represents a part based dual loss-score for a local part r that might be proposed by a candidate output $y \in \text{GEN}(x_i)$. Global dual vector $\boldsymbol{\theta}$ stores the part based loss $\theta_{i,r}$, $\forall i, \forall r \in R(x_i)$.

$$Z_i = \sum_{r \in R(x_i)} \sum_{y : r \in R(x_i,y)} \exp\{\theta_{i,r}\} \tag{3.9}$$

Equation 3.9 represents the total sum across all exponentiated dual loss-scores ($\exp\{\theta_{i,r}\}$), for all parts r that can possibly get proposed for the sequence x_i ($\forall r \in R(x_i)$). For each such part, the loss is marginalized across all sequence-candidate generations (x_i, y) in which part r lies ($\forall y : r \in R(x_i, y)$).

Algorithm 3 provides the pseudo code for the part based EG formulation, which was outlined in (Collins et al., 2008) as a means to overcome the exponential inference prevalent in the candidate generation based EG. We shall first introduce the algorithm and then go over the various aspects of the part based formulation that were extended from the candidate-enumeration based EG.

Algorithm 3 Online EG Training using part enumerations

Input: Training set (x_i, y_i) for $i = 1, 2 \dots N$

Input: learning rate η, margin parameter C, number of training iterations T

Primal Parameters: Global weight vector $\mathbf{w}$ that maintains updates for the local feature-part weights $\forall r \in R$

Parameters: Global dual vector $\boldsymbol{\theta}$ that stores the part based loss $\theta_{i,r}$ $\forall r \in R(x_i)$

Define: $\phi(x_i, r)$ as the number of times part r occurs in x_i

Define: Dual candidate score $\sigma_{i,y}(\boldsymbol{\theta})$ over all part-loss scores $\theta_{i,r}$ in $R(x_i, y)$

Define: Part r marginals: $\mu_{i,r}(\boldsymbol{\theta}) = \sum_{y:r \in R(x_i,y)} \exp\{\eta\theta_{i,r}\}$ as a sum over all candidates y in x_i that contain the part r

Initialize: $\mathbf{w} = [0, 0, 0 \dots 0]$

Initialize: $\boldsymbol{\theta}^1$ to a vector with random initial values for each $\theta_{i,r}$

for $t = 1, 2, \dots, (T + 1)$ **do**

  for $i = 1, 2, \dots, N$ **do**

    if $y_i \neq \arg\max y \in GEN(x_i)$ **then**

      for each $r \in R(x_i)$ **do**

        Set the dual loss scores: $\boldsymbol{\theta}_r^{t+1} \leftarrow \boldsymbol{\theta}_r^t + \eta(l_{i,r} + \mathbf{w}^t \cdot \phi_{i,r}/C)$

      end for

      Update $\mu_{i,r}(\boldsymbol{\theta}^{t+1})$ using Forward-Backward re-estimation $\forall r$ in $R(x_i)$

      for $r \in R(x_i, y_i)$ or $r \in R(x_i, \arg\max y)$ **do**

        if $t = 1$ **then**

          Set: $\mathbf{w} = \mathbf{w} + \boldsymbol{\Phi}(x_i, y_i) - \sum_r \mu_{i,r}(\boldsymbol{\theta}^1) \times \phi(x_i, r)$

        else if $t > 1$ **then**

$$\begin{aligned}\text{Set: } \mathbf{w} &= \sum_i \boldsymbol{\Phi}(x_i, y_i) - \sum_{i,y} \sigma_{i,y}(\boldsymbol{\theta}_i^{t+1}) \times \boldsymbol{\Phi}(x_i, y) \\ &= \mathbf{w} + \sum_y \sigma_{i,y}(\boldsymbol{\theta}^t) \times \boldsymbol{\Phi}(x_i, y) - \sum_y \sigma_{i,y}(\boldsymbol{\theta}^{t+1}) \times \boldsymbol{\Phi}(x_i, y) \\ &= \mathbf{w} + \sum_{r \in R(x_i)} \mu_{i,r}(\boldsymbol{\theta}^t) \times \phi(x_i, r) - \sum_{r \in R(x_i)} \mu_{i,r}(\boldsymbol{\theta}^{t+1}) \times \phi(x_i, r)\end{aligned}$$

        end if

      end for

    end if

  end for

end for

Output: $\frac{1}{C}.\mathbf{w}$

The dual loss updates $\theta_{i,r}$ for each part r proposed in x_i is based on whether r lies in the truth segmented candidate (($l_{i,r} = 0$)) or not (($l_{i,r} = 1$)). For a given training iteration t, the part based EG algorithm requires that a $\theta^t_{i,r}$ update for a given sentence x_i be propagated to all $\theta^t_{i',r}$ $\forall x_{i'} \neq x_i$. This step requires iterating over all training sentences which might possibly contain the part r. To avoid looping over all $\theta^t_{i',r}$ the algorithm maintains a global part-score vector $\boldsymbol{\theta}$ of the same dimension R as the global weight vector $\mathbf{w}$, where each indexed value in the global vector $\boldsymbol{\theta}$ represents a dual loss-score for each part across all training sentences. This formulation for the dual distribution significantly helps reduce the memory requirements, as maintaining an independent $\theta_{i,r}$ vector over all parts proposed in a given x_i would be very memory intensive. This update is seen in line 14 of algorithm 3. Thus, any part-indexed update for the dual loss-score need not be explicitly propagated to all training sentences that might possibly contain the part; only the global vector $\boldsymbol{\theta}$ gets updated.

For each iteration over a sentence x_i, the $\boldsymbol{\theta}$ updates are done for all $r \in R(x_i)$, where $R(x_i)$ represents a set of all possible parts that can be proposed for a given x_i (Collins et al., 2008):

$$\boldsymbol{\theta}^{t+1}_r = \begin{cases} \boldsymbol{\theta}^t_r + \eta \frac{\mathbf{w}(\boldsymbol{\theta}) * \phi_{i,r}}{C} & \text{if } r \in R_{truth} \\ \boldsymbol{\theta}^t_r + \eta \left(l_{i,r} + \frac{\mathbf{w}(\boldsymbol{\theta}) * \phi_{i,r}}{C} \right) & \text{if } r \notin R_{truth} \end{cases} \tag{3.10}$$

$l_{i,r} = 1$ represents a local-loss value for wrongly proposing the part r for a sentence x_i. The weight vector $\mathbf{w}$ is represented as $\mathbf{w}(\boldsymbol{\theta})$, indicating that its update is dependent on the update of the global vector $\boldsymbol{\theta}$. In other words, only after the online updates for the dual parameters are completed, are the primal weight parameters updated in an online fashion.

An important aspect of the EG algorithm can be seen from line 17 onwards of algorithm 3 where the online weight update procedure takes place. It is seen that the marginal $\mu_{i,r}$ values for a given iteration t need to be memorized for the update step in $t+1$. Unlike the $\theta_{i,r}$ values which can be maintained globally by $\boldsymbol{\theta}$, the $\mu^t_{i,r}$ values need to be explicitly memorized for usage in the next iteration over the training example x_i. This is because the marginal values $\mu^t_{i,r}$ are locally maintained by each sentence and do not get propagated across all sentences.

The $\mu_{i,r}$ values represent the marginals of local parts r across all $y = \text{GEN}(x_i)$. That is, they represent an accumulation of exponentiated dual loss scores across all candidate

generations that contain r. The marginals $\mu_{i,r}$ as defined in (Bartlett et al., 2004) are:

$$\mu_{i,r}^{t} = \sum_{y} \alpha_{i,y}(\boldsymbol{\theta}^{t}) I(x_i, y, r)$$

$$\text{where, part indicator function } I(x_i, y, r) = \begin{cases} 0 & \text{if } r \notin R(x_i, y) \\ 1 & \text{otherwise} \end{cases}$$

A part r might be proposed multiple times in a given sentence x_i, resulting in multiple loss-score $\theta_{i,r}$ updates for the part in that sentence. A marginalization $\mu_{i,r}$ over all such exponentiated loss-updates at different sentence indices, captures the total exponentiated loss-score update for that part in x_i. This marginalization can also be expressed as:

$$\mu_{i,r} = \sum_{y : r \in R(x_i, y)} \sigma_{i,y}(\boldsymbol{\theta})$$

where $\sigma_{i,y}(\boldsymbol{\theta})$ is a dual-score assigned to candidate y, which can be represented as a sum over scores of all local feature parts that can exist in the sequence-candidate (x_i, y) pair. The total summed exponentiated loss for r is finally normalized with Z_i to give us a probabilistic loss $\mu_{i,r}$ for a part r proposed across candidate generations of x_i. The estimation of the marginals $\mu_{i,r}$ requires a forward-backward re-estimation (Rabiner, 1989) of the per-part exponentiated loss score across all $y \in \text{GEN}(x_i)$ (at each sentence index $\imath$), for a given part $r \in R(x_i)$. $\mu_{i,r}$, along with $\phi_{i,r}$, the number of times the part was proposed in x_i, is then used in an online weight update procedure to update the weight vector $\mathbf{w}$. Line 17 onwards of algorithm 3 demonstrates the online weight update step. An important point to be noted for the weight update step is that weights for only those parts of x_i which lie in the truth y_i or which are proposed in the Viterbi best decoded sequence $\arg\max y$ get updated. This helps avoid weight updates for irrelevant parts.

For a given training iteration t and for a given training sentence x_i of length n, the forward-backward re-estimation for $\mu_{i,r}$ is based on the following recursive derivation at each sentence index:

$$
\begin{aligned}
\mu_{i,r} &= \sum_{y} \frac{\exp\left(\sum_{r \in R(x_i,y)} \theta_{i,r}\right)}{Z_i} I(x_i, y, r) \\
&= \sum_{y_1 \ldots y_n} \frac{\exp\left(\sum_{r \in R(x_i,y_1 \ldots y_n)} \theta_{i,r}\right)}{Z_i} I(x_i, y_1 \ldots y_n, r) \\
&= \sum_{y_1 \ldots y_{n-1}} \frac{\exp\left(\sum_{r \in R(x_i,y_1 \ldots y_{n-1})} \theta_{i,r}\right)}{Z_i} I(x_i, y_1 \ldots y_{n-1}, r) \\
&\quad + \sum_{y_n} \frac{\exp\left(\sum_{r \in R(x_i,y_n,Unigram)} \theta_{i,r}\right)}{Z_i} I(x_i, y_n, r) \\
&\quad + \sum_{y_{n-1}} \sum_{y_n} \frac{\exp\left(\sum_{r \in R(x_i,y_n,Bigram)} \theta_{i,r}\right)}{Z_i} I(x_i, y_n, r)
\end{aligned}
$$

where each $y_1, y_2, \ldots y_n$ represent one of 23 discrete phrasal tags (say $\mathtt{t_1}, \mathtt{t_2}, \ldots \mathtt{t_{23}}$) that can possibly be generated at each sentence index $\imath = [1..n]$, for the candidate y.

For the previously introduced sequence tagging HMM example, $(x_1, y_1) \equiv (\mathtt{w_0 w_1 w_1}, \mathtt{t_0 t_1 t_1})$, the forward and backward table computations are shown in table 3.4. During training, all forward-backward computations are performed independently for each sentence and hence for simplifying notation, we shall maintain our forward-backward notation sentence independent. Each entry in the forward table can be represented as $F(index, tag)$. For the simple HMM case, tag represents one of $\mathtt{t_0}$ or $\mathtt{t_1}$, indicating a segmentation or no segmentation at the current sequence index. For the phrasal chunking task, index represents the current word index within a sentence and tag represents one of $\mathtt{t_0} \ldots \mathtt{t_{23}}$ discrete phrasal chunks that can possibly be generated at the current index.

The forward recurrence relation is given by:

$$F(\imath, tag) = \sum_{prev-tag} F(\imath - 1, prev - tag) \exp\{\theta_{\langle U:feat,tag \rangle}\} * \exp\{\theta_{\langle B:prev-tag,tag \rangle}\}$$

The backward recurrence relation is given by:

$$B(\imath - 1, prev - tag) = \sum_{tag} * B(\imath, tag) * \exp\{\theta_{\langle U:feat,tag \rangle}\} * \exp\{\theta_{\langle B:prev-tag,tag \rangle}\}$$

For a given sequence index $\imath$, the forward table provides a probabilistic marginal across all candidate paths from $[0 \ldots \imath]$ that contain the part r. The backward table provides a probabilistic marginal across all candidate paths from $[\imath + 1 \ldots n]$ that contain the part r. Thus, combining the forward-backward inference at each index, and then summing over all

such per-index inferences, provides a marginal value μ_r for r across all candidate generations. The normalization for the marginals is done globally with Z_i. The forward-backward relation for the part marginals μ_r at each sequence index $\imath$ is shown below.

$$\mu_r(\imath, \langle \mathtt{U} : \mathtt{feat}, \mathtt{tag} \rangle) = \frac{F(\imath, tag) * B(\imath, tag)}{Z_i} \quad \text{for unigram feature parts}$$

$$\mu_r(\imath - 1, \langle \mathtt{B} : p\mathtt{rev} - \mathtt{tag}, \mathtt{tag} \rangle) = \frac{F(\imath - 1, prev - tag) * B(\imath, tag) * \exp\{\theta_{\langle B:prev-tag,tag \rangle}\} * \exp\{\theta_{\langle U:feat,tag \rangle}\}}{Z_i}$$

for bigram feature parts

We shall outline the part based EG methodology using a simple sequence-truth label example $(x_1, y_1) \equiv (\mathtt{w_0 w_1 w_1}, \mathtt{t_0 t_1 t_1})$, which was previously introduced in section 3.2. The initializations for the part based EG algorithm are:

$$\eta = 1, C = 1 \text{ and } \mathbf{w} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

$$\begin{aligned} \boldsymbol{\theta} &\equiv \left(\theta_{\langle \mathtt{U:w_0,t_0} \rangle}, \theta_{\langle \mathtt{U:w_0,t_1} \rangle}, \theta_{\langle \mathtt{U:w_1,t_0} \rangle}, \theta_{\langle \mathtt{U:w_1,t_1} \rangle}, \theta_{\langle \mathtt{B:S_{-1},t_0} \rangle}, \theta_{\langle \mathtt{B:S_{-1},t_1} \rangle}, \theta_{\langle \mathtt{B:t_0,t_0} \rangle}, \theta_{\langle \mathtt{B:t_1,t_0} \rangle}, \theta_{\langle \mathtt{B:t_0,t_1} \rangle}, \theta_{\langle \mathtt{B:t_1,t_1} \rangle}\right) \\ &\equiv (1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \end{aligned}$$

If we work in the probabilistic space for the marginal computations, then the forward table is initialized to $F(\imath = 0, \mathtt{S_{-1}}) = 1$, which represents the start state of the sequence. The initialization for the backward table is $B(\imath = \mathtt{END}, \mathtt{S_{+1}}) = 1$, which represents the end state of the sequence.

Practically, the forward-backward computations for EG are done in the logarithmic space due to the large loss penalties imposed by exponentiated multiplicative updates. For the HMM example, in the logarithmic space, the initialization for the forward table is $F(\imath = 0, \mathtt{S_{-1}}) = 0$, and the initialization for the backward table is $B(\imath = \mathtt{END}, \mathtt{S_{+1}}) = 0$.

We illustrate the forward table recurrence relation at each sequence index for the sequence-truth output example pair $(x_1, y_1) \equiv (\mathtt{w_0 w_1 w_1}, \mathtt{t_0 t_1 t_1})$. In our implementation we use a exponential of base 2. We use a notation $F(\imath, \mathtt{t}_{\imath-1} \rightarrow \mathtt{t}_\imath)$ to indicate the transition from a state at the previous index $(\imath - 1)$ to the current index $\imath$.
At $Index = 1$:

$$\begin{aligned} F(1, \mathtt{t_0}) &= F(1, \mathtt{S_{-1}} \rightarrow \mathtt{t_0}) \\ &= F(0, \mathtt{t_0}) * \exp\{\theta_{\langle \mathtt{U:w_0,t_0} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:S_{-1},t_0} \rangle}\} \end{aligned}$$

$$\begin{aligned} F(1, \mathtt{t_1}) &= F(1, \mathtt{S_{-1}} \rightarrow \mathtt{t_1}) \\ &= F(0, \mathtt{t_1}) * \exp\{\theta_{\langle \mathtt{U:w_0,t_1} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:S_{-1},t_1} \rangle}\} \end{aligned}$$

At $Index = 2$:

$$\begin{aligned} F(2, \mathtt{t_0}) &= F(2, \mathtt{t_0} \rightarrow \mathtt{t_0}) + F(2, \mathtt{t_1} \rightarrow \mathtt{t_0}) \\ &= F(1, t_0) * \exp\{\theta_{\langle \mathtt{U:w_1, t_0} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_0, t_0} \rangle}\} + \\ &\quad F(1, t_1) * \exp\{\theta_{\langle \mathtt{U:w_1, t_0} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_1, t_0} \rangle}\} \end{aligned}$$

$$\begin{aligned} F(2, \mathtt{t_1}) &= F(2, \mathtt{t_0} \rightarrow \mathtt{t_1}) + F(2, \mathtt{t_1} \rightarrow \mathtt{t_1}) \\ &= F(1, \mathtt{t_0}) * \exp\{\theta_{\langle \mathtt{U:w_1, t_1} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_0, t_1} \rangle}\} + \\ &\quad F(1, \mathtt{t_1}) * \exp\{\theta_{\langle \mathtt{U:w_1, t_1} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_1, t_1} \rangle}\} \end{aligned}$$

At $Index = 3$:

$$\begin{aligned} F(3, \mathtt{t_0}) &= F(3, \mathtt{t_0} \rightarrow \mathtt{t_0}) + F(3, \mathtt{t_0} \rightarrow \mathtt{t_0}) \\ &= F(2, \mathtt{t_0}) * \exp\{\theta_{\langle \mathtt{U:w_1, t_0} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_0, t_0} \rangle}\} + \\ &\quad F(2, \mathtt{t_1}) * \exp\{\theta_{\langle \mathtt{U:w_1, t_0} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_1, t_0} \rangle}\} \end{aligned}$$

$$\begin{aligned} F(3, \mathtt{t_1}) &= F(3, t_0 \rightarrow t_1) + F(3, t_0 \rightarrow t_1) \\ &= F(2, t_0) * \exp\{\theta_{\langle \mathtt{U:w_1, t_1} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_0, t_1} \rangle}\} + \\ &\quad F(2, t_1) * \exp\{\theta_{\langle \mathtt{U:w_1, t_1} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_1, t_1} \rangle}\} \end{aligned}$$

At $Index = 4$:

$$\begin{aligned} F(4, \mathtt{S_{+1}}) &= F(4, \mathtt{t_0} \rightarrow \mathtt{S_{+1}}) + F(4, \mathtt{t_1} \rightarrow \mathtt{S_{+1}}) \\ &= F(3, \mathtt{t_0}) * \exp\{\theta_{\langle \mathtt{U:w_{S_{+1}}, S_{+1}} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_0, S_{+1}} \rangle}\} + \\ &\quad F(3, \mathtt{t_1}) * \exp\{\theta_{\langle \mathtt{U:w_{S_{+1}}, S_{+1}} \rangle}\} * \exp\{\theta_{\langle \mathtt{B:t_1, S_{+1}} \rangle}\} \end{aligned}$$

For the HMM based example sequence as shown in table 3.4, the forward and backward table formations in probability space use an exponentiated value of 2 for the loss. For each $(index, tag)$ cell of the Forward table, we show how the forward table values are incrementally computed as summation values up to that sequence index. A summation over all sequence paths identifiable at a particular $(index, tag)$ gives us the value of the forward table cell at that particular index. The backward table is also formed incrementally, but in a reverse manner starting from the goal state's index and marginalizing the values in a backward manner. For practical implementations, especially when dealing with long sequences, the forward-backward and marginal computations are done in logarithmic space to avoid overflow errors arising due to propagation of large unnormalized marginal inferences.

	$\imath = 0$	$\mathtt{w}_0(\imath = 1)$	$\mathtt{w}_1(\imath = 2)$	$\mathtt{w}_1(\imath = 3)$	$\imath = \mathtt{END}$
$\mathtt{S}_{-1}$	$\mathtt{F}(0, \mathtt{S}_{-1}) = 1$ $\mathtt{B}(0, \mathtt{S}_{-1}) = 2048$				
Proposed Part Marginals	$\mu_{\langle \mathtt{B}:\mathtt{S}_{-1}, \mathtt{t}_0 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{S}_{-1}, \mathtt{t}_1 \rangle} = 1024$				
$\mathtt{t}_0$		$\mathtt{F}(1, \mathtt{t}_0) = 4$ $\mathtt{B}(1, \mathtt{t}_0) = 256$	$\mathtt{F}(2, \mathtt{t}_0) = 32$ $\mathtt{B}(2, \mathtt{t}_0) = 32$	$\mathtt{F}(3, \mathtt{t}_0) = 256$ $\mathtt{B}(3, \mathtt{t}_0) = 4$	
Proposed Part Marginals		$\mu_{\langle \mathtt{U}:\mathtt{w}_0, \mathtt{t}_0 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{t}_0, \mathtt{t}_0 \rangle} = 512$	$\mu_{\langle \mathtt{U}:\mathtt{w}_1, \mathtt{t}_0 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{t}_1, \mathtt{t}_0 \rangle} = 512$	$\mu_{\langle \mathtt{U}:\mathtt{w}_1, \mathtt{t}_0 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{t}_0, \mathtt{S}_{+1} \rangle} = 512$	
$\mathtt{t}_1$		$\mathtt{F}(1, \mathtt{t}_1) = 4$ $\mathtt{B}(1, \mathtt{t}_1) = 256$	$\mathtt{F}(2, \mathtt{t}_1) = 32$ $\mathtt{B}(2, \mathtt{t}_1) = 32$	$\mathtt{F}(3, \mathtt{t}_1) = 256$ $\mathtt{B}(3, \mathtt{t}_1) = 4$	
Proposed Part Marginals		$\mu_{\langle \mathtt{U}:\mathtt{w}_0, \mathtt{t}_1 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{t}_0, \mathtt{t}_1 \rangle} = 512$ $\mu_{\langle \mathtt{B}:\mathtt{t}_1, \mathtt{t}_1 \rangle} = 512$	$\mu_{\langle \mathtt{U}:\mathtt{w}_1, \mathtt{t}_1 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{t}_0, \mathtt{t}_1 \rangle} = 512$ $\mu_{\langle \mathtt{B}:\mathtt{t}_1, \mathtt{t}_1 \rangle} = 512$	$\mu_{\langle \mathtt{U}:\mathtt{w}_1, \mathtt{t}_1 \rangle} = 1024$ $\mu_{\langle \mathtt{B}:\mathtt{t}_1, \mathtt{S}_{+1} \rangle} = 512$	
$\mathtt{S}_{+1}$					$\mathtt{F}(\mathtt{END}, \mathtt{S}_{+1}) = 2048$ $\mathtt{B}(\mathtt{END}, \mathtt{S}_{+1}) = 1$
Proposed Part Marginals					$\mu_{\langle \mathtt{U}:\mathtt{w}_{\mathtt{S}_{+1}}, \mathtt{S}_{+1} \rangle} = 2048$
Truth Parts		$\langle \mathtt{U} : \mathtt{w}_0, \mathtt{t}_0 \rangle$ $\langle \mathtt{B} : \mathtt{S}_{-1}, \mathtt{t}_0 \rangle$	$\langle \mathtt{U} : \mathtt{w}_1, \mathtt{t}_1 \rangle$ $\langle \mathtt{B} : \mathtt{t}_0, \mathtt{t}_1 \rangle$	$\langle \mathtt{U} : \mathtt{w}_1, \mathtt{t}_1 \rangle$ $\langle \mathtt{B} : \mathtt{t}_1, \mathtt{t}_1 \rangle$	$\langle \mathtt{U} : \mathtt{w}_{\mathtt{S}_{+1}}, \mathtt{S}_{+1} \rangle$ $\langle \mathtt{B} : \mathtt{t}_1, \mathtt{S}_{+1} \rangle$

Table 3.4: Forward, Backward, and Marginal table formations for part based EG.

The global normalization factor for the part marginals computed in table 3.4 is:

$$\begin{aligned} Z &= \sum_{r} \sum_{index=\imath} \mu_{\langle \imath, r \rangle} \\ &= 18432 \end{aligned}$$

After computing the forward-backward values recursively as shown in table 3.4, the marginal computations μ_r for each part r, at each index $\imath$, are done as shown in table 3.5. The computation of values for the marginals are done in the backward direction, in a manner similar to the backward table. However, unlike the backward table, the marginal values are not recursively propagated, rather independently computed from forward and backward table values. The detailed per-index computations for the unnormalized marginal values at each sequence index and for each distinct

part, are shown in table 3.5.

Index=0	
Part r	μ_r
$\langle \mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_0\rangle$	$\mu_{\langle \imath=0,\mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_1\rangle} = F(0,\mathtt{S}_{-1}) * B(1,\mathtt{t}_0) * 2^{\theta_{\langle \mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_0,\mathtt{t}_0\rangle}}$
$\langle \mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_1\rangle$	$\mu_{\langle \imath=0,\mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_1\rangle} = F(0,\mathtt{S}_{-1}) * B(1,\mathtt{t}_1) * 2^{\theta_{\langle \mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_1\rangle}} * 2^{\theta_{\langle \mathtt{B}:\mathtt{S}_{-1},\mathtt{t}_1\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_0,\mathtt{t}_1\rangle}}$
Index=1	
Part r	μ_r
$\langle \mathtt{U}:\mathtt{w}_0,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=1,\mathtt{U}:\mathtt{w}_0,\mathtt{t}_0\rangle} = F(1,\mathtt{t}_0) * B(1,\mathtt{t}_0)$
$\langle \mathtt{U}:\mathtt{w}_0,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=1,\mathtt{U}:\mathtt{w}_0,\mathtt{t}_1\rangle} = F(1,\mathtt{t}_1) * B(1,\mathtt{t}_1)$
$\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=1,\mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle} = F(1,\mathtt{t}_0) * B(2,\mathtt{t}_0) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle}}$
$\langle \mathtt{B}:\mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=1,\mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle} = F(1,\mathtt{t}_1) * B(2,\mathtt{t}_0) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle}}$
$\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=1,\mathtt{B}:\mathtt{t}_0,\mathtt{t}_1\rangle} = F(1,\mathtt{t}_0) * B(2,\mathtt{t}_1) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_1\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle}}$
$\langle \mathtt{B}:\mathtt{B}:\mathtt{t}_1,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=1,\mathtt{B}:\mathtt{t}_1,\mathtt{t}_1\rangle} = F(1,\mathtt{t}_1) * B(2,\mathtt{t}_1) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_1\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle}}$
Index=2	
Part r	μ_r
$\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=2,\mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle} = F(2,\mathtt{t}_0) * B(2,\mathtt{t}_0)$
$\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=2,\mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle} = F(2,\mathtt{t}_1) * B(2,\mathtt{t}_1)$
$\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=2,\mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle} = F(2,\mathtt{t}_0) * B(3,\mathtt{t}_0) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle}}$
$\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=2,\mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle} = F(2,\mathtt{t}_1) * B(3,\mathtt{t}_0) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle}}$
$\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=2,\mathtt{B}:\mathtt{t}_0,\mathtt{t}_1\rangle} = F(2,\mathtt{t}_0) * B(3,\mathtt{t}_1) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_1\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle}}$
$\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=2,\mathtt{B}:\mathtt{t}_1,\mathtt{t}_1\rangle} = F(2,\mathtt{t}_1) * B(3,\mathtt{t}_1) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_1\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle}}$
Index=3	
Part r	μ_r
$\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle$	$\mu_{\langle \imath=3,\mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle} = F(3,\mathtt{t}_0) * B(3,\mathtt{t}_0)$
$\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle$	$\mu_{\langle \imath=3,\mathtt{U}:\mathtt{w}_1,\mathtt{t}_1\rangle} = F(3,\mathtt{t}_1) * B(3,\mathtt{t}_1)$
$\langle \mathtt{B}:\mathtt{t}_0,\mathtt{S}_{+1}\rangle$	$\mu_{\langle \imath=2,\mathtt{B}:\mathtt{t}_0,\mathtt{S}_{+1}\rangle} = F(3,\mathtt{t}_0) * B(\mathtt{END},\mathtt{S}_{+1}) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_0,\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle}}$
$\langle \mathtt{B}:\mathtt{t}_1,\mathtt{S}_{+1}\rangle$	$\mu_{\langle \imath=2,\mathtt{B}:\mathtt{t}_1,\mathtt{S}_{+1}\rangle} = F(3,\mathtt{t}_1) * B(\mathtt{END},\mathtt{S}_{+1}) * 2^{\theta_{\langle \mathtt{B}:\mathtt{t}_1,\mathtt{t}_0\rangle}} * 2^{\theta_{\langle \mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle}}$
Index=END	
Part r	μ_r
$\langle \mathtt{U}:\mathtt{w}_{\mathtt{S}_{+1}},\mathtt{S}_{+1}\rangle$	$\mu_{\langle \imath=3,\mathtt{U}:\mathtt{w}_1,\mathtt{t}_0\rangle} = F(\mathtt{END},\mathtt{S}_{+1}) * B(\mathtt{END},\mathtt{S}_{+1})$

Table 3.5: Computation of unnormalized marginals using forward-backward table values.

We show a simulation of the part based EG algorithm's weight update procedure in table 3.6. The unnormalized marginal values μ_r for each part are taken from table 3.4. The total marginal

values for each part are obtained by summing across marginal values computed at each index for that part. These unnormalized marginals are then divided by the global normalization factor Z for a given input sequence, to ensure that a globally normalized framework is maintained.

Part $\mathtt{r}$	$\mu_{\langle \mathtt{r} \rangle} = \sum_i \mu_{\langle i,\mathtt{r} \rangle}$	$\mu_{\langle \mathtt{r} \rangle} = \mu_{\langle \mathtt{r} \rangle}/Z_i$	$\phi_{\langle \mathtt{r} \rangle}(x_i)$	$\phi_{\langle \mathtt{r} \rangle}(x_i, y_i)$	$\mathbf{w}$
$\langle \mathtt{U} : \mathtt{w}_0, \mathtt{t}_0 \rangle$	1024	0.055	1	1	0.945
$\langle \mathtt{U} : \mathtt{w}_0, \mathtt{t}_1 \rangle$	1024	0.055	1	0	−0.055
$\langle \mathtt{U} : \mathtt{w}_1, \mathtt{t}_0 \rangle$	1024	0.055	2	0	−0.111
$\langle \mathtt{U} : \mathtt{w}_1, \mathtt{t}_1 \rangle$	1024	0.055	2	2	1.889
$\langle \mathtt{B} : \mathtt{S}_{-1}, \mathtt{t}_0 \rangle$	1024	0.055	1	1	0.945
$\langle \mathtt{B} : \mathtt{S}_{-1}, \mathtt{t}_1 \rangle$	1024	0.055	1	0	−0.055
$\langle \mathtt{B} : \mathtt{t}_0, \mathtt{t}_0 \rangle$	2048	0.111	2	0	−0.222
$\langle \mathtt{B} : \mathtt{t}_1, \mathtt{t}_0 \rangle$	2048	0.111	2	0	−0.222
$\langle \mathtt{B} : \mathtt{t}_0, \mathtt{t}_1 \rangle$	2048	0.111	2	1	0.778
$\langle \mathtt{B} : \mathtt{t}_1, \mathtt{t}_1 \rangle$	2048	0.111	2	1	0.778
$\langle \mathtt{U} : \mathtt{S}_{+1}, \mathtt{S}_{+1} \rangle$	1024	0.055	1	1	0.945
$\langle \mathtt{B} : \mathtt{t}_0, \mathtt{S}_{+1} \rangle$	1024	0.055	1	0	−0.055
$\langle \mathtt{B} : \mathtt{t}_1, \mathtt{S}_{+1} \rangle$	2048	0.111	1	1	0.889

Table 3.6: Simulation of part based EG weight update procedure where $\mathbf{w}_{\langle \mathtt{r} \rangle} = \mathbf{w}_{\langle \mathtt{r} \rangle} + \mathbf{w}_{\langle \mathtt{r} \rangle} * \phi_{\langle \mathtt{r} \rangle}(x_i, y_i) - \mu_{\langle \mathtt{r} \rangle} \phi_{\langle \mathtt{r} \rangle}(x_i)$.

The update for the weight vector $\mathbf{w}$, using the part marginals is shown in table 3.6. In table 3.6, it can be observed that weight values for parts that lie in the truth are greater than those which do not lie in the truth. It can also be observed that parts which are proposed frequently, but which do not lie in the truth (like $\langle \mathtt{B} : \mathtt{t}_1, \mathtt{t}_0 \rangle$), suffer a greater reduction in the weight values than others.

The online update mechanism for the global weight vector $\mathbf{w}$ and the global loss-score vector $\boldsymbol{\theta}$ is shown in equation 3.11. The parts r which are proposed for a training sentence x_i, index into the global loss-score vector as $\boldsymbol{\theta}_r$. The appropriate values of $\boldsymbol{\theta}$ get updated. $\boldsymbol{\theta}^{new}$ is then used in a forward-backward re-estimation step for the training sentence x_i, to form the part marginals for parts r proposed in x_i. The global weight vector $\mathbf{w}^{new}$ makes use of the updated marginal values $\mu_{i,r}(\boldsymbol{\theta}^{new})$ in its weight update step. The weight values are updated for only those $r \in R(x_i, y_i)$ or $r \in R(x_i, \arg\max y)$. The updated $\mathbf{w}^{new}$ values are then used to compute the new values of $\boldsymbol{\theta}$ for the next training sentence and the online process repeats itself, each time with parts r proposed through a possibly different training sentence $x_{i'}$.

$$\boldsymbol{\theta}_r^{new} = \boldsymbol{\theta}_r^{old} + \eta(l_{\mathtt{i,r}} + \frac{\mathbf{w}^{old}}{C}) \qquad \forall r \in R(x_i) \tag{3.11}$$

$$\mu_{i,r}(\boldsymbol{\theta}^{new}) = \sum_{y:r \in R(x_i,y)} \boldsymbol{\sigma}_{i,y}(\boldsymbol{\theta}^{new}) \tag{3.12}$$

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \sum_r \mu_{i,r}(\boldsymbol{\theta}^{new}) \times \phi(x_i, r) - \sum_r \mu_{i,r}(\boldsymbol{\theta}^{new}) \times \phi(x_i, r) \tag{3.13}$$

An important implementation detail noted in (Collins et al., 2008) is that for certain parts that get frequently updated due to the online update mechanism, an upper or lower bound can be placed on their dual-score value if they become too large or too small. This ensures that the frequent online updates for one part, do not degrade the entire learning mechanism.

3.4 Chapter summary

Chapter 3 provided us with a detailed explanation for the derivations and interpretations of the dual-space EG loss parameters, which form the core components of the online weight update procedure. The chapter discussed the Exponentiated Gradient algorithms for both the candidate enumeration based EG and the part enumeration based EG. It demonstrated how online updates in EG worked and provided a comprehensive simulation of the online weight update procedure of EG, including forward-backward inference learning to form globally normalized part based marginals.

In the next chapter, we shall see the experimental results by running a perceptron and part based EG algorithm with Viterbi decoding on a phrasal chunking dataset.

Chapter 4

Experimental Results

We present the overall accuracy, precision, recall and F-score measure, and per-phrase precision and recall accuracies for the perceptron and part based EG algorithms used with the Viterbi decoding scheme for the sequence learning task of finding non-recursive phrasal chunks in text. Precision is defined as the number of correctly segmented words divided by the total number of words in the segmentation result, where the correctness of the segmented words is determined by matching the segmentation with the gold standard test set. Recall is defined as the number of correctly segmented words divided by the total number of words in the gold standard test set. The independent training and test sentences for this task were taken from sections of the Wall Street Journal corpus (WSJ) with sections 15-18 as training data (211727 tokens) and section 20 as test data (47377 tokens). All accuracy evaluations were done on the independent test set.

As previously outlined in section 1.1, non-recursive chunking can be modeled as a tagging task. We revisit the example "The burglar robbed the apartment." previously introduced in section 1.1 to illustrate phrasal chunking as a tagging task. By using leftmost derivations of context free grammar(CFG) rules, we get "The burglar" as a bracketed noun phrase chunk (`NP`) in the sentence, and "robbed the apartment" as a bracketed verb phrase chunk (`VP`). Applying a per-word tagging structure to the above bracketed phrases, we get "The" as beginning noun phrase (`B-NP`) and "burglar" as an intermediate noun phrase(`I-NP`), "robbed" as beginning verb phrase (`B-VP`) and so on. Figure 4.1 illustrates how phrasal chunking can be represented as a tagging task.

The	burglar	robbed	the	apartment	.
[	NP]	[	VP]		[O]
B-NP	I-NP	B-VP	B-NP	I-NP	O

Figure 4.1: Phrasal chunking as a tagging task (Example taken from (Collins, 2003)).

The precision and recall accuracies of the test data are computed over all kinds bracketed phrasal chunks. The overall F-score measure (van Rijsbergen, 1979) on the test data is measured as a harmonic mean of the overall precision and recall accuracies as:

$$\text{F-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

We use the CRF++ feature generation framework (Kudo, 2005), which is an open source implementation of Conditional Random Fields (CRFs) for generating local structural features at each index of the sequence data. CRF++ consists of observed unigram features and bigram features, where unigram features are formed by combinations of the word under consideration and its 2 preceding and succeeding neighborhood words. Bigram features are used to extract the hidden contextual information from tags.

Figure 4.2 shows a representation of the CRF++ feature generation framework used for the chunking task. Features starting with 'U' are the observed unigram features generated at each index, and 'B' represents the fact that the given learning framework will make use of bigram features defined over phrasal tags that are defined at each sentence index and its preceding index. The distinct numbering framework adopted by the unigram tags (for example, U:00 or U:01) ensures that the generated features are not treated as a bag of words framework and that each feature's generation index has an associated significance. The unigram feature U00:%$x[-2, 0]$ indicates that the feature considers the word at the current index (indicated by the 0) and its preceding neighbor (indicated by the -1). The unigram feature U11:%$x[-1, 1]$ indicates that the single preceding and succeeding words of the indexed word are considered for generating this feature.

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]

U10:%x[-2,1]
U11:%x[-1,1]
U12:%x[0,1]
U13:%x[1,1]
U14:%x[2,1]
U15:%x[-2,1]/%x[-1,1]
U16:%x[-1,1]/%x[0,1]
U17:%x[0,1]/%x[1,1]
U18:%x[1,1]/%x[2,1]

U20:%x[-2,1]/%x[-1,1]/%x[0,1]
U21:%x[-1,1]/%x[0,1]/%x[1,1]
U22:%x[0,1]/%x[1,1]/%x[2,1]

# Bigram
B
```

Figure 4.2: CRF++ feature generation framework (Kudo, 2005)

.

The chunking accuracy was evaluated with a Perl script conlleval, provided as part of the CoNLL-2000 shared task. Conlleval can be used for measuring the performance of any system that has processed the CoNLL-2000 shared task data. It computes the precision and recall numbers for 11 different phrases that were derived from the WSJ corpus, except the O chunk which is used for tokens which are not part of any chunk. It must be noted that though the precision and recall values for the O chunk are not explicitly computed, they do affect the overall accuracy of the system. The `UDCP` phrasal chunk was not found in the data and hence do not get displayed in the final accuracy table for this data, as per the output obtained from conlleval.

4.0.1 Perceptron algorithm

The perceptron looks to minimize the number of mistakes made in each iteration and needs a fixed number of iterations to be determined beforehand. The perceptron algorithm does not require any input training parameters. Based on a few trial runs, we fix our perceptron training to 20 iterations. The best decoded sequence needed by the perceptron algorithm to perform the loss based additive updates is obtained using the Viterbi decoding scheme. The overall prediction accuracy (as a percentage), along with the overall precision, recall and F-score measures for the phrasal chunking task on the test set, is as shown in table 4.1. The perceptron algorithm might take more iterations to reach convergence than the EG algorithm, however in terms of absolute training times required, the perceptron's training period is significantly faster and unlike the EG algorithm, it does not have a significant memory overhead.

The phrase based precision and recall accuracies are shown in table 4.2. As expected, the phrase based precision and recall accuracies for commonly occurring parts of speech entities like `NP`,`VP` and `PP`, are seen to be better than other less frequent tags like `LST` and `CONJP`.

Overall Accuracy	Overall-Precision	Overall-Recall	Overall F-score
95.44	92.98	92.68	92.83

Table 4.1: Overall accuracy, precision, recall and F-score measures obtained using the perceptron algorithm for Phrasal Chunking.

Phrasal Chunk	Precision	Recall
ADJP	72.83	74.66
ADVP	79.47	80.02
CONJP	45.45	55.56
INTJ	100.00	50.00
LST	0.00	0.00
NP	93.33	92.69
PP	96.57	97.84
PRT	77.91	63.21
SBAR	87.93	80.37
VP	93.70	93.62

Table 4.2: Phrase based precision and recall accuracies using the perceptron algorithm.

4.0.2 Exponentiated Gradient algorithm

The part based Exponentiated Gradient learning algorithm is used with the Viterbi decoding scheme train the global weight vector $\mathbf{w}$. The EG algorithm requires an initial parameter setting for the learning rate η and a margin regularization constant C. To avoid overflow errors, the forward, backward and marginal computations for the EG algorithm are done in logarithmic space. An exponential value of 2 was used to update the loss.

The EG algorithm has a large parametric space. It includes the learning rate η for tuning the online loss-descent rate and the margin regularization constant C for controlling the strength of an update to affect the decision margin. The set of optimal parameters for a task may vary across different datasets and they may need to be empirically decided through a few trial runs. A good evaluation on the EG parameter space was outlined in (Collins et al., 2008) for the structured prediction task of dependency parsing.

To understand the effect of the parameters on the EG learning, we need to revisit equation 3.10. From this equation we gather that a larger η will impose a larger penalty on the dual-loss variable $\boldsymbol{\theta}$, thus resulting in a faster descent of the loss gradient. A very fast descent however would not give the algorithm sufficient freedom to effectively learn the weight parameters.

Our strategy in choosing the input parameters, as outlined in (Collins et al., 2008), was to start with parameters that are more stringent (like a larger value for the regularizer C) so as to ensure a slower initial descent of the loss. This ensures that the algorithm starts its learning with measured descent steps.

At each succeeding iteration, we allow for the regularizer C to be reduced, thus increasing the overall descent rate for the loss. For the EG implementation for which we have quoted our results, we select the margin elasticity constant C to have a initial value of 10, and we exponentially damp the C values at the end of each epoch as $C_t = 1 + C_0 * 0.7^t$ (Collins et al., 2008), where C_0 is the 0^{th} epoch's C value and t is current iteration. We found that this strategy for C works better than fixing the value of C across all training iterations. For each epoch's online update process, based on the parametric space evaluation done in (Collins et al., 2008),and based on our evaluations for this dataset, we fix η to be 0.01, with the η being damped towards zero at the rate of $\frac{\eta}{1+\frac{k}{n}}$. At the start of each epoch η is restored to a value of 0.01. (Collins et al., 2008) also noted that a randomized framework for the online EG algorithm gave better accuracies than a deterministic parse through the dataset. We use this idea in our implementation, where we randomize the train data before the start of each epoch.

Another aspect of the EG algorithm that needs to be considered is its slow run-times. The forward, backward and marginal table computations over all local parts $r \in R(x_i)$ result in slow run times for the EG algorithm. Hence, running the EG algorithm on the entire training dataset was found to be unsuitable for training over a large number of epochs. To improve upon the EG algorithm's run-time efficiency, we use a distributed training mechanism for the algorithm, where we

splice up the training data into non-overlapping contiguous chunks, and we run the iterative training phase on each of those chunks. We form the final weight vector by combining learnt weight vectors from all the distributed chunk data, and then averaging out the globally combined weight vector. For our EG implementation, we divided the train data into 6 sub-files, each consisting of around 1500 sentences.

The overall accuracy, precision, recall and F-score for the part based EG implementation, is shown in table 4.3. The per-phrase EG accuracy is shown in table 4.4.

Overall Accuracy	Overall-Precision	Overall-Recall	Overall F-score
91.31	85.77	84.17	84.96

Table 4.3: Overall accuracy, precision, recall and F-score measures obtained using the part based online EG algorithm for Phrasal Chunking.

Phrasal Chunk	Precision	Recall
ADJP	97.50	8.90
ADVP	88.02	46.65
CONJP	0.00	0.00
INTJ	0.00	0.00
LST	0.00	0.00
NP	83.63	87.05
PP	86.33	96.11
PRT	100.00	10.38
SBAR	95.70	20.93
VP	90.73	87.44

Table 4.4: Phrase based precision and recall accuracies using EG algorithm.

Figure 4.3 shows the loss gradient descent for the online EG algorithm. The loss function that we plot is the averaged-loss per epoch across all sub-files. The first 13 epochs are needed by the EG algorithm to stabilize itself after its rapid initial descent. After that the EG loss gradually descends towards convergence. Due to to the significant per-epoch run times of EG, we could not run the algorithm to convergence on the sub-files. [1]

[1]The average loss plotted is not a smooth function that is strictly reducing the loss because of the parallel training resulting in an averaged weight vector.

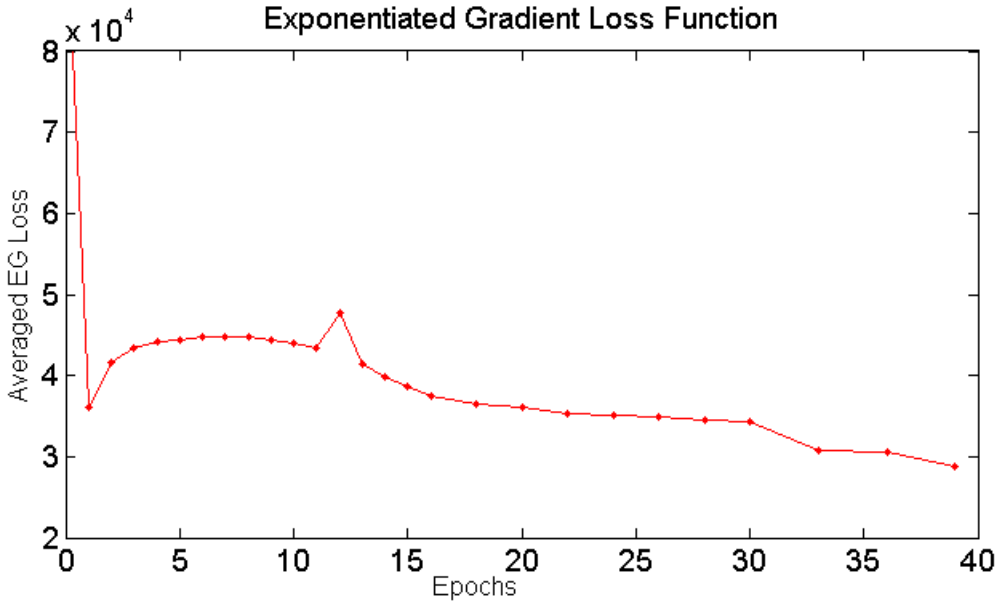

Figure 4.3: EG Loss function.

Chapter 5

Conclusion

The part based EG algorithm provides a computationally feasible way to perform online margin-maximization learning for Natural Language structured prediction tasks such as phrasal chunking. However, the EG algorithm when used in a max-margin structured prediction setting does possess a high degree of complexity with regards to complex tabular computations, arithmetic overflows, feature-space value degradations etc.. Our implementation of the EG algorithm tries to minimize the complexity inherently associated with this algorithm, and we have shown how this algorithm can be used in a practical setting for a Natural Language based tagging task. Our EG algorithm has achieved fair accuracy on the phrasal chunking task, which we hope to further improve by greater exploration of the EG parameter space and by other optimization techniques where applicable.

5.1 Future Work

Our main aim would be to further improve upon the existing accuracy of the part based EG algorithm for the phrasal chunking task, and to find ways to minimize the implementation complexity associated with this algorithm, wherever possible. Also, we hope to find more applications for the EG algorithm across different computing spectra, especially in areas where structured prediction based sequence learning might be useful.

Bibliography

[Abney1991] Steven Abney. 1991. *Parsing By Chunks in "Principle-Based Parsing by Robert Berwick, Steven Abney and Carol Tenny"*. Kluwer Academic Publishers.

[Bartlett et al.2004] Peter Bartlett, Michael Collins, Ben Taskar, and David McAllester. 2004. Exponentiated gradient algorithms for large-margin structured classification. In *Proceedings of NIPS 2004*.

[Berger et al.1996] Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Proceedings of Computational Linguistics, (22-1), March 1996*.

[Collins et al.2008] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. In *Journal of Machine Learning Research*.

[Collins2002] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP)*.

[Collins2003] Michael Collins. 2003. Tutorial on machine learning methods in natural language processing. In *Computational Learning Theory 2003*.

[Globerson et al.2007] Amir Globerson, Terry Koo, Xavier Carreras, and Michael Collins. 2007. Exponentiated gradient algorithms for log-linear structured prediction. In *Proceedings of ICML 2007*.

[Kivinen and Warmuth1997] J. Kivinen and M. Warmuth. 1997. Exponentiated gradient versus gradient descent for linear predictors. In *Information and Computation*, pages 132(1):1–63.

[Kudo2005] Taku Kudo. 2005. Crf++: Yet another crf toolkit. http://crfpp.sourceforge.net/.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Max-margin parsing. In *Proceedings of the Eighteenth International Conference on Machine Learning,San Francisco,CA, 2001*, pages 282–289. MIT press.

[McDonald et al.2010] Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, June 2010*, pages 456–464, Los Angeles, California.

[Rabiner1989] Lawrence Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 77(2):257–286.

[Ratnaparkhi1996] Adwait Ratnaparkhi. 1996. A maximum entropy model for parts of speech tagging. In *In proceedings of Association for Computational Linguistics*, New Brunswick,New Jersey.

[Rosenblatt1958] Frank Rosenblatt. 1958. The perception: A probabilistic model for information storage and organization in the brain. pages 65(6):386–408.

[Rosset et al.2004] Saharon Rosset, Ji Zhu, and Trevor Hastie. 2004. Margin maximizing loss functions. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2003), volume 16*, Cambridge, MA. MIT Press.

[Sang and Buchholz2000] Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task chunking. In *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000.*

[Skut and Brants1998] Wojciech Skut and Thorsten Brants. 1998. Chunk tagger: Statistical recognition of noun phrases.

[Song and Sarkar2009] Dong Song and Anoop Sarkar. 2009. Training global linear models for chinese word segmentation. In *In Proceedings of the 22nd Canadian Conference on Artificial Intelligence, Canadian AI 2009. May 25-27, 2009*, Kelowna, BC.

[Song2008] Dong Song. 2008. Experimental comparison of discriminative learning approaches for chinese word segmentation. In *M.Sc.Thesis.*

[Taskar et al.2004a] B. Taskar, C. Guestrin, and D. Koller. 2004a. Max margin markov networks. In and Schölkopf Thrun, Saul, editor, *Advances in Neural Information Processing Systems 16*. MIT Press.

[Taskar et al.2004b] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004b. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP).*

[van Rijsbergen1979] C.J. van Rijsbergen. 1979. *Information Retrieval.* Buttersworth.