A Family of Latent Variable Convex Relaxations for IBM Model 2

Andrei Arsene Simion

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2015

ABSTRACT

A Family of Latent Variable Convex Relaxations for IBM Model 2

Andrei Arsene Simion

The IBM translation models of (Brown et al., 1993) [8] were the first Statistical Machine Translation (SMT) systems; their primary use in the current SMT pipeline is to seed more sophisticated models which need alignment tableaus to start their optimization procedure. Although there are several IBM Models, only IBM Model 1 can be formulated as a convex optimization problem. Other IBM Models have non-concave objective functions with multiple local optima, and solving to optimality the non-convex problems that arise on account of these complex objectives is typically a computationally intractable task. This thesis focuses on the formulation and analysis of several new convex alignment models in SMT.

We now give a high level summary of this thesis.

In Chapter 1 we first give a general review of SMT and present some examples. In the second half of Chapter 1 we give a very specific outline of the contributions of this thesis. In Chapter 2 we focus on the relevant definitions which are ubiquitous in the paper and provide a literature review. The alignment and SMT summary on the IBM Models that we present is developed further in several sources, for example, in [8; 24; 28; 43].

Chapter 3 of this work details research found in [33] and [34]. Specifically, we describe the first convex relaxation of IBM Model 2, design an algorithm for its optimization, and conduct experiments showing that the model's performance is on par with that of the non-convex IBM Model 2. Moreover, we also derive a specific decoding rule for the new convex model and highlight some applications which given the new model favorable F-Measure performance.

In Chapter 4 we focus on work presented in [35] and [36]. In [35] we introduced a generalization of the previous results [33]. These new results not only subsume the previous work but also provide some new favorable alternatives. Specifically, the main of this chapter looks at a new relaxation

of IBM Model 2 based on the geometric mean. This new relaxation can be optimized via an EM algorithm that does not require the tuning of a learning rate. As before, the new relaxation performs on par with IBM Model 2 even though it is not the tightest relaxation in terms of objective value (in this section we also also show that the model in [33] is actually the tightest relaxation for IBM Model 2). Furthermore, the mechanism we introduce allows one to create a plethora of relaxations which could be studied and optimized by the EG algorithm detailed in [33]. Lastly, we also look at an application of the technique to IBM Model 1 [36]. As was shown in [41], IBM Model 1 is not a strictly convex optimization problem and because of this there is some alignment variance within its optimal solution set. We present a variant of IBM Model 1 that is strictly convex without the need to append an l_2 loss. The benefit of this last model is that there is no penalty cost to choose or learning rate to carefully set: the particular reparameterization we study allows for an EM algorithm that is as easy to implement as the original EM algorithm of IBM Model 1. On some level, this work on IBM Model 1 is an attempt to formalize (via another path) some of the heuristic improvements studied in the notable work of (Moore, 2004) [27] over ten years ago.

The final technical Chapter 5 of this thesis outlines an attempt [37] to find a convex alternative to the hidden Markov model (HMM), typically knows as IBM Model 2.5 [43]. Unlike IBM Model 2 which assumes the latent alignments variables are independent, the HMM allows alignments to depend on each other via “jump" parameters. The HMM can be optimized by an EM algorithm and is a very powerful, but unfortunately it still needs to be seeded carefully as the model is not convex. To attack this problem, we detail a powerful new variant of IBM Model 2 which combines some of the structure of HMM and then derive a relaxation of this surrogate. Although the new convex model does not beat the HMM, experiments shows that it performs well against other baselines: it still substantially improves (over 30%) upon IBM Model 2's performance, does better in practice than the popular new FastAlign [16] IBM Model 2, and is close in performance to the HMM.

Table of Contents

List of Figures

Acknowledgments

I first want to express my deepest gratitude to my adviser Michael Collins. I stumbled upon Mike's research in Optimization and NLP after passing the first leg of graduate study in the IEOR department and I am so glad that he made room for me in his popular research group. I thank Mike so much for pushing me and helping me constantly, especially at the beginning when I was trying hard to get that first publication and make the transition from good student to nascent researcher.

Secondly, I want to thank my advisor in the IEOR department, Cliff Stein. Although Mike was the main influence to my research development these last few years, Cliff was always insightful at meetings and offered much support and encouragement. My unorthodox path towards research in NLP was also supported by Cliff, so I was very fortunate.

It goes without saying that I would not be where I am now where it not for my parents and their guidance and encouragement. A consummate math aficionado, my father Iftimie Simion provided me the opportunity to see the beauty in math and helped me in everything that he could. My mother Felicia Simion has always been there for me and I could never have completed this work without her. Additionally, I also wanted to thank Neli, Dan, Radu and Anca: your encouragement was more than I could have hoped for.

I would like to thank all the staff and faculty members of the IEOR department. In particular, I want to thank Krupa for helping me book rooms during my busy job search and Jaya for finance and reimbursement related work. I also wanted to thank Vineet for advising me during my second year. I should also thank some of the department's Professors for teaching me some of the best courses on Optimization: Garud, Dan, and Ciamac Moallemi .

I met and learned a lot from my office mates at Columbia. I wanted to thank Shyam, JB, Peter, Carlos, Karl, Sasha, Yin-Wen and Avner especially as they've all influenced my life both professionally and socially in the best of ways. Outside of Columbia, I am all gratitude for the help and support of Kamaldeep, Dimitry, Raj, Ken, Stephen, David, Yuriy and Tommy.

Lastly, of course, I would like to thank my girlfriend Isamar. I met Isa when I started here at Columbia and I cannot tell her how lucky I was to have had her next to me everyday through this journey: Real and Fun and almost always (because of you!) Real-Fun!

To my mother Felicia Coralia Simion and my father Iftimie Arsenel Simion

Chapter 1

Introduction

1.1 NLP and Statistical Machine Translation

1.1.1 Introduction to Statistical Machine Translation (SMT)

The primary objective of Statistical Machine Translation (SMT) is to automate the translation of one natural language to another. Chiefly, SMT aims to translate the written form of language. Modern applications of SMT can also be used in conjunction with speech recognition and text-to-speech synthesis to translate spoken language on the fly (one popular application is Apple's Siri system[1]). Combining aspects of Optimization, Machine Learning, and Statistics, SMT is an important part of Natural Language Processing (NLP) research. As quality translation has grown in importance, in the last ten years SMT research has become very popular.

There are many applications of SMT and the field has both a social and commercial impact. For example, translation is very important in the European Union (E.U.) since it is a highly multilingual community and so documents and proceedings need to be translated. Translators are also used extensively in the United Nations (U.N.) and the Canadian Parliament. Machine translation can be used between any pair of languages, but it is a challenging and robust problem which has both general methods and pair specific ones. For example, for some language pairs such as English to Arabic and Chinese to English, the massive word reordering, lack of training data, and/or general differences in language structure require methods that are necessarily pair dependent. On the other

[1]https://www.apple.com/ios/siri/

hand, Latin languages are similar and more global techniques can be used. There are many generic SMT technologies: Google Translate[2] supports 90 different languages which it can automatically translate between on the fly. Automated SMT systems such as Google Translate are crucial to the dissemination of knowledge as they provide multilingual access to the vast amount of information available on the internet and other cross-language information sources.

1.1.2 Problem Specifics in SMT

At a high level, the fundamental aim of SMT is to take a fragment in one written language and translate it into another written language. A fragment can be anything from a single transcribed utterance to a book, document, newspaper or web page. We translate at the sentence level, viewing an input sentence as a sequence of words f and transforming it via the use of statistical models into a sequence of words in the output language e that represents an accurate translation of the original sentence. For most cases, there is no unique solution, but we aim to produce a sentence that is fluent and grammatically correct in the output language that has the same meaning as the original sentence. We build probabilistic models and choose a translation $\hat{e}$ that maximizes $p(e|f)$, the probability of output sentence e given input sentence f.

Although interest in building automated SMT systems was present since the Second World War [9], arguably the most influential work was carried out by a team of researchers at IBM (Brown et al., 1993) [8]. The IBM team were the first to formulate a source-channel model where it is assumed for modeling purposes that f is the result of passing e through the noisy channel defined by $p(f|e)$, and we wish to recover the original sentence e from the observed sentence f. Using Bayes' rule, we can rewrite our goal as the finding of

$$\hat{e} = \arg\max_{e \in \mathbf{E}} p(e|f) = \arg\max_{e \in \mathbf{E}} p(e)p(f|e) \ ,$$

where $\mathbf{E}$ is the set of all English sentences. In the above, we denote by $p(e)$ the language model and by $p(f|e)$ the translation model. In the development of SMT systems, sometimes researchers focus on one of $p(e)$ or $p(f|e)$ as these problems are usually very different to model. We note that the above process is known as decoding and occurs *after* the model parameters are estimated. Because of the complexity of some SMT systems, not only is training sometimes difficult, but decoding as

[2]http://translate.google.com

well. A detailed survey of some SMT systems and the specifics involved can be found in several sources such as Koehn (2010) [24] and Collins (2015) [12].

Generally speaking, the training of SMT models is very much like the training of any other Machine Learning application. Specifically, we train the translation model on data and using this data we estimate parameters via an optimization routine. For SMT, the data is usually corpora of parallel text, i.e. sentences that have same meaning in each language, which are obtained from a number of sources of manually translated documents. In choosing the specific corpus, we have a number of options. For example, in much of this work we use Canadian parliamentary proceedings because of its multilingual nature, but similar options such as corpa from the European Union or United Nations abound. Moreover, news articles that have been translated into other languages can be used. Bilingual data can also be gathered from web pages and fed to SMT systems. As is common in most Machine Learning problems, the more training data available for a given language pair, the better the quality of translation will be. State of the art SMT system design not only involves the use of clever modeling and cutting edge code optimization, but also a coverage of a wide variety of domains is essential so that a very robust model may be evaluated. Finally, in the construction of $p(e)$ above, a large amount of monolingual data in the output language is needed to train the language model. Although modeling $p(f|e)$ well is important, a state of the art language model is just as necessary for the the translation system to produce correct sentences in the output language.

1.1.3 Alignment Models in SMT

The usual approach to building a statistical machine translation system is to first build a model of alignment between the input and output languages. We use this to determine translational correspondences between the words and phrases of a sentence in one language with the words and phrases of a sentence in another language. For almost all the models we consider in this thesis, we can view alignment as a hidden variable and model training as a way to estimate parameters using this alignment. After training is complete and the parameter are estimated, we then use the models to output the best set of alignment links for each sentence in the training corpus. Since alignment is best understood visually at first, we present a possible alignment between the two sentences

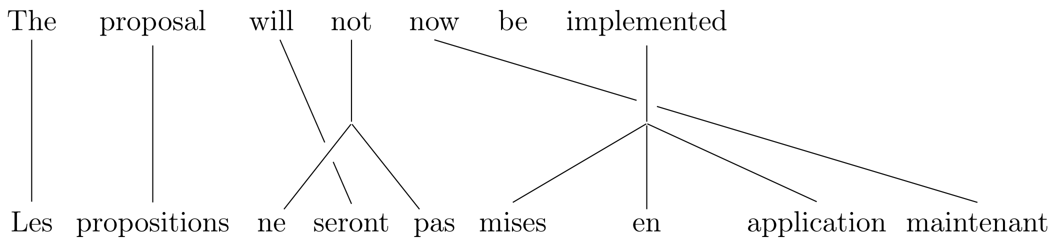
$$e = \text{The proposal will not now be implemented}$$

and

$$f = \text{Les propositions ne seront pas mises en application maintenant}$$

in Fig. 1.1. We will detail the mathematics behind this picture in Chapter 2.

Figure 1.1: An example alignment between two sentences. The mathematics behind this alignment varies from model to model with some examples given in Chapter 2.



Alignment models are currently used by almost all translation systems. The they are used by phrase-based systems for extracting phrase pairs from training data and building tables of possible translations of a phrase, which are then used for generation and scoring of hypotheses. As an example [24], we show a possible alignment and phrase extraction table in Fig. 1.2 between the two sentences

$$e = \text{Mary did not slap the green witch}$$

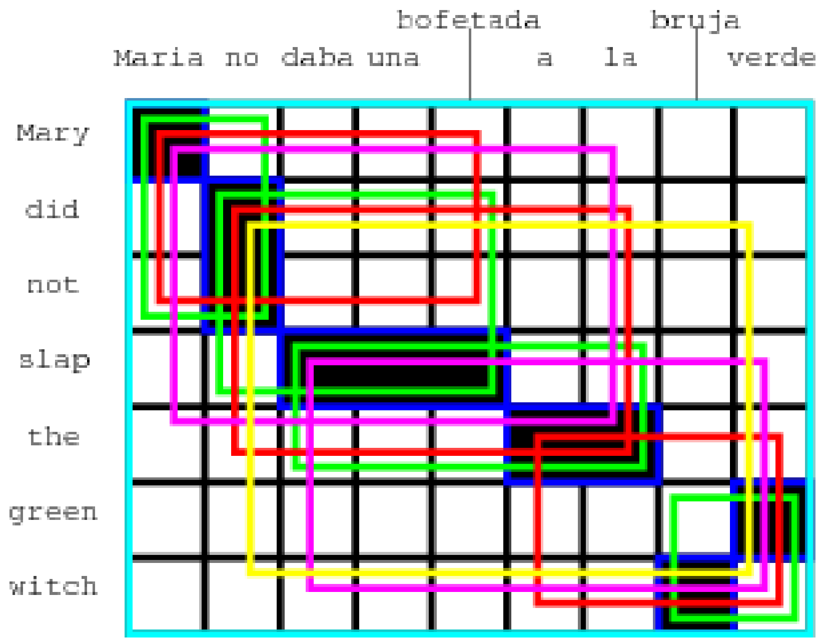
and

$$s = \text{Maria no daba una befetada a la bruja verde .}$$

In Fig. 1.2, the blackened squares refer to alignments and the boxes refer to consistent phrases extracted from the alignments (for more background material, see [24]); among the extracted phrases, some of these include: (Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not) and (Maria no daba una bofetada a la bruja verde, Mary did not slap the green witch).

Syntax-based and hierarchical phrase-based systems also initialize their translation models from word-aligned parallel text. Intuitively, more accurate alignments between the parallel text should lead to better phrase and syntax based initialization and this should ultimately lead to better translation systems being built. Therefore, the development and training of alignment models is very important to the quality of translation. This thesis focuses on the development of new convex alignment models for machine translation, and develops algorithms used for training them. In

Figure 1.2: An alignment and phrase extraction table for two sentences. Consistent alignments are circled in boxes.



particular, most alignment models used in the literature currently are non convex and cannot be solved to optimality, so heuristics are heavily used. Doing away with initialization heuristics is our main goal in introducing convexity into the alignment family of models. Indirectly, convexity adds theoretical justification by allowing for guaranteed model optimization.

1.2 Convex Alignment Models

The IBM models were the first generation of SMT systems. More recently, they play a central role in deriving the alignments used within many modern SMT approaches, for example phrase-based translation models [24] and syntax-based translation systems (e.g., [11; 14]). As we saw, since the original IBM paper, there has been a large amount of research exploring the original IBM models and modern variants (e.g., [27; 25; 41; 31; 43]).

Nevertheless, excluding IBM Model 1, the IBM translation models, and practically all variants proposed in the literature, have relied on the optimization of likelihood functions or similar functions that are non-concave. Unfortunately, non-concave objective functions have multiple local optima, and finding a global optimum of a non-concave function is typically a computationally intractable problem. To optimize the IBM Models, typically, an EM algorithm is used, which often runs in a reasonable amount of time, but with no guarantees of finding a global optima (or for that matter, even a near-optimal solution).

The focus of this work is on unsupervised learning of alignment models using convex optimization. Framing the unsupervised learning of alignment models as a convex optimization problem, with guaranteed convergence to a global optimum, has several clear advantages. First, the method is easier to analyze, as the objective function is being truly maximized. Second, there is no need for initialization heuristics with the approach, given that the method will always converge to a global optimum. Lastly, it may be easier to extend the approach to more complex models: it is possible that problems with local optima are a severe impediment to the development of more complex models. Finally, we expect that our convexity-based approach may facilitate the further development of more convex models. There has been a rich interplay between convex and non-convex methods in machine learning: as one example consider the literature on classification problems, with early work on the perceptron (linear/convex), then work on neural networks with back-propagation (non-linear/non-convex), then the introduction of support vector machines (non-linear/convex), and finally recent work on deep belief networks (non-linear/non-convex). In view of these developments, the lack of convex methods in translation alignment models has been noticeable, and we hope that our work will open up new directions and lead to further progress in this area.

The first part of this thesis focuses on the work in (Simion et al, 2013) and (Simion et al., 2014) [33; 34]: we detail the first convex relaxation of IBM Model 2 and describe an algorithm for its optimization. In particular:

- We introduce a convex relaxation of IBM Model 2. At a very high level, the relaxation is derived by replacing the product $t(f_j|e_i) \times d(i|j)$ with a relaxation that is commonly used in the linear programming literature (e.g., see [5; 6; 1]). (Here $t(f|e)$ are the translation parameters of the model, and $d(i|j)$ are the distortion parameters; the product is non-linear, effectively introducing non-convexity into the problem.)

- We describe an optimization algorithm for the relaxed objective, based on a combination of stochastic subgradient methods with the exponentiated-gradient (EG) algorithm [23; 3].

- We describe experiments with the method on standard alignment datasets, showing that the

EG algorithm converges in only a few passes over the data, and that our method achieves accuracies that are very similar to those of IBM Model 2.

This second technical part of this work is based on (Simion et al., 2015) and (Simion et al., 2015a) [35; 36] and generalizes the above. For this part of the thesis we:

- We introduce a convexification method that may be applicable to a wide range of probabilistic models in NLP and machine learning. In particular, since the likelihood we are optimizing and the metric we are testing against are often not the same (e.g. for alignment tasks we want to maximize F-Measure, but F-Measure is not directly in the likelihood function), different relaxations should potentially be considered for different tasks. The crux of our approach relies on approximating the product function $\prod_{i=1}^{n} x_i$ with a concave function and as a supplement we present some theoretical analysis characterizing concave functions h that approximate this function.

- As a specific application, we introduce a generalized family of convex relaxations for IBM Model 2.[3] As before, the relaxation is derived by replacing the product $t(f_j|e_i) \times d(i|j)$ with $h(t(f_j|e_i), d(i|j))$ where $h(x_1, x_2)$ is a concave upper envelope for $x_1 x_2$. We show how our results encompass the work of (Simion et al., 2013) [33] as a special case. Moreover, the proofs and arguments used to show that the new problem is convex are a generalization of the method used in [33].

- We detail an optimization algorithm for a particularly simple relaxation of IBM Model 2. Unlike the previous work in [33] which relied on a exponentiated subgradient (EG) optimization method and required the tuning of a learning rate, this relaxation can be approached in a much simpler fashion and can be optimized by an EM algorithm that is very similar to the one used for IBM Models 1 and 2. We show that our method achieves a performance very similar to that of IBM Model 2 seeded with IBM 1.

- Lastly, we highlight an application of our new method by showing how it can be applied to make IBM Model 1 both strictly convex and more powerful. Even though IBM Model 1 is

[3]We note that there are negative results which show that certain latent variable problems will have convex relaxations having the uniform solution as optimal [21]. However, for IBM Model 2, the data breaks such symmetries, so any relaxation will be nontrivial.

convex, there is some variance in the AER and F-Measure performance of optimal solutions [41]. We show how to make the IBM Model 1 have 1 unique solution and moreover, using some of the characteristics elements in IBM Model 2, we improve its performance to the level presented in [27] but using a more structured argument.

The third part of this thesis details a HMM relaxation that significantly improves upon the previous work dealing with IBM Model 2's relaxation. This last work is based on (Simion et al., 2015b) [37] and has the following contributions:

- Because finding a relaxation of the HMM proved to be particularly difficult, we combine the HMM and IBM Model 2 structure and we develop a new non-convex model whose performance is close to that of the HMM.
- For the new IBM2-HMM mixture model, we propose a convex relaxation. For both the original model and the new convex formulation, we also derive an EM algorithm that can be used for its optimization.
- We present experiments showing that the new convex model performs significantly better than IBM Model 2 and any of the older convex models developed. Although the new model lags the HMM in performance, it does beat IBM Model 3 and the popular FastAlign IBM Model 2 variant of (Dyer et al, 2013) [16].

The main goal of the new translation models we introduce is to generate quality alignments while at the same time optimizing the log-likelihoods of the models that result. Because our models are convex, gradient based methods are guaranteed to reach a global solution. Although alignment is at the start of the translation pipeline, it is our hope that ultimately more and more complicated non convex models will be replaced by simpler and just as powerful convex models that moreover also have provable gaurantees.

Chapter 2

The IBM Alignment Models

2.1 Convexity and Statistical Machine Translation

(Brown et al., 1993) [8] introduced the original IBM Models and described optimization methods for these models based on the EM algorithm. While the original purpose of these IBM models was to produce automated full translation, they are now mainly used to derive alignments which are then used by modern Statistical Machine Translation (SMT) systems such as phrase-based models. Since the original IBM models were introduced, many variants have been studied in the literature. (Vogel et al., 1996) [43] introduced a model, sometimes referred to as IBM 2.5, which uses a parameterization that is similar to the standard hidden Markov model (HMM) and allows the value of each alignment variable to be conditioned on a previous alignment variable. (Liang et al., 2006) [25] describe a method that explicitly incorporates agreement preferences during training. (Och and Ney, 2003) [28] give a systematic comparison of several alignment models in the literature. (Moore, 2004) [27] gives a detailed study of IBM Model 1, showing various steps that can be used to improve its performance. (Ganchev et al., 2010) [19] describes a method based on posterior regularization that incorporates additional constraints within the EM algorithm for estimation of IBM models. All of these approaches are unsupervised, in that they do not require labeled alignment data; however several authors have considered supervised models (e.g., see [38; 40; 2]). All of he unsupervised variants described above (except [27]) make use of non-concave objective functions during training, with the usual problems with multiple local maxima.

2.2 Convexity and Convex Optimization

For easy reference we list here the key definitions of convexity and convex optimization. Many of these definitions are standard and are found in several sources such as [7]. In what follows, we denote **dom** h by the domain of h.

Definition 1. *A set S is convex if and only if all $x, y \in S$ and all $\theta \in [0,1]$, we have $\theta x + (1-\theta)y \in S$.*

Definition 2. *A function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if and only if **dom** h is convex and and for all $x, y \in$ **dom** h and all $\theta \in [0,1]$, Jensen's inequality holds:*

$$h(\theta x + (1-\theta)y) \leq \theta h(x) + (1-\theta)h(y) \ .$$

Definition 3. *A function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is strictly convex if and only if **dom** h is convex and and for all $x \neq y \in$ **dom** h and all $\theta \in (0,1)$, Jensen's inequality holds:*

$$h(\theta x + (1-\theta)y) < \theta h(x) + (1-\theta)h(y) \ .$$

Definition 4. *A function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is concave if and only if **dom** h is convex and and for all $x, y \in$ **dom** h and all $\theta \in [0,1]$, Jensen's inequality holds:*

$$h(\theta x + (1-\theta)y) \geq \theta h(x) + (1-\theta)h(y) \ .$$

Definition 5. *A function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is strictly concave if and only if **dom** h is convex and and for all $x \neq y \in$ **dom** h and all $\theta \in (0,1)$, Jensen's inequality holds:*

$$h(\theta x + (1-\theta)y) > \theta h(x) + (1-\theta)h(y) \ .$$

Given the above definitions, we define convex optimization problems that are easy to solve and have a given structure (by "easy" here we mean that these problems have algorithms which are off the shelf and have probable guarantees). We have the following:

Definition 6. *A minimization optimization problem*

$$\begin{aligned} \underset{x}{\textit{minimize}} \quad & h_0(x) \\ \textit{subject to} \quad & h_i(x) \leq 0, \ i = 1, \ldots, m. \\ & a_j^T(x) = b_j, \ j = 1, \ldots, l. \end{aligned}$$

is said to be convex if h_i are convex $\forall i$.

Definition 7. *A maximization optimization problem*

$$\begin{aligned} \underset{x}{\textit{maximize}} \quad & h_0(x) \\ \textit{subject to} \quad & h_i(x) \geq 0, \ i = 1, \ldots, m. \\ & a_j^T(x) = b_j, \ j = 1, \ldots, l. \end{aligned}$$

is said to be convex if h_i are concave $\forall i$.

We note that the main issue with the above is that the equality constraints have to be linear (for our running example in this thesis, IBM Model 2, all equality constraints are linear). Under the above setup, it can be shown that the feasible set (the set of points that satisfy the constraints) is convex and that any local optimum for the problem is a global optimum. If h_0 is strictly convex then any local optimum is actually then the unique global optimum.

2.3 The IBM Model 1, IBM Model 2, and HMM Optimization Problems

In this part we give a brief review of IBM Models 1 and 2, the alignment HMM model, and the optimization problems arising from these models. In Chapters 3-5 of this thesis we will derive convex relaxations or IBM Model 2 and the HMM, so it is important to lay the foundation here. The standard approach for optimization within these models is the EM algorithm [15].

Throughout this section, and the remainder of this work, we assume that our set of training examples is $(e^{(k)}, f^{(k)})$ for $k = 1, \ldots, n$, where $e^{(k)}$ is the k'th English sentence and $f^{(k)}$ is the k'th French sentence. Following standard convention, we assume the task is to translate from *French* (the "source" language) into *English* (the "target" language). We use E to denote the English vocabulary (set of possible English words), and F to denote the French vocabulary. The k'th English sentence is a sequence of words $e_1^{(k)}, \ldots, e_{l_k}^{(k)}$ where l_k is the length of the k'th English sentence, and each $e_i^{(k)} \in E$; similarly the k'th French sentence is a sequence $f_1^{(k)}, \ldots, f_{m_k}^{(k)}$ where each $f_j^{(k)} \in F$. We define $e_0^{(k)}$ for $k = 1, \ldots, n$ to be a special NULL word (note that E contains the NULL word). We define $L = \max_{k=1}^n l_k$ and $M = \max_{k=1}^n m_k$. Lastly, throughout this work, for any natural number N, we use $[N]$ to denote $\{1, \ldots, N\}$ and $[N]_0$ to denote $\{0, \ldots, N\}$.

For each English word $e \in E$, we will assume that $D(e)$ is a *dictionary* specifying the set of possible French words that can be translations of e. The set $D(e)$ is a subset of F. In practice, $D(e)$ can be derived in various ways, the most standard one being to simply define $D(e)$ to include all French words f such that e and f are seen in a translation pair (we adopt this convention in our model formulations).

2.3.1 Noisy-Channel Approach to SMT

We review here the noisy-channel approach to SMT, basing our discussion on (Collins, 2015) [12] and (Koehn, 2008) [24].

The noisy-channel approach to SMT was developed over twenty years ago and is a way to model translation while at the same time utilizing as much information as possible. For a translation task, we want to translate a source French sentence f into a target English sentence e. To do this, we need to find $\hat{e} = \arg\max_{e \in \mathbf{E}} p(e|f)$, where we take the maximum over the entire set of English sentences $\mathbf{E}$. However, we notice that by Bayes' rule we also have that $p(e|f) = \frac{p(f|e)p(e)}{p(f)}$ and hence the problem we need to solve is equivalent to

$$\hat{e} = \arg\max_{e \in \mathbf{E}} p(f|e)p(e) \ .$$

The above formulation is used in SMT for two reasons.

- First, it allows us to model and use $p(e)$ to introduce grammaticality and fluency into the choice of which English sentence to choose.
- Second, we still are left with modeling the purely translational component $p(f|e)$, which can be approached in the same fashion as the direct modeling of $p(e|f)$.

Thus, the noisy channel approach in some sense breaks up the problem of finding the best sentence $\hat{e}$ into two parts and we can use different techniques and models to decide on $\hat{e}$. In this work, we focus on models that address $p(f|e)$, the translation component. In this thesis we will introduce IBM Model 1, 2, and the alignment HMM model. Each of these different models will have a different structure for their associated terms $p(f|e)$. For each of these models, we will study how we can estimate the parameters of each of these model given a structure on $p(f|e)$ and bitext training data $(e^{(k)}, f^{(k)})$ $k = 1, \ldots, N$.

The word-based translation models we explore in this thesis are still very useful for a number of reasons, including:

- The IBM models make use of the key alignment idea found in translation. Acting as a latent variable for these models, the alignment idea is still used in modern SMT systems to generate phrase tables, which are the basic blocks on translating one piece of language into another (see Chapter 1 for an example).
- The parameters of the IBM models are estimated using the Expectation Maximization (EM) algorithm. For latent variable models, the EM algorithm is pervasive for parameter optimization, and the IBM Models serve as a natural application.

2.3.2 Alignments

For any French sentence $f = (f_1, \ldots, f_m)$ paired with an English sentence $e = (e_1, \ldots, e_l)$, we now turn to the construction and development of the conditional probability $p(f|e)$. For the models we study, writing out $p(f|e)$ is too complicated directly, and we need an intermediate (or latent) alignment variable. For each word f_j we associate a corresponding alignment variable a_j in $\{0, \ldots, l\}$ that tells which word $e_i \in e$ in is "aligned" or generates f_j. In particular, it will turn out that the e_{a_j} is the word which generates f_j.

Given the introduction of alignment, we then have that $p(f, a|e)$ will be defined and (by summing) we can now write

$$p(f|e) = \sum_{a=(a_1,\ldots,a_m)} p(f, a|e) \ .$$

Lastly, we mention that although the number of alignment variables is exponential, there are several tricks that will allow us to compute $p(f|e)$ efficiently, and it is this last simplification will be a key element in bringing out the new results of this thesis.

We now describe the alignment variables in detail using some concrete examples for motivation. We recall that each alignment variable a_j specifies that the French word f_j is aligned to the English word e_{a_j}. Since in some languages certain source words do not generate other target words, we define e_0 to be a special NULL word so that $a_j = 0$ specifies that word f_j is generated from the NULL word.

Consider a pair of translations (e, f) with

$$e = \text{And the program has been implemented}$$

and

$$f = \text{Le programme a ete mis en application .}$$

For this example the length of f, m, is equal to 7 and the length of e, l, equal to 6. In this example we thus have alignment variables $a_1, a_2, \ldots a_7$ that take values in $\{0, \ldots, 6\}$. As mentioned previously, we have an exponential number of possible alignments, but one very plausible alignment would be

$$(a_1, a_2, \ldots, a_7) = (2, 3, 4, 5, 0, 0, 6) \ .$$

Using the definitions of an alignment, the above setup would specifying the following:

$$Le \to the$$

$$Programme \to program$$

$$a \to has$$

$$ete \to been$$

$$mis \to NULL$$

$$en \to NULL$$

$$application \to implemented$$

In the above example, we note that each French word is aligned to exactly one English word, but there are cases of English words (such as “And”) that are not aligned to any French words. Given this, we see that the alignment is neither bijective nor symmetric, and several French words can be aligned to the same source word (for example, we have “mis” and “en” aligned to NULL). The asymmetric nature of the alignments is one of the key features of the IBM models: each English word can be aligned to any number (zero or more) French words (we will return to this point later but note here that certain models such as IBM 3 implement the concept of “fertility" whose main goal is to discourage having a particular English word being aligned to too many French words). As an even more extreme example alignment, we could have

$$(a_1, a_2, \ldots, a_7) = (1, 0, 0, 1, 1, 0, 0) \ ,$$

specifying the following alignment:

$$\begin{aligned} Le &\rightarrow And \\ Programme &\rightarrow NULL \\ a &\rightarrow NULL \\ ete &\rightarrow And \\ mis &\rightarrow And \\ en &\rightarrow NULL \\ application &\rightarrow NULL \end{aligned}$$

The above is clearly a poor alignment for this example and a good model would discourage this alignment from having a very high probability. In the subsequent sections, we will specify how each of IBM Model 1, IBM Model 2 and the alignment HMM model the terms $p(f, a|e)$. In some sense, the main difference in performance between these models can be traced to exactly how they define $p(f, a|e)$.

2.3.3 IBM Model 2

Given the alignment setup above, the IBM Model 2 optimization problem is given in Figure 2.1. The parameters in this problem are $t(f|e)$ and $d(i|j)$. The $t(f|e)$ parameters are *translation* parameters specifying the probability of English word e being translated as French word f. The *distortion* parameters $d(i|j)$ specify the probability of the j'th French word in a sentence being aligned to the i'th English word (or, put another way, that alignment variable $a_j = i$). We use a variant of IBM Model 2 where the distortion variables are shared across all sentence lengths (similar variants have been used in [25] and [24]). Using the parameters of IBM2, we then have that

$$p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l) = \prod_{j=1}^{m} t(f_j | e_{a_j}) d(a_j | j) \ . \tag{2.6}$$

Moreover, the objective function is then the log-likelihood of the training data (see Eq. 2.5):

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log p(f_j^{(k)} | e^{(k)}) \ , \tag{2.7}$$

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2.3.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.

Constraints:

$$\forall e \in E, f \in D(e), \qquad t(f|e) \geq 0 \tag{2.1}$$

$$\forall e \in E, \qquad \sum_{f \in D(e)} t(f|e) = 1 \tag{2.2}$$

$$\forall i \in [L]_0, j \in [M], \qquad d(i|j) \geq 0 \tag{2.3}$$

$$\forall j \in [M], \qquad \sum_{i \in [L]_0} d(i|j) = 1 \tag{2.4}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) d(i|j) \tag{2.5}$$

with respect to the $t(f|e)$ and $d(i|j)$ parameters.

Figure 2.1: The IBM Model 2 Optimization Problem.

where

$$p(f_j^{(k)}|e^{(k)}) = \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)})d(i|j) \ .$$

Some explanation of the preceding formula is required. For IBM Model 2 (in fact, for any of the IBM Models), the main objective is to maximize the log-likelihood of the data given by

$$\frac{1}{n}\sum_{k=1}^{n} \log p(f^{(k)}|e^{(k)}) \ , \tag{2.8}$$

To simplify the above formula, we need to know the underlying latent variables a which is not observed from the data. Given an alignment, the generative rule for IBM Model 2 is given by Eq. 2.6, and we note that for an (e, f) pair we have exponentially many choices for $a = (a_1, \dots, a_m)$ since each $a_j \in [l]_0$. What this implies is that actually the log-likelihood of IBM Model 2 is given by

$$\frac{1}{n}\sum_{k=1}^{n} \log \sum_{a^{(k)}} p(f^{(k)}, a^{(k)}|e^{(k)}) \ , \tag{2.9}$$

where the sum inside the logarithm is exponential in magnitude. However, we also note algebra implies that

$$\sum_{a^{(k)}} p(f^{(k)}, a^{(k)}|e^{(k)}) = \prod_{j=1}^{m_k} \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)})d(i|j) \ . \tag{2.10}$$

The above simplification allows us to rewrite the objective of IBM Model 2 (and, as we will see later, several other models) in polynomial time as in Figure 2.1.

Crucially, while the constraints in the IBM Model 2 optimization problem are linear, the objective function in Eq. 2.5 is non-concave. Therefore, optimization methods for IBM Model 2, in particular the EM algorithm, are typically only guaranteed to reach a local maximum of the objective function. The standard EM algorithm for IBM Model 2 is detailed below. We note that the algorithm is guaranteed to find a local optima without the use of a learning rate.

2.3.4 IBM Model 1

Figure 2.3 shows the optimization problem for IBM Model 1. In IBM Model 1 the distortion parameters $d(i|j)$ are all fixed to be the uniform distribution (i.e., $1/(L+1)$). The objective function for

```
1: **Input**: Define $E$, $F$, $L$, $M$, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$. An integer $T$ specifying the number of passes over the data.
2: **Parameters:**
   • A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
   • A parameter $d(i|j)$ for each $i, \in [L]_0, j \in [M]$.
3: **Initialization:**
   • $\forall e \in E,\ f \in D(e)$, set $t(f|e) = 1/|D(e)|$.
   • $\forall i \in [L]_0,\ j \in [M]$, set $d(i|j) = 1/(L+1)$.
4: **EM Algorithm:**
5: for all $t = 1 \dots T$ do
6:    $\forall e \in E,\ f \in D(e),\ count(f, e) = 0$
7:    $\forall e \in E,\ count(e) = 0$
8:    $\forall i \in [L]_0, j \in [M]$, $count(i, j) = 0$
9:    $\forall j \in [M]$, $count(j) = 0$
10:   **EM Algorithm: Expectation**
11:   for all $k = 1 \dots n$ do
12:      for all $j = 1 \dots m_k$ do
13:         $\delta[i] = 0\ \forall i \in [l_k]_0$
14:         $\Delta = 0$
15:         for all $i = 0 \dots l_k$ do
16:            $\delta[i] = t(f_j^{(k)}|e_i^{(k)})d(i|j)$
17:            $\Delta \mathrel{+}= \delta[i]$
18:         for all $i = 0 \dots l_k$ do
19:            $\delta[i] = \frac{\delta[i]}{\Delta}$
20:            $count(f_j^{(k)}, e_i^{(k)}) \mathrel{+}= \delta[i]$
21:            $count(e_i^{(k)}) \mathrel{+}= \delta[i]$
22:            $count(i, j) \mathrel{+}= \delta[i]$
23:            $count(j) \mathrel{+}= \delta[i]$
24:   **EM Algorithm: Maximization**
25:   for all $e \in E$ do
26:      for all $f \in D(e)$ do
27:         $t(f|e) = \frac{count(e,f)}{count(e)}$
28:   for all $\forall i \in [L]_0, j \in [M]$, do
29:      $d(i|j) = \frac{count(i,j)}{count(j)}$
30: **Output:** $t$, $d$ parameters.
```

Figure 2.2: Pseudocode for T iterations of the EM Algorithm for the IBM Model 2 problem.

IBM Model 1 can be shown to be concave, so the EM algorithm will converge to a global maximum. However IBM Model 1 is much weaker than Model 2, and typically gives far worse performance. A common heuristic is to initialize the $t(f|e)$ parameters in EM optimization of IBM Model 2 using

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2.3.

Parameters:

• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), \qquad t(f|e) \geq 0 \tag{2.11}$$

$$\forall e \in E, \qquad \sum_{f \in D(e)} t(f|e) = 1 \tag{2.12}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)} \tag{2.13}$$

with respect to the $t(f|e)$ parameters.

Figure 2.3: The IBM Model 1 Optimization Problem.

the output from IBM Model 1. The intuition behind this heuristic is that the IBM Model 1 values for $t(f|e)$ will be a reasonable starting point, and the EM algorithm will climb to a "good" local optimum. We are not aware of any guarantees for this initialization heuristic, however.

2.3.5 The HMM Alignment Model

Before discussion the specifics of the alignment HMM model, we should review the general HMM model structure so that we better understand the alignment variant we study. At a high level, the HMM is a latent variable model that has the latent variable "jump" in between latent states and generate signals. In training, one only has access to the signals $(x_1, \ldots, x_m)$ and is interested in estimating model parameters. Once again, $p(x_1, \ldots, x_m)$ is not readily gotten but once we know the latent states $(y_1, \ldots, y_m)$ that generate $(x_1, \ldots, x_m)$ we have that

$$p(x_1, \ldots, x_m, y_1, \ldots, y_m) = \prod_{j=1}^{m} p(x_j|y_j)p(y_j|y_{j-1}) \ ,$$

where we set y_0 as some special constant. We note that the standard HMM has two type of parameters, namely *emission* and *transition* probabilities. The emission probabilities model how we jump from one latent step to the other, while the transition probabilities model how we generate the signal x_j given the latent state y_j.

For the HMM alignment model, the distortion parameters act as the transition parameters $d(a_j|a_{j-1}, l)$. These transition parameters specify the probability of the next alignment variable for the j^{th} target word is a_j, given that the previous target word was aligned to a source word whose position was a_{j-1} in a source sentence with length of l. For the emission probabilities, we have the standard lexical probabilities $t(f_j|e_{a_j})$ and we note that the alignment HMM variant has e_{a_j} as the latent generating state (a standard model would have just a_j). Figure 2.4 summarizes the HMM alignment model. We note that unlike IBM Models 1 and 2, we cannot simplify the exponential sum within the likelihood of the HMM. The ramifications of this inability to simplify the log-likelihood is important in EM training for this model: the HMM makes use of a special EM implementation knows as the Baum-Welch algorithm [30]. Once again, the HMM optimization can be shown to be non-convex. Since the HMM shares the same lexical t parameters as IBM 1 and 2, one can optimize the HMM likelihood by training IBM1, giving the t parameters to IBM2 and then optimizing IBM2 until convergence, and finally giving the IBM2 t parameters to the HMM and running EM training on the HMM. Although, as with IBM Model 2, the preceding initialization heuristics have no provable guarantee on performance, they serve as a good way to optimize a non-convex model such as the HMM and get a sensible local solution.

2.4 Extracting the Viterbi Alignments

Once we have optimized the parameters for one of the models above, we are chiefly concerned with extracting the most likely alignments. For a particular sentence pair (e, f) with lengths of l and m respectively this boils down to finding the vector $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_m)$ such that

$$\hat{a} = \arg\max_a p(f, a|e) \ .$$

For IBM Model 1 the above becomes

$$\hat{a} = \arg\max_a \prod_{j=1}^{m} t(f_j|e_{a_j})$$

and this last computation can be simplified so that $\hat{a}_j = \arg\max_{i=0}^{l} t(f_j|e_i)$ and each component can be recovered independent of its neighbor. Similarly, for IBM Model 2 we have $\hat{a}_j = \arg\max_{i=0}^{l} t(f_j|e_i)d(i|j)$. For the HMM, this last computation is more involved and requires the use of dynamic programming as detailed in [43]. Typically the alignment quality of a model is gotten by

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|i, l_k)$ for each $i \in [l_k]_0$, $i' \in [l_k]_0$.

Constraints:

$$\forall e \in E, f \in D(e), t(f|e) \geq 0 \tag{2.14}$$

$$\forall e \in E, \sum_{f \in D(e)} t(f|e) = 1 \tag{2.15}$$

$$\forall i, i' \in [l_k]_0, d(i'|i, l_k) \geq 0 \tag{2.16}$$

$$\forall i \in [l_k]_0, \sum_{i' \in [l_k]_0} d(i'|i, l_k) = 1 \tag{2.17}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{a_1^{(k)} \dots a_{m_k}^{(k)}} \log \prod_{j=1}^{m_k} t(f_j^{(k)} | e_{a_j^k}^{(k)}) d(a_j^{(k)} | a_{j-1}^{(k)}, l_k)$$

with respect to the $t(f|e)$ parameters $d(i'|i, l)$.

Figure 2.4: The HMM Optimization Problem

comparing the model's alignments to a hand derived set of alignments by computing a score over the two results such as F-Measure. We discuss this further in the experiments sections inside the main chapters of this work.

2.4.1 Alignment Evaluation

Once we obtain the final highest probability alignments, we can evaluate the quality of these alignments against a manually (human) generated set of alignment. Model alignments and human generated alignments can be put into formatted text files and the similarity of these alignments can be used to assign a score signifying of the model's quality. There are a number of useful measures for alignment quality, including Alignment Error Rate (AER) and F-Measure [28]. Typically, for the hand aligned test data, the annotated alignments between source and target words in the corpus are marked as either "sure" (S) or "possible" (P) alignments, as described in [28]. If A is the set of alignments produced by an algorithm, S is the set of sure alignments as annotated in test data, and P is the set of possible alignments, then the key alignment quality quantities are defined as

$$\text{Recall} = \frac{|A \cap S|}{|S|} \ ,$$

$$\text{Precision} = \frac{|A \cap S|}{|A|} \ ,$$

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \ ,$$

$$\text{F-Measure} = \frac{1}{\frac{.5}{\text{Recall}} + \frac{.5}{\text{Precision}}} \ .$$

Using the above, we note that for AER lower is better while for F-Measure higher is better. Moreover, once we get the alignment quality, evaluating the final translation result (typically using the BLEU metric [24]) is also of value as the alignments and the downstream translations are related (alignments are used to seed the more complicated translation models). In our experiments, we will see several concrete examples of the above.

2.5 A Literature Survey of Alignment Models

In this section we review the literature on alignment models. Being only 20 years old, the field of Statistical Machine Translation (SMT) is new, but very rich. Alignment is a very active part of modern day Natural Language Processing (NLP) research.

The paper of (Brown et al., 1993) [8] gave rise to the field of SMT. In this work, the authors introduced word-based translation via a string of increasingly more complex (and powerful) models. Specifically, this work introduced IBM Models 1 to 5 and also studied their parameter optimization, which is accomplished via the EM algorithm [15]. For IBM Models 1 and 2, the EM algorithm has a closed form in the sense that both the E and M steps are tractable. Moreover, since IBM Model 1 is a convex optimization problem, the EM algorithm applied to IBM1 is guaranteed to converge to an optimal solution. For IBM Model 2, EM training has no formal guarantees other than that it will converge to a local solution. Recall that main variable types in IBM Model 2 are lexical parameters ($t(f|e)$) which model the probability that a word generates another word, and distortion parameters ($d(i|j)$) which model the positional dependency between a source word (in position i) and a target word (in position j). Besides having the lexical and distortion terms, the other IBM Models each in turn add more structure. For example, IBM Model 3 adds in a“fertility term" which limits the number of target words that a source word may generate. Although these models perform better then Models 1 and 2 [28], their structure is far more complicated, and because of this the EM implementation for these models is inexact. Specifically, the E and M steps for Model 3 onwards are based on heuristic search and are quite intractable problems on their own. Nevertheless, experimental research has shown that the IBM Models do improve in performance as one goes up in the ladder. Moreover, to address this complexity and ensure proper initialization, one typically runs IBM Model 1, then seeds Model 2 with 1, then seed Model 3 with 2, and so on. Although this method of optimization is a heuristic, it has been shown to perform well [28].

The most natural addition to the original IBM Models was the work of (Vogel et al., 1996) [43]. In this last work, word-based translation was achieved by fitting a hidden Markov Model (HMM) [30] to parallel bitext data. Known colloquially as IBM Model 2.5, the alignment HMM model was (and still is) a very powerful addition to the IBM family of models for several reasons. For one, the alignment HMM Model, like IBM Models 1 and 2, has an exact EM algorithm which requires no heuristics in the E and M steps. Moreover, the alignment HMM Model performs very well: it typically archives performance on par with the much more complicated (and standard) IBM Model 4 [28; 24]. Although the HMM alignment Model is still a non-convex optimization, the model has influenced the field greatly and new variants of it have been proposed [25; 45].

The above IBM Models serve as the core word-based translation methods. As the field evolved, researchers found that it was not words but phrases which should serve as the fundamental building blocks for translation [24]. Nowadays, the IBM Models are no longer used to translate, but their alignment (translation) tables are used to generate phrases and these phrases are used to seed more complicated models which then perform the translation task. Additionally, there was found to be a (arguably complex) relationship between alignment and translation quality, so alignment has been a topic of much research. We now review several other works which have been important to the development of the field and this thesis.

One of the seminal papers on the IBM alignment models was (Och and Ney, 2003) [28]. In this work, the authors evaluate the performance of the IBM Models by introducing metrics such as F-Measure and alignment error rate (AER) and evaluating the performance of the models on several different datasets and training regimes. Specifically, the authors of this last work compare the less complicated IBM Model 1 to running IBM Model 2 after it was seeded by IBM Model 1 and find that in this simple scenario (and in others such as running Model 3 seeded with Model 2) alignment performance improves. Additionally, the experiments developed here have served as a baseline for the setup used in several different sources: like in this work, most new models are non-convex and do benefit from being seeded by less complicated models. (as a caveat, however, we note that there are cases where this is not the scenario [16], but typically we do not have any proofs on why this is the case). The empirical results of this work showed that although there is some improvement to translation quality as the alignment models get more and more complicated, the actual performance experiences diminishing returns. As a result of this, the standard method of evaluating a new model nowadays is to compare with IBM Model 4 seeded with its preceding models (Models 5 and 6 are largely ignored). Finally, as a consequence of this work, the authors also developed a standardized implementation of the IBM Models called GIZA++ [24]. Even today, GIZA++ is arguably the most popular alignment generator.

Aside from the above work, another important evaluation work was that of (Fraser and Marcu, 2007) [18]. In this paper, the authors try to not only look at model performance, but also on *which* metric to use. In particular, since the SMT pipeline is so long and we are mainly interested in BLEU score, it is important to choose between AER and F-Measure (typically, a model that does well on AER also does well on F-Measure; however, achieving the best performance on one metric does not usually yield the bet performance on another). From the experiments, it seems that F-Measure is the better of the two to use. Moreover, the balance between Recall and Precision within F-Measure is something that is *language dependent.*

Spurred on by the above experimental exercises, other papers on the IBM Models and alignment appeared frequently in the last decade. Among the papers on IBM Model 1, the paper by (Moore, 2004) [27] stands out. In they work, the author discusses some very small changes to IBM Model 1 which can be used to dramatically improve its performance. Among these changes, the paper argues that one should (1) trim the dictionary counts in the lexical parameters (this effectively reduces the number of variables $t(f|e)$) (2) use an initialization heuristic and (3) modify IBM Model 1's EM algorithm by using smoothing. From these changes, several interesting observations can be drawn. To start, dictionary filtering might be a very powerful heuristic: even though a word e is seen with a word f in some bitext sentence, filtering out f from the possible translations of e would allow the lexical probabilities $t(f|e)$ to focus on what is important and performance would increase dramatically. Indeed, much of (Moore, 2004) focuses on fixing the "garbage collection problem" which has rare words tending to act as garbage collectors and align to too many other words. Moreover, from the fact that we have a significant AER decrease by using an initialization heuristic, we infer that IBM Model 1 does tend to over train quite severely, and using smoothing within the EM algorithm fixes the overtraining quite a bit. Although very interesting, the work also has its issues as the author introduces several hyper parameters and increases a fairly simple model to a more complicated level without any formal guarantees. Of course, because of the sheer size of modern NLP problems, empirical evaluation may just as well suffice.

Other work on IBM Model 1 also found very interesting ideas via simple observation. In particular, the work of (Toutanova et al., 2011) [41] explored IBM Model 1 and found that it was not strictly convex. For the IBM Models, the optimal parameters are used to decode and AER (which measures alignment quality) is not within the log-likelihood of the data, so having many optimal solutions with a high variance among the alignment quality results would not be good. By formulating the optimal solution set as a (feasibility, or existence) linear program, the authors show that IBM Model 1 has quite a bit of variance within its optimal solution set. This was a stark realization, as it shows that even with convex models one needs to be careful when optimizing: getting the optimal solution may not be enough.

Another seminal work on the IBM Models was developed in (Liang et al., 2006) [25]. When training the IBM Models and getting the optimal alignment, one trains both source to target and target to source models and then intersects alignments. This intersection step is, again, heuristic and one either gets the intersection or the union or performs a combination of these two steps [24]. What the authors of this work did was force agreement *during* training, not after. The results of the paper were quite strong. Although models do benefit greatly by intersecting alignments, having the two models agree during training improves the AER and F-Measure of the alignment task by over 30%. Applying their method to both the HMM and IBM Model 2, the authors showed that EM training for their implementation is rather simple, and moreover, they also developed a new method of decoding via *posterior decoding.*

Bayesian methods have also found their way into alignment research. In particular, (Riley and Gildea, 2012) [31] showed how to improve the IBM Models by using Variational Bayes. The idea of this paper is to limit the "garbage collection" affect by instilling a prior on the IBM Models which limits this type of behavior. In the paper, the EM algorithm for IBM Models 1 and 2 is modified to accept a prior, new experiments are conducted, and using the same data as (Och and Ney, 2003) the authors make considerable gains. As the authors discuss, this work can be viewed as a more theoretical justification of the methods discussed in (Moore, 2004) [27]. Finally, we mention that [31] is not the only work to use a Bayesian prior in conjunction with the IBM Models. Indeed, work done by (Vaswani et al., 2012) [42] also went in this direction, but in this case the focus was on applying a smooth approximation of the l_0 (sparsity) norm.

Since the EM algorithm is so pervasive in NLP literature, some new research has focused on introducing new regularization techniques within the alignment literature. In particular, the work of (Graca et al., 2010) [20; 19] introduces posterior regularization and the authors modify IBM 1 and the HMM by inserting *fertility* and *symmetry* constraints that improve both of these models substantially. The chief goal of these constraints is to correct EM overtraining and also instill new features into the tractable IBM Models so that a proper EM algorithm (one that does not have heuristics such as local search within the E and M steps), can be applied.

Recently, a new alignment system developed at Carnegie Mellon University and called cdec [17] has become very popular. A central component of this pipeline is the alignment model, which is a modified IBM Model 2 variant called FastAlign that introduces some new ideas and makes use of Variational Bayes [31]. Specifically, aside from using Bayesian methods, the new model also forces the distortion parameters to favor words on the diagonal in the sense that if we tokenize the source and target sentences and place word positions into two vectors we make the distortions for that pair favor alignments on the first bisector. Additionally, using this parametrization and a set of elementary rules about geometric series, this approach allows one to develop a very fast EM algorithm that runs orders faster than the standard GIZA++ IBM Model 4 method which runs the traditional IBM Models in sequence. Using this new model, the authors also achieve some favorably comparisons against IBM Model 4 in terms of AER and translation (BLEU [29]) quality.

There has been a lot of work on the IBM Models and their role in the translation pipeline. SMT research is an area filled with active research and application that combines multiple techniques from varied disciplines.

Chapter 3

A Convex Alternative to IBM Model 2

3.1 A Convex Relaxation of IBM Model 2

We now introduce a convex optimization problem, the I2CR (IBM 2 Convex Relaxation) problem. As its name suggests, this optimization problem is closely related to IBM Model 2, but is convex. Because of this it will be relatively easy to derive an optimization algorithm that is guaranteed to converge to a global optimum. Our experiments show that the relaxation gives very similar performance to the original IBM 2 optimization problem, as described in the previous chapter.

We first describe an optimization problem, I2CR-1, that illustrates the basic idea behind the convex relaxation. We then describe a refined relaxation, I2CR-2, that introduces a couple of modifications, and which performs well in experiments.

3.1.1 The I2CR-1 Problem

The I2CR-1 problem is shown in Figure 3.1. A first key idea is to introduce a new variable $q(i,j,k)$ for each $k \in [n]$, $i \in [l_k]_0$, $j \in [m_k]$: that is, a new variable for each triple (i,j,k) specifying a sentence pair, and a specific English and French position in that sentence. Each q variable must satisfy the constraints in Eqs. 3.5-3.7, repeated here for convenience:

$$\forall i,j,k, \qquad q(i,j,k) \geq 0 \ ,$$

$$\forall i,j,k, \qquad q(i,j,k) \leq d(i|j) \ ,$$

$$\forall i,j,k, \qquad q(i,j,k) \leq t(f_j^{(k)}|e_i^{(k)}) \ .$$

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3.

Parameters:

- A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
- A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.
- A parameter $q(i,j,k)$ for each $k \in [n]$, $i \in [l_k]_0$, $j \in [m_k]$.

Constraints:

$$\forall e \in E, f \in D(e), \quad t(f|e) \geq 0 \tag{3.1}$$

$$\forall e \in E, \quad \sum_{f \in D(e)} t(f|e) = 1 \tag{3.2}$$

$$\forall i \in [L]_0, j \in [M], \quad d(i|j) \geq 0 \tag{3.3}$$

$$\forall j \in [M], \quad \sum_{i \in [L]_0} d(i|j) = 1 \tag{3.4}$$

$$\forall i,j,k, \qquad q(i,j,k) \geq 0 \tag{3.5}$$

$$\forall i,j,k, \qquad q(i,j,k) \leq d(i|j) \tag{3.6}$$

$$\forall i,j,k, \qquad q(i,j,k) \leq t(f_j^{(k)}|e_i^{(k)}) \tag{3.7}$$

Objective: Maximize

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} q(i,j,k) \tag{3.8}$$

with respect to the $q(i,j,k)$, $t(f|e)$ and $d(i|j)$ parameters.

Figure 3.1: The I2CR-1 (IBM 2 Convex Relaxation) Problem, version 1.

The objective function is

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} q(i,j,k)$$

which is similar to the objective function in Figure 2.1, but where $t(f_j^{(k)}|e_i^{(k)}) \times d(i|j)$ has been replaced by $q(i,j,k)$. The intuition behind the new problem is as follows. If, instead of the constraints in Eqs. 3.5-3.7, we had the constraint

$$q(i,j,k) = t(f_j^{(k)}|e_i^{(k)}) \times d(i|j) \ , \tag{3.9}$$

then the I2CR-1 problem would clearly be identical to the IBM Model 2 optimization problem. We have used a standard relaxation of the non-linear constraint $x = y \times z$ where x, y, z are all variables in the range $[0, 1]$, namely

$$\begin{aligned} x &\leq y \ , \\ x &\leq z \ , \\ x &\geq y + z - 1 \ . \end{aligned}$$

These inequalities are a relaxation in the sense that any (x, y, z) triple that satisfies $x = y \times z$ also satisfies these constraints. Applying this relaxation to Eq. 3.9 gives

$$\begin{aligned} q(i,j,k) &\leq t(f_j^{(k)}|e_i^{(k)}) \ , \\ q(i,j,k) &\leq d(i|j) \ , \\ q(i,j,k) &\geq t(f_j^{(k)}|e_i^{(k)}) + d(i|j) - 1 \ . \end{aligned} \tag{3.10}$$

The final observation to note is that the constraint in Eq. 3.10 can be omitted in the I2CR-1 problem. This is because the task is to maximize the objective with respect to the q variables and the objective is strictly increasing as the q values increase—thus lower bounds on their values are redundant in the I2CR-1 problem.

It is easily verified that the constraints in the I2CR-1 problem are linear, and that the objective function is convex. In Section 3.3 of this paper we describe an optimization method for the problem.

Note that because the objective function is being maximized, and the objective increases monotonically as the q values increase, at the global optimum[1] we have

$$q(i,j,k) = \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\} \ ,$$

where $\min\{x, y\}$ returns the minimum of the two values x and y. Thus, we could actually eliminate the q variables and write an optimization problem that is identical to the IBM Model 2 optimization problem, but with the objective function

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k}\min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\} \ .$$

[1]More precisely, at *any* global optimum: the objective function may not be strictly convex, in which case there will be multiple global optima.

Input: Same as in I2CR-1 (Figure 3.1). **Parameters**: Same as in I2CR-1 (Figure 3.1).

Constraints: Same as in I2CR-1 (Figure 3.1).

Objective: Maximize

$$\frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} q(i,j,k) \tag{3.11}$$

$$+\quad \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)} \tag{3.12}$$

with respect to the $q(i,j,k)$, $t(f|e)$ and $d(i|j)$ parameters.

Figure 3.2: The I2CR-2 (IBM 2 Convex Relaxation) Problem. The problem is identical to the I2CR-1 problem, but it also includes a term in the objective function that is identical to the IBM Model 1 objective. We define $\log'(z) = \log(z+\lambda)$ where λ is a small positive constant.

It will turn out that both views of the I2CR-1 problem—with and without the q variables—are helpful, so we have included both in our discussion.

3.1.2 The I2CR-2 Problem

Figure 3.2 shows the refined optimization problem, which we call I2CR-2. The problem incorporates two modifications. First, we modify the objective function to be

$$\frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} q(i,j,k)$$

$$+\quad \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k} \frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)}\ .$$

Thus the objective function includes a second term that is identical to the objective function for IBM Model 1 (see Figure 2.3). In preliminary experiments with the I2CR-1 optimization problem, we found that the I2CR-1 objective was not sufficiently dependent on the interaction between the t and d parameters and its performance was very close to that of IBM Model 1; intuitively, if the d parameters achieve the min on many training examples, the values for the t variables become unimportant. The addition of the IBM Model 1 objective fixed this problem by introducing a term that depends on the t values alone and ensured that this variable is given the necessary importance often.

Second, we replace log by log$'$, where $\log'(z) = \log(z + \lambda)$, and λ is a small positive constant (in our experiments we used $\lambda = 0.001$). Under this definition the derivatives of log$'$ are upper-bounded by $1/\lambda$, in contrast to log, where the derivatives can diverge to infinity. The optimization methods we use are gradient-based methods (or more precisely, *subgradient*-based methods), and we have found them to be considerably more stable when the values for gradients do not diverge to infinity. Moreover, in order to have the optimization methods we describe in Section 3.2 apply on a theoretical and practical level, we will need to have bounded subgradients, and this modification of the log yields precisely this. Finally, we note that the modified objective remains convex.

3.2 Convex Optimization via Exponentiated-Gradient Descent

In this section we discuss Exponentiated-Gradient (EG) Decent, an algorithm which we will adapt for optimizing our new convex IBM Model 2 variant. EG algorithms are gradient-based methods that maintain simplex constraints; see for example: [23; 3; 13]. The discussion we present here is adapted from (Kakade, 2011) [22]. The key feature of EG-based algorithms will be that the projection step is *given* to us and will not require solving another intermediate problem. Specifically, because all of our variables are on a probability simplex, the algorithm readily applies.

EG based optimization is introduced most readily by looking at the online optimization setting [23]. In this case, we have a sequence of times $t = 1, \ldots, T$ and receive convex functions c_t for each time t. At each t, choose actions w_t and the end goal is to minimize the regret

$$R_T(EG) = \sum_{t=1}^{T} c_t(w_t) - \inf_{w \in D} \sum_{t=1}^{T} c_t(w) \ .$$

In what follows we assume that w^* is an optimal solution to $\inf_{w\in D} \sum_{t=1}^{T} c_t(w)$. To start, assume the decision space D is a d-dimensional simplex, i.e.

$$D = \{w | w_i \geq 0 \text{ and } ||w||_1 = 1\} \ .$$

The Exponentiated-Gradient Descent algorithm is defined as follows: at time $t = 1$, choose w_1 as the center point of this scaled simplex, namely $w_{1,i} = \frac{1}{d}$, and then use the update:

$$w_{t+1,i} = \frac{w_{t,i} \exp(-\gamma[\nabla c_t(w_t)]_i)}{Z_t} \ ,$$

where

$$Z_t = \sum_{j=1}^{d} w_{t,j} \exp(-\gamma[\nabla c_t(w_t)]_j) \ .$$

Here $[\nabla c_t(w_t)]_i$ denotes the i^{th} component of the gradient vector. The division by Z_t serves as a form of normalization, so that we have $w_{t+t} \in D$, i.e. $||w_{t+1}||_1 = 1$.

The main result we base our analysis on for I2CR-1 and 2's optimization is below, and we include the proof given in [22] for completeness. For this proof, we set $\nabla_t = \nabla c_t$ to simplify notation.

Theorem 1. *Assume that D is a simplex and assume that the gradient is bounded as follows:*

$$||\nabla c_t(w_t)||_\infty \leq G_\infty$$

where $||u||_\infty = \max_i |u_i|$ is the L_∞. If $\gamma = \frac{1}{G_\infty}\sqrt{\frac{\log d}{T}}$, the regret of EG at time T is bounded as:

$$R_T(EG) \leq 2G_\infty\sqrt{T \log d}$$

Proof. We start by noting that

$$RG(T) = \frac{1}{T}\sum_{t=1}^{T}(c(w_t) - c(w^*)) \leq \frac{1}{T}\sum_{t=1}^{T}\nabla_t(w_t - w^*) \ .$$

The key result is that we have

$$\sum_{t=1}^{T}\nabla_t(w_t - w^*) \leq \frac{KL(w^*||w_1)}{\gamma} + \gamma G_\infty^2 T \ .$$

In the above, the KL terms is the $Kullback - Leiber$ [7] divergence between two vectors which is given by $KL(u||v) = \sum_i u_i \log \frac{u_i}{v_i}$. First, $\exp(x) \leq 1 + x + x^2$, if $x \leq 1$. Let us examine how the KL divergence changes with respect to w^*. We have

$$\begin{aligned} KL(w^*||w_t) - KL(w^*||w_{t+1}) &= \sum_i w_i^* \log \frac{w_{t+1,i}}{w_{t,i}} \\ &= \sum_i w_i^*(-\gamma\nabla_{t,i} - \log(Z)) \\ &= -\gamma w^* \nabla_t - \log(Z) \end{aligned}$$

Now let us use the inequality $\exp(x) \leq 1 + x + x^2$ for $x \leq 1$ to get an upper bound on $\log(Z)$. We note that $\gamma\nabla_{t,i} \leq 1$ since $\gamma \leq \frac{1}{G_\infty}$ for large enough T. Now we have

$$\begin{aligned}
\log Z &= \log \sum_i w_{t,i} \exp(-\gamma \nabla_{t,i}) \\
&\leq \log \sum_i w_{t,i}(1 - \gamma \nabla_{t,i} + \gamma^2 \nabla_{t,i}^2) \\
&= \log(1 - \gamma w_t \nabla_t + \gamma^2 \sum_i w_{t,i} \nabla_{t,i}^2) \\
&\leq \log(1 - \gamma w_t \nabla_t + \gamma G_\infty^2) \\
&\leq -\gamma w_t \nabla_t + \gamma^2 G_\infty^2
\end{aligned}$$

In the above, we use the known inequality $\log(1 + x) \leq x$. Combining the above bound on $\log Z$ and the bound on the KL difference, we have:

$$KL(w^*||w_t) - KL(w^*||w_{t+1}) \geq -\gamma w^* \nabla_t + \gamma w_t \nabla_t - \gamma^2 G_\infty^2$$

and so

$$\nabla_t(w_t - w^*) \leq \frac{1}{\gamma}(KL(w^*||w_t) - KL(w^*||w_{t+1})) + \gamma G_\infty^2 \ .$$

Summing the above, we now have:

$$\sum_{t=1}^{T} \nabla_t(w_t - w^*) \leq \frac{1}{\gamma}(KL(w^*||w_1) - KL(w^*||w_{T+1})) + \gamma G_\infty^2 T \ .$$

For the uniform distribution we have $KL(w^*||w_1) \leq \log d$ and generally we also have $0 \leq KL(u||v)$ for any $u, v \in D$. Finally, this leads to

$$\sum_{t=1}^{T} \nabla_t(w_t - w^*) \leq \frac{1}{\gamma} KL(w^*||w_1) + \gamma G_\infty^2 T \ .$$

To get the needed bound, we now just substitute $\gamma = \frac{1}{G_\infty}\sqrt{\frac{\log d}{T}}$.

□

Although EG based algorithms minimize a regret where c_t depends on time, the algorithm can also be applied to the setting when we want to minimize a single function. In this case, we have that

$$R_T(EG) = \sum_{t=1}^{T} c(w_t) - T c(w^*) \ ,$$

and, given the above bound, we then have that

$$\frac{1}{T}\sum_{t=1}^{T}(c(w_t) - c(w^*)) \le \frac{2G_\infty\sqrt{\log d}}{\sqrt{T}} .$$

By the convexity of c we have that

$$c(\frac{1}{T}\sum_{t=1}^{T} w_t) \le \frac{1}{T}\sum_{t=1}^{T} c(w_t)$$

so:

$$c(\frac{1}{T}\sum_{t=1}^{T} w_t) - c(w^*) \le \frac{2G_\infty\sqrt{\log d}}{\sqrt{T}} .$$

Hence, as an optimization procedure, it is sufficient to run this algorithm $O(\frac{1}{\epsilon^2})$ steps to get an ϵ near optimal solution given by $\overline{w}_s = \frac{1}{s}\sum_{t=1}^{s} w_t$.

In summarizing the above, we note that when optimizing a function c in rounds we need not use the averaged solution $\overline{w}_s = \frac{1}{s}\sum_{t=1}^{s} w_t$, but could also use $\hat{w}_s = \arg\max_{t=1}^{s} c(w_t)$. The last realization implies that if we are truly optimizing at each round so that $c(w_t) > c(w_{t+1})$ we need not adopt the more conservative averaged solution. For our experiment with I2CR-2, we tried both methods but the averaged solution was not needed since we were making progress at each step. Moreover, to generalize the above results a bit further, we note that if we have a function $c(w_1, \ldots, w_K)$ where each $w_i \in \mathbb{R}^{d_i}$ is in a simplex, the same proof will still cary through except that the constant would now depend on K. For our purposes, this will be the case with IBM Model 2's relaxation, since we can order the lexical and distortion variables appropriately (for example, we have have $w_i = t(f|e_i)$ for $e_i \in E$ and $f \in D(e_i)$ and similarly for the d terms). As a caveat, the fact that our log-likelihood is concave changes nothing except some signs, and having the log-likelihood depend on multiple simplexes only affects our choice of learning rate. Finally, the above bounds, although theoretically justified, are in practice overly pessimistic: the choice of learning rate for EG is a key, and for our application we cross-validate our algorithm's learning rate on a small dataset to make a sensible choice.

3.3 A Stochastic Exponentiated-Gradient Algorithm for Optimization

We now describe an algorithm for optimizing the I2CR-2 problem in Figure 3.3, and note that this algorithm can be applied with slight modification to I2CR-1 as well. The algorithm is closely related to stochastic gradient ascent, but with two modifications:

- First, as mentioned previously, because the $t(f|e)$ and $d(i|j)$ parameters have simplex constraints (see Figure 2.1), we use *exponentiated gradient* (EG) updates. The main benefit of this algorithm as opposed to classical Gradient Descent is that the projection step is a *recipe*: we do not have to solve an l_2 problem.

- Second, the objective function in the I2CR-2 problem is convex, but is not differentiable (the gradient may not exist at all points). For this reason we use *subgradients* in the place of gradients. In spite of the non-differentiability of the objective function, subgradient methods still have strong convergence guarantees when combined with EG updates (e.g., the convergence proofs in [3] go through with minor modifications; see also [4]).

To derive the updates, recall that we are maximizing the following objective function:

$$\begin{aligned}
& h(t,d) \\
= \quad & \frac{1}{2|\mathcal{T}|}\sum_{k\in\mathcal{T}}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}\min\left\{t(f_j^{(k)}|e_i^{(k)}),d(i|j)\right\} \\
+ \quad & \frac{1}{2|\mathcal{T}|}\sum_{k\in\mathcal{T}}\sum_{j=1}^{m_k}\log'\sum_{i=0}^{l_k}\frac{t(f_j^{(k)}|e_i^{(k)})}{(L+1)}\ . \qquad (3.13)
\end{aligned}$$

Here we use $\mathcal{T}$ to denote the set $\{1\ldots n\}$; we will see shortly why this notation is convenient. We use t and d to refer to the full set of t and d parameters respectively; $h(t,d)$ is the function to be maximized. Recall that $\log'(z) = \log(z+\lambda)$ where λ is a small positive parameter.

Given a concave function $f(x)$ where $x\in\mathbb{R}^d$, a subgradient of $f(x)$ at x is any vector $g(x)\in\mathbb{R}^d$ such that for any $y\in\mathbb{R}^d$,

$$f(y)\leq f(x)+g(x)\cdot(y-x)\ ,$$

where $u\cdot v$ is the inner product between vectors u and v. Subgradients are similar to gradients for differentiable concave functions, in that gradients satisfy the above property. Subgradients can be used in the place of gradients in many optimization algorithms (see for example [4]).

The subgradients for the objective function made up of Eq. 3.13 take a simple form. First, define

$$\begin{aligned} R(j,k) &= \lambda + \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) \ , \\ Q(j,k) &= \lambda + \sum_{i=0}^{l_k} \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\} \ , \end{aligned}$$

and

$$I(i,j,k) = \begin{cases} 1 & \text{if } t(f_j^{(k)}|e_i^{(k)}) \leq d(i|j) \\ 0 & \text{otherwise .} \end{cases}$$

Then the subgradients[2] are

$$\nabla t(f|e) = \frac{1}{2|\mathcal{T}|} \sum_{\substack{i,j,k: \\ f_j^{(k)}=f \\ e_i^{(k)}=e}} \left(\frac{1}{R(j,k)} + \frac{I(i,j,k)}{Q(j,k)} \right)$$

and

$$\nabla d(i|j) = \frac{1}{2|\mathcal{T}|} \sum_{k: i \leq l_k, j \leq m_k} \frac{1 - I(i,j,k)}{Q(j,k)} \ .$$

Exponentiated-gradient updates then take the following form:

$$t(f|e) \leftarrow \frac{t(f|e) \times \exp\{\gamma \times \nabla t(f|e)\}}{\sum_{f'} t(f'|e) \times \exp\{\gamma \times \nabla t(f'|e)\}} \tag{3.14}$$

and

$$d(i|j) \leftarrow \frac{d(i|j) \times \exp\{\gamma \times \nabla d(i|j)\}}{\sum_{i'} d(i'|j) \times \exp\{\gamma \times \nabla d(i'|j)\}} \ , \tag{3.15}$$

where $\gamma > 0$ is a constant step size in the algorithm. Again, note that the EG updates make use of subgradients, but maintain the simplex constraints on the t and d variables.

The method just described is a *batch* gradient method, where the entire training set $\mathcal{T} = \{1 \ldots n\}$ is used to derive the subgradients before the updates in Eqs. 3.14 and 3.15 are made. Many results in Machine Learning and NLP have shown that *stochastic gradient methods*, where a subset of the training examples is used before each gradient-based update, can converge much more quickly than batch gradient methods. In our notation, this simply involves replacing $\mathcal{T}$ by some subset $\mathcal{T}'$ of the training examples in the above definitions, where $|\mathcal{T}'|$ is typically much smaller than $|\mathcal{T}|$.

[2]We set $\nabla t(f|e)$ and $\nabla d(i|j)$ as the subgradients for the objective function in Eq. 3.8 with respect to $t(f|e)$ and $d(i|j)$ respectively.

Figure 3.3 shows our final algorithm, a stochastic version of the exponentiated-gradient method. The method takes S passes over the data. For each pass, it randomly partitions the training set into mini-batches $\mathcal{T}_1 \ldots \mathcal{T}_K$ of size B, where B is an integer specifying the size of each mini-batch (in our experiments we used $B = 125$ or $B = 250$). The algorithm then performs EG updates using each mini-batch $\mathcal{T}_1 \ldots \mathcal{T}_K$ in turn. As can be seen in Table 3.3, our experiments show that the algorithm makes very significant progress in the first pass over the data, and takes very few iterations to converge to a good solution even though we initialized with uniform parameter values.

3.4 I2CR-2 Experiments

In this section we describe experiments using the I2CR-2 optimization problem combined with the stochastic EG algorithm for parameter estimation. We first describe the data sets we use, and then describe experiments with the method, comparing our approach to results from IBM Model 2. We compare the various algorithms in terms of their accuracy in recovering alignments, using metrics such as F-Measure and AER.

3.4.1 Data Sets

We use data from the bilingual word alignment workshop held at HLT-NAACL 2003 [26]. As a first dataset, we use the Canadian Hansards bilingual corpus, with 247,878 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data (note that we use a randomly chosen subset of the original training set of 1.1 million sentences, similar to the setting used in [27]). The development and test data have been manually aligned at the word level, annotating alignments between source and target words in the corpus as either "sure" (S) or "possible" (P) alignments, as described in [28].

As a second data set, we used the Romanian-English data from the HLT-NAACL 2003 workshop. This consisted of a training set of 48,706 Romanian-English sentence-pairs, a development set of 17 sentence pairs, and a test set of 248 sentence pairs.

```
1: Input: Define $E$, $F$, $L$, $M$, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2.3. An integer $B$
   specifying the batch size. An integer $S$ specifying the number of passes over the data. A step size $\gamma > 0$. A
   parameter $\lambda > 0$ used in the definition of $\log'$.
2: Parameters:
   • A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
   • A parameter $d(i|j)$ for each $i \in [L]_0, j \in [M]$.
3: Definitions:
   $R(j,k) = \lambda + \sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)})$
   $Q(j,k) = \lambda + \sum_{i=0}^{l_k} \min\{t(f_j^{(k)}|e_i^{(k)}), d(i|j)\}$
4: Initialization:
   • $\forall e \in E,\ f \in D(e),\ t(f|e) = 1/|D(e)|$
   • $\forall j \in [M],\ i \in [L]_0,\ d(i|j) = 1/(L+1)$
5: Algorithm:
6: for all $s = 1$ to $S$ do
7:     Randomly partition $[n]$ into subsets $\mathcal{T}_1 \ldots \mathcal{T}_K$ of size $B$ where $K = n/B$.
8:     for all $b = 1$ to $K$ do
9:         $\forall e \in E,\ f \in D(e),\ \alpha(e,f) = 0$
10:        $\forall j \in [M],\ i \in [L]_0,\ \beta(i,j) = 0$
11:        for all $k \in \mathcal{T}_b$ do
12:            for all $j = 1$ to $m_k$ do
13:                for all $i = 0$ to $l_k$ do
14:                    $\alpha(e_i^{(k)}, f_j^{(k)}) \mathrel{+}= 1/(2R(j,k))$
15:                    if $t(f_j^{(k)}|e_i^{(k)}) \leq d(i|j)$ then
16:                        $\alpha(e_i^{(k)}, f_j^{(k)}) \mathrel{+}= 1/(2Q(j,k))$
17:                    else
18:                        $\beta(i,j) \mathrel{+}= 1/(2Q(j,k))$
19:        $\forall e, f,\ t(f|e) = t(f|e) \exp(\gamma \times \alpha(e,f)/B)$
20:        $\forall i, j, d(i|j) = d(i|j) \exp(\gamma \times \beta(i,j)/B)$
21:        Renormalize $t$ and $d$ parameters to satisfy $\sum_f t(f|e) = 1$ and $\sum_i d(i|j) = 1$.
22: Output: $t$ and $d$ parameters.
```

Figure 3.3: The stochastic exponentiated-gradient algorithm for optimization of I2CR-2.

3.4.2 Methodology

For each of the models—IBM Model 1, IBM Model 2, and I2CR-2—we follow convention in applying the following methodology: first, we estimate the t and d parameters using models in both source-target and target-source directions; second, we find the most likely alignment for each development or test data sentence in each direction; third, we take the intersection of the two alignments as the final output from the model.

For the EG algorithm we use a batch size $B = 125$ and step size $\gamma = 0.5$ on the Hansards data, and $B = 250$ and $\gamma = 0.5$ for the Romanian-English data.

We report the performance of the models in terms of *Precision*, *Recall*, *AER*, and *F-Measure* as defined by [28]. If A is the set of alignments produced by an algorithm, S is the set of sure alignments as annotated in test data, and P is the set of possible alignments, then these quantities are defined as

$$\text{Recall} = \frac{|A \cap S|}{|S|} \ ,$$

$$\text{Precision} = \frac{|A \cap S|}{|A|} \ ,$$

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \ ,$$

$$\text{F-Measure} = \frac{1}{\frac{.5}{\text{Recall}} + \frac{.5}{\text{Precision}}} \ .$$

Note that we report results in both AER and F-Measure; however there is evidence [18] that F-Measure is better correlated with translation quality when the alignments are used in a full system.

In training IBM Model 1 we follow [27] in running EM for 15 iterations. In training IBM Model 2 we first train IBM Model 1 for 15 iterations to initialize the t parameters, then train IBM Model 2 for a further 10 iterations. For the EG algorithm, we use 10 iterations over the training data for the Hansards data, and 15 iterations on the Romanian-English data (on the latter dataset results on the trial data showed that the method took slightly longer to converge). We report F-measure and AER results for each of the iterations under the IBM Model 2 and I2CR-2 models. See Table 3.1 for the results on the Hansards data, and Table 3.2 for the results on the English-Romanian dataset. It can be seen that both I2CR-2 and IBM Model 2 converge to a fairly stable result after 2-3 iterations. The two models give very similar levels of performance, for example after 10 iterations on the Hansard data IBM Model 2 gives 14.22 AER and 0.7516 F-Measure versus 14.60 AER and 0.7506 F-Measure for I2CR-2.

On the right, Table 3.3 shows the values of the objective function at each iteration when using the EG algorithm to optimize the I2CR-2 objective. The method makes a large amount of progress on

Iteration	IBM2	I2CR-2	IBM2	I2CR-2
	AER	AER	F-Measure	F-Measure
		Test Set Statistics		
1	0.1491	0.1556	0.7530	0.7369
2	0.1477	0.1489	0.7519	0.7456
3	0.1451	0.1476	0.7527	0.7467
4	0.1426	0.1488	0.7536	0.7449
5	0.1422	0.1495	0.7535	0.7472
6	0.1431	0.1476	0.7511	0.7478
7	0.1434	0.1506	0.7506	0.7456
8	0.1437	0.1495	0.7501	0.7470
9	0.1434	0.1494	0.7501	0.7468
10	0.1422	0.1460	0.7516	0.7506
		Development Set Statistics		
1	0.1871	0.1971	0.6823	0.6676
2	0.1896	0.1870	0.6758	0.6827
3	0.1964	0.1860	0.6648	0.6739
4	0.1912	0.1835	0.6713	0.6775
5	0.1884	0.1813	0.6740	0.6773
6	0.1836	0.1851	0.6767	0.6811
7	0.1831	0.1806	0.6749	0.6765
8	0.1842	0.1843	0.6739	0.6775
9	0.1864	0.1928	0.6694	0.6640
10	0.1845	0.1829	0.6703	0.6721

Table 3.1: Results on the Hansards data for IBM Model 2 and the I2CR-2 method.

Iteration	IBM2 AER	I2CR-2 AER	IBM2 F-Measure	I2CR-2 F-Measure
Test Set Statistics				
1	0.4041	0.5354	0.5959	0.4646
2	0.4010	0.4764	0.5990	0.5256
3	0.4020	0.4543	0.5980	0.5457
4	0.4012	0.4384	0.5988	0.5617
5	0.4003	0.4277	0.5997	0.5723
6	0.3990	0.4266	0.6010	0.5834
7	0.4000	0.4162	0.6000	0.5838
8	0.4023	0.4114	0.5977	0.5886
9	0.4022	0.4081	0.5978	0.5919
10	0.4027	0.4043	0.5973	0.5957
11	0.4031	0.4040	0.5969	0.5960
12	0.4042	0.4027	0.5958	0.5973
13	0.4043	0.4021	0.5957	0.5979
14	0.4062	0.4007	0.5938	0.5993
15	0.4057	0.4014	0.5943	0.5986
Development Set Statistics				
1	0.4074	0.5841	0.5926	0.4159
2	0.3911	0.4938	0.6089	0.5062
3	0.3888	0.4673	0.6112	0.5327
4	0.3904	0.4596	0.6096	0.5404
5	0.3881	0.4463	0.6119	0.5537
6	0.3904	0.4306	0.6096	0.5694
7	0.3936	0.4175	0.6094	0.5826
8	0.3897	0.4060	0.6103	0.5940
9	0.3961	0.4014	0.6039	0.5986
10	0.3970	0.4072	0.6030	0.5928
11	0.4018	0.3956	0.5982	0.6044
12	0.4035	0.3931	0.5965	0.6069
13	0.4035	0.3862	0.5965	0.6138
14	0.4014	0.3908	0.5986	0.6092
15	0.4063	0.3858	0.5937	0.6142

Table 3.2: Results on the English-Romanian data for IBM Model 2 and the I2CR-2 method.

Iteration	EF Objective	FE Objective
0	-99.6053	-79.5566
1	-32.4528	-27.4925
2	-31.1641	-26.2620
3	-30.6311	-25.7093
4	-30.3367	-25.3714
5	-30.1428	-25.1456
6	-30.0000	-24.9920
7	-29.8736	-24.8605
8	-29.8093	-24.7551
9	-29.7326	-24.6840
10	-29.6771	-24.6099

Table 3.3: Objective values for the EG algorithm optimization of I2CR-2 at each iteration. "EF Objective" corresponds to training a model with $t(f|e)$ parameters, "FE Objective" corresponds to the reverse direction, with $t(e|f)$ parameters. Iteration 0 corresponds to the objective value under the initial, uniform parameter values.

the first iteration and then continues to improve. Finally, we note that the memory requirements for I2CR-2 and IBM2 are about the same, but that the time for one iteration of I2CR-2 on the Hansards data is approximately one hour, while the time for one iteration of IBM2 was approximately 10 minutes.

3.5 The Viterbi Alignment for I2CR-2

Alignment models have been compared using methods other than Viterbi comparisons; for example, in the previous section we use IBM Model 2's optimal rule given by (see below) Eq. 3.17 to compare models while Liang et al. (2006) use posterior decoding. Here, we derive and use I2CR-2's Viterbi alignment. To get the Viterbi alignment of a sentence pair (e, f) using I2CR-2 we need to find $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_m)$ which yields the highest probability $p(f, \hat{a}|e)$. Referring to the I2CR-2 objective, this corresponds to finding $\hat{a}$ that maximizes

$$\frac{\log \prod_{j=1}^{m} t(f_j|e_{\hat{a}_j})}{2} + \frac{\log \prod_{j=1}^{m} \min\{t(f_j|e_{\hat{a}_j}), d(\hat{a}_j|j)\}}{2} .$$

Putting the above terms together and using the monotonicity of the logarithm, the above reduces to finding the vector $\hat{a}$ which maximizes

$$\prod_{j=1}^{m} t(f_j|e_{a_j}) \min\{t(f_j|e_{a_j}), d(a_j|j)\}.$$

As with IBM Models 1 and 2, we can find the optimal alignment vector $\hat{a}$ by splitting the maximization over the components of $\hat{a}$ and focusing on finding $\hat{a}_j$ given by

$$\hat{a}_j = \text{argmax}_{i=0}^{l}(t(f_j|e_i) \min\{t(f_j|e_i), d(i|j)\}) . \tag{3.16}$$

In previous experiments we presented for I2CR-2, we compared I2CR-2 and IBM Model 2 using the standard alignment formula derived in a similar fashion from IBM Model 2:

$$\hat{a}_j = \text{argmax}_{i=0}^{l}(t(f_j|e_i)d(i|j)) . \tag{3.17}$$

Since the parameters t and d above are non-negative, Eq. 3.16 can be rewritten as

$$\text{argmax}_a(\min\{t^2(f_j|e_a), t(f_j|e_a)d(a|j)\}) .$$

From the above, we see that the candidate alignment a of for a position j reduces to the square score produced by IBM Model 1 if the lexical probability is smaller than the distortion while it is that of IBM Model 2 if the distortion term $d(a|j)$ is smaller than the lexical probability.

3.6 Experiments

In this section we describe further experiments and applications using the I2CR-2 optimization problem combined with the stochastic EG algorithm [33] for parameter estimation. The experiments conducted here use a similar setup to those in (Simion et al., 2014) [33] as presented above.

Training	2^{10}	$1^5 2^{10}$	$\text{EG}^1_{125} 2^{10}$	$\text{EG}^1_{1250} 2^{10}$
Iteration		Objective		
0	-224.0919	-144.2978	-91.2418	-101.2250
1	-110.6285	-85.6757	-83.3255	-85.5847
2	-91.7091	-82.5312	-81.3845	-82.1499
3	-84.8166	-81.3380	-80.6120	-80.9610
4	-82.0957	-80.7305	-80.2319	-80.4041
5	-80.9103	-80.3798	-80.0173	-80-1009
6	-80.3620	-80.1585	-79.8830	-79.9196
7	-80.0858	-80.0080	-79.7911	-79.8048
8	-79.9294	-79.9015	-79.7247	-79.7284
9	-79.8319	-79.8240	-79.6764	-79.6751
10	-79.7670	-79.7659	-79.6403	-79.6354

Table 3.4: Objective results for the English $\rightarrow$ French IBM Model 2 seeded with either uniform parameters, IBM Model 1 ran for 5 EM iterations, or I2CR-2 ran for 1 iteration with either $B =$ 125 or 1250. Iteration 0 denotes the starting IBM 2 objective depending on the initialization.

3.6.1 Initialization and Timing Experiments

We first report the summary statistics on the test set using a model trained only in the English-French direction. In these experiments we seeded IBM Model 2's parameters either with those of IBM Model 1 run for 5, 10 or 15 EM iterations or I2CR-2 run for 1 iteration of EG with a batch size of either $B = 125$ or 1250. For uniform comparison, all of our implementations were written in C++ using STL/Boost containers.

There are several takeaways from our experiments, which are presented in Table 3.5. We first note that with $B = 1250$ we get higher F-Measure and lower AER even though we use less training time: 5 iterations of IBM Model 1 EM training takes about 3.3 minutes, which is about the time it takes for 1 iteration of EG with a batch size of 125 (4.1 minutes); on the other hand, using $B = 1250$ takes EG 1.7 minutes and produces the best results across almost all iterations. Additionally, we note that the initial solution given to IBM Model 2 by running I2CR-2 for 1 iteration with $B = 1250$ is fairly strong and allows for further progress: IBM2 EM training improves upon this solution during the first few iterations. We also note that this behavior is global: no IBM 1 initialization scheme produced subsequent solutions for IBM 2 with as low in AER or high in F-Measure. Finally, comparing Table 3.4 which lists objective values with Table 3.5 which lists alignment statistics, we see that although the objective progression is similar throughout, the alignment quality is different.

To complement the above, we also ran intersection experiments. Seeding IBM Model 2 by Model 1 and intersecting the alignments produced by the English-French and French-English models gave both AER and F-Measure which were better than those that we obtained by any seeding of IBM Model 2 with I2CR-2. However, there are still reasons why I2CR-2 would be useful in this context. In particular, we note that I2CR-2 takes roughly half the time to progress to a better solution than IBM Model 1 run for 5 EM iterations. Second, a possible remedy to the above loss in marginal improvement when taking intersections would be to use a more refined method for obtaining the joint alignment of the English-French and French-English models, such as "grow-diagonal" [28].

3.6.2 Viterbi Comparisons

For the decoding experiments, we used IBM Model 1 as a seed to Model 2. To train IBM Model 1, we follow [27] and [28] in running EM for 5, 10 or 15 iterations. For the EG algorithm, we initialize all parameters uniformly and use 10 iterations of EG with a batch size of 125. Given the lack of

Training	2^{10}	$1^5 2^{10}$	$1^{10} 2^{10}$	$1^{15} 2^{10}$	$\text{EG}^1_{125} 2^{10}$	$\text{EG}^1_{1250} 2^{10}$
Iteration			AER			
0	0.8713	0.3175	0.3177	0.3160	**0.2329**	0.2662
1	0.4491	0.2547	0.2507	0.2475	0.2351	**0.2259**
2	0.2938	0.2428	0.2399	0.2378	0.2321	**0.2180**
3	0.2593	0.2351	0.2338	0.2341	0.2309	**0.2176**
4	0.2464	0.2298	0.2305	0.2310	0.2283	**0.2168**
5	0.2383	0.2293	0.2299	0.2290	0.2268	**0.2188**
6	0.2350	0.2273	0.2285	0.2289	0.2274	**0.2205**
7	0.2320	0.2271	0.2265	0.2286	0.2274	**0.2213**
8	0.2393	0.2261	0.2251	0.2276	0.2278	**0.2223**
9	0.2293	0.2253	0.2246	0.2258	0.2284	**0.2217**
10	0.2288	0.2248	0.2249	0.2246	0.2275	**0.2223**
Iteration			F-Measure			
0	0.0427	0.5500	0.5468	0.5471	**0.6072**	0.5977
1	0.4088	0.5846	0.5876	0.5914	0.6005	**0.6220**
2	0.5480	0.5892	0.5916	0.5938	0.5981	**0.6215**
3	0.5750	0.5920	0.5938	0.5947	0.5960	**0.6165**
4	0.5814	0.5934	0.5839	0.5952	0.5955	**0.6129**
5	0.5860	0.5930	0.5933	0.5947	0.5945	**0.6080**
6	0.5873	0.5939	0.5936	0.5940	0.5924	**0.6051**
7	0.5884	0.5931	0.5955	0.5941	0.5913	**0.6024**
8	0.5899	0.5932	0.5961	0.5942	0.5906	**0.6000**
9	0.5899	0.5933	0.5961	0.5958	0.5906	**0.5996**
10	0.5897	0.5936	0.5954	0.5966	0.5910	**0.5986**

Table 3.5: Results on the Hansards data for English $\rightarrow$ French IBM Model 2 seeded using different methods. The first four columns are for a model seeded with IBM Model 1 ran for 0, 5, 10 or 15 EM iterations. The fifth and sixth columns show results when we seed with I2CR-2 ran for 1 iteration either with $B = 125$ or 1250. Iteration 0 denotes the starting statistics.

Training	$1^5 2^{10}$	$1^{10} 2^{10}$	$1^{15} 2^{10}$	EG_{125}^{10}	EG_{125}^{10}
Viterbi Rule	$t \times d$	$t \times d$	$t \times d$	$t \times d$	$t \times \min\{t \times d\}$
Iteration			AER		
0	**0.2141**	0.2159	0.2146	0.9273	0.9273
1	0.1609	0.1566	**0.1513**	0.1530	0.1551
2	0.1531	0.1507	0.1493	0.1479	**0.1463**
3	0.1477	0.1471	0.1470	0.1473	**0.1465**
4	0.1458	**0.1444**	0.1449	0.1510	0.1482
5	0.1455	0.1438	**0.1435**	0.1501	0.1482
6	0.1436	0.1444	**0.1429**	0.1495	0.1481
7	0.1436	**0.1426**	0.1435	0.1494	0.1468
8	0.1449	**0.1427**	0.1437	0.1508	0.1489
9	0.1454	**0.1426**	0.1430	0.1509	0.1481
10	0.1451	0.1430	**0.1423**	0.1530	0.1484
Iteration			F-Measure		
0	**0.7043**	0.7012	0.7021	0.0482	0.0482
1	0.7424	0.7477	0.7534	0.7395	**0.7507**
2	0.7468	0.7499	0.7514	0.7448	**0.7583**
3	0.7489	0.7514	0.7520	0.7455	**0.7585**
4	0.7501	0.7520	0.7516	0.7418	**0.7560**
5	0.7495	0.7513	0.7522	0.7444	**0.7567**
6	0.7501	0.7501	0.7517	0.7452	**0.7574**
7	0.7493	0.7517	0.7507	0.7452	**0.7580**
8	0.7480	0.7520	0.7504	0.7452	**0.7563**
9	0.7473	0.7511	0.7513	0.7450	**0.7590**
10	0.7474	0.7505	0.7520	0.7430	**0.7568**

Table 3.6: Intersected results on the English-French data for IBM Model 2 and I2CR-2 using either IBM Model 1 trained to 5, 10, or 15 EM iterations to seed IBM2 and using either the IBM2 or I2CR-2 Viterbi formula for I2CR-2.

development data for the alignment data sets, for both IBM Model 2 and the I2CR-2 method, we report test set F-Measure and AER results for each of the 10 iterations, rather than picking the results from a single iteration.

In Table 3.6 we report F-Measure and AER results for each of the iterations under IBM Model 2 and I2CR-2 models using either the Model 2 Viterbi rule of Eq. 3.17 or I2CR-2's Viterbi rule in Eq. **??**. We note that unlike in the previous experiments presented in [33], we are directly testing the quality of the alignments produced by I2CR-2 and IBM Model 2 since we are getting the Viterbi alignment for each model (for completeness, we also have included in the fourth column the Viterbi alignments we get by using the IBM Model 2 Viterbi formula with the I2CR-2 parameters as Simion et al. (2013) had done previously). For these experiments we report intersection statistics. Under its proper decoding formula, I2CR-2 model yields a higher F-Measure than any setting of IBM Model 2. Since AER and BLEU correlation is arguably known to be weak while F-Measure is at times strongly related with BLEU [14], the above results favor the convex model.

We close this section by pointing out that the main difference between the IBM Model 2 Viterbi rule of Eq. 3.17 and the I2CR-2 Viterbi rule in Eq. 3.16 is that the Eq. 3.16 yield fewer alignments when doing intersection training. Even though there are fewer alignments produced, the quality in terms of F-Measure is better.

3.7 Conclusions and Future Work

In this section we have introduced the first convex model for unsupervised learning of alignments in statistical machine translation with performance comparable to the commonly-used IBM Model 2. We believe that introducing convexity without sacrificing performance will open the door to further improvements in this area. Moreover, we have also explored some of the further details of the I2CR2-2 model and shown that it may potentially be used as a new initialization technique for IBM Model 2 or as a model in its own right, especially if the F-Measure is the target metric. With regard to the current model, other possible topics of interest include performing efficient sensitivity analysis on the I2CR-2 model, analyzing the balance between the IBM Model 1 and I2CR-1 [33] components of the I2CR-2 objective, studying I2CR-2's intersection training performance using methods such as "grow diagonal" or "agreement" [25], and integrating it into the GIZA++ open source library so we can see how much it affects the downstream system.

Chapter 4

A Family of Latent Variable Convex Relaxations for IBM Model 2

4.1 A Class of Concave Functions based on the Generalized Weighted Mean

In the previous chapter, model I2CR-2 was studied and, at a high level, the key component in its construction was to replace the non-concave function $f(x) = \prod_{i=1}^{n} x_i$ by the concave function $h(x) = \min_{i=1}^{n} x_i$. This is only one possible convexification; we now explore a much larger set of ways to convexify a product. While some of this work is found in the literature (e.g. [7; 10]), its application in the context we are interested in (and the methods we developed in Section 5), is new. For clarity, we present all the element that we need below.

Definition 8. *Let $(\alpha_1, \dots, \alpha_n) \in (0,1)^n$ be such that $\sum_{i=1}^{n} \alpha_i = 1$. For $p \neq 0$ denote $f_p : \mathbb{R}^n_{++} \to \mathbb{R}_+$ given by*

$$f_p(x_1, \dots, x_n) = \left(\sum_{i=1}^{n} \alpha_i x_i^p \right)^{1/p} \tag{4.1}$$

as the generalized weighted mean function. For $p = 0$ denote $f_0 : \mathbb{R}^n_{++} \to \mathbb{R}_+$ given by

$$f_0(x_1, \dots, x_n) = \prod_{i=1}^{n} x_i^{\alpha_i} \tag{4.2}$$

as the generalized weighted geometric mean function.

Although the above definition restricts the domain to $\mathbb{R}^n_{++}$, we extend the domain of f_p to $\mathbb{R}^n$ by setting $f_p(x)$ to $-\infty$ for any $x \notin \mathbb{R}^n_{++}$. With this definition, f_p is defined everywhere and is a *concave* function [7]. The results we need on the generalized weighted mean are detailed next along with some new material that serves as supplement. Theorems 2-3 and Lemma 1 are implicit in several sources in the literature ([7; 44; 10]).

Theorem 2. *If $p \leq 1$ then any f_p within the class of functions in Definition 8 is concave.*

Proof. Clearly f_p is linear when $p = 1$ so in this case there is nothing to prove. We address the concavity of f_p for $p < 1$, $p \neq 0$ first. By the rules of differentiation, we have that

$$\frac{f_p(x)}{\partial x_i} = \alpha_i \left(\frac{f_p(x)}{x_i}\right)^{1-p} .$$

Using the formula for the gradient above we now have that the Hessian matrix, H_p, of f_p is relatively easy to compute. Specifically, we have that $\frac{\partial^2 f_p(x)}{\partial x_i^2}$ is equal to

$$\frac{\alpha_i^2(1-p)}{f_p(x)}\left(\frac{f_p(x)^2}{x_i^2}\right)^{1-p} - \frac{\alpha_i(1-p)}{x_i}\left(\frac{f_p(x)}{x_i}\right)^{1-p}$$

while (if $i \neq j$) $\frac{\partial^2 f_p(x)}{\partial x_i \partial x_j}$ is

$$\frac{\alpha_i\alpha_j(1-p)}{f_p(x)}\left(\frac{f_p(x)^2}{x_i x_j}\right)^{1-p} .$$

To conclude that f_p is concave it suffices to show that the Hessian is negative definite so that $\forall\, z$ we have $z^T H_p z \leq 0$. Simplifying the expression that results when evaluating $z^T H_p z$ we have that it is equal to

$$\frac{1-p}{f_p(x)}\left(\left(\sum_{i=1}^{n}\frac{\alpha_i z_i f_p(x)^{1-p}}{x_i^{1-p}}\right)^2 - \sum_{i=1}^{n}\frac{\alpha_i z_i^2 f_p(x)^{2-p}}{x_i^{2-p}}\right) .$$

To show that the above is negative, we first note that we can ignore the $\frac{1-p}{f_p(x)}$ term since it is positive. Concavity now following by applying the Cauchy-Schwatz inequality $v^T w \leq ||v||_2||w||_2$ to the vectors v and w with $v_i = z_i\sqrt{\alpha_i}(\frac{f_p(x)}{x_i})^{1-p/2}$ and $w_i = \frac{\sqrt{\alpha_i} f_p(x)^{-p/2}}{x_i^{-p/2}}$ and noting that $||w||_2 = 1$.

We conclude the proof by considering the case $p = 0$. In this case, we note that the Hessian matrix H_0 is given by

$$\frac{\partial^2 f_0(x)}{\partial x_i \partial x_j} = \begin{cases} \frac{\alpha_i(\alpha_i - 1) f_0(x)}{x_i^2} & : \text{if } i = j \\ \frac{\alpha_i \alpha_j f_0(x)}{x_i x_j} & : \text{if } i \neq j \end{cases}$$

Using the form of H_0 above we then have

$$z^T H_0 z = (\sum_{i=1}^{n} \frac{\alpha_i z_i}{x_i})^2 - \sum_{i=1}^{n} \frac{\alpha_i z_i^2}{x_i^2} \leq 0 \ ,$$

via the Cauchy-Schwartz inequality applied to the vectors $w = (\sqrt{\alpha_1}, \ldots, \sqrt{\alpha_n})$ and $v = (\frac{\sqrt{\alpha_1} z_1}{x_1}, \ldots, \frac{\sqrt{\alpha_n} z_n}{x_n})$.

□

Using Theorem 2 and extending f_p to $\mathbb{R}^n$, the generalized mean function thus gives us a family of concave functions defined everywhere. Interestingly, we note that the extremes

$$\lim_{p \to 0} f_p(x) = \prod_{i=1}^{n} x_i^{\alpha_i} = f_0(x)$$

and

$$\lim_{p \to -\infty} f_p(x) = \min\{x_1, \ldots, x_n\} = f_{-\infty}(x) \ ,$$

are both concave and belong to this family.

Lemma 1. *Let $f_p(x)$ be defined as in Definition 8. We have*

$$\lim_{p \to 0} f_p(x) = \prod_{i=1}^{n} x_i^{\alpha_i} \ ,$$

and

$$\lim_{p \to -\infty} f_p(x) = \min\{x_1, \ldots, x_n\} \ .$$

Proof. Letting $p \to 0$ directly we reach an undetermined case of the type 1^∞. Using standard techniques, we have that the sought limit is the same as

$$\exp^{\sum_{i=1}^{n} \alpha_i \lim_{p \to 0} \frac{x_i^p - 1}{p}} \ .$$

For any $y > 0$ we know that $\lim_{p \to 0} \frac{y^p - 1}{p} = \ln(y)$ via L'Hospital's rule. The result now follows using the properties of the logarithm. For the second result, without loss of generality suppose that x_1 is the smallest of $\{x_i\}_{i=1}^n$. We then have that for $p < 0$

$$x_1 \leq f_p(x) \leq \alpha_1^{1/p} x_1 \ .$$

Letting $p \to -\infty$ and using the squeeze theorem, we get the result.

□

Lemma 1 implies that $f_p(x)$ can be identified for any $p \leq 1$ and all x as being concave. Moreover, for $x \in [0,1]^n$, $f_p(x)$ provides a monotonic concave upper envelope for $\prod_{i=1}^n x_i$.

Theorem 3. *Let $f_p(x)$ be defined as in Definition 8. For $x \in [0,1]^n$ the generalized weighted mean function $f_p(x)$ provides a monotonic concave envelope for $\prod_{i=1}^n x_i$. In particular, we have*

$$\prod_{i=1}^n x_i \leq f_p(x) \leq f_q(x)$$

for any $p \leq q \leq 1$.

Proof. We note that the result we present can be derived as a consequence of the generalized-weighted mean inequality. However, we present a more direct proof within the context of the domain we are dealing with. We show that $\forall\ x \in (0,1]^n$

$$\prod_{i=1}^n x_i \leq f_p(x)\ .$$

Consider $p \neq 0$. Using the chain rule, we have that the partial derivative $\frac{\partial f_p(x)}{\partial p}$ is equal to

$$-\frac{1}{p^2} f_p(x)^{1-p} (\sum_{i=1}^n \alpha_i x_i^p \log x_i)\ .$$

From the fact that $x_i \in (0,1]$ we have $\log x_i < 0$. Looking at the form of the partial derivative, it follows that $\frac{\partial f_p(x)}{\partial p} \geq 0$ for any $p \neq 0$. Finally, the case that $f_0(x) \leq f_p(x)$ for $p > 0$ and $f_p(x) \leq f_0(x)$ for $p < 0$ follows by using Lemma 1 and the monotonicity for $p \neq 0$ established above.

□

We next show that $f_{-\infty}(x) = \min_{i=1}^n x_i$ is a special function when used to bound $\prod_{i=1}^n x_i$ above by a positive-valued concave envelope. Specifically, we have that $\min_{i=1}^n x_i$ is the tightest such upper bound, *regardless* of the class of functions we consider.

Theorem 4. *Consider any concave function $h : \mathbb{R}_{++}^n \to \mathbb{R}_+$ such that*

$$\prod_{i=1}^n x_i \leq h(x)$$

for all $x \in [0,1]^n$. Then

$$\min_{i=1}^n x_i \leq h(x)$$

for all $x \in [0,1]^n$.

Proof. The proof is by strong induction on n. Consider first the case $n = 2$. Note that for any point of the type $(x_1, 1)$, $(x_1, 0)$, $(1, x_2)$, or $(0, x_2)$ the result follows easily, so without loss of generality consider $x = (x_1, x_2) \in (0, 1)^2$ with $x_1 \leq x_2$ and suppose by way of contradiction that $h(x_1, x_2) < x_1$ and note that we have

$$x_1 \leq x_2 h\left(\frac{x_1}{x_2}, 1\right)$$

and

$$0 \leq (1 - x_2)h(0, 0)$$

by the positivity of h and the fact that h bounds the product of its arguments. Adding the above and using Jensen's inequality we then have

$$x_1 \leq x_2 h\left(\frac{x_1}{x_2}, 1\right) + (1 - x_2)h(0, 0) \leq h(x_1, x_2) < x_1 \ .$$

The above result yields a contradiction, and we now have that $\min\{x_1, x_2\}$ is the tightest positive-valued upper bound on $x_1 x_2$. Now, assume that the result holds for the case $n = k$ and consider some $x \in [0, 1]^n$ with $n = k + 1$. First note that if any component of x is zero the result is trivial. Now suppose that $1 \leq l \leq n$ components of x are 1. Without loss of generality, we can assume that these components are $x_1, \ldots, x_l$ and note that in this case $x = (1, \ldots, 1, x_{l+1}, \ldots, x_n)$. Then, $h'(x') = h(1, \ldots, 1, x_{l+1}, \ldots, x_n)$ is a concave function in $\mathbb{R}^{n-l}_{++}$ with $x' = (x_{l+1}, \ldots, x_n) \in [0, 1]^{n-l}$. Moreover, h' is satisfies $\prod_{i=l+1}^{n} x_i \leq h'(x')$, so that by the induction hypothesis we have

$$\min_{i=1}^{n} x_i = \min_{i=l+1}^{n} x_i \leq h'(x') = h(x) \ ,$$

as needed. Suppose now by way of contradiction that we have an $x \in (0, 1)^n$ with $h(x) < \min_{i=1}^{n} x_i$. To simplify notation, suppose furthermore without loss of generality that $x_1 \leq x_2 \leq \ldots \leq x_n$. Under this setting let $x' = (\frac{x_1}{x_2}, 1, \ldots, 1)$ and $x'' = (0, 0, \frac{x_3 - x_2}{1 - x_2}, \ldots, \frac{x_n - x_2}{1 - x_2})$ and note that x' and $x'' \in [0, 1]^{k+1}$, $x_1 \leq x_2 h(x')$, $0 \leq (1 - x_2)h(x'')$, and $x = x_2 x' + (1 - x_2)x''$. Applying Jensen's inequality yields

$$\begin{aligned} x_1 &\leq x_2 h(x') + (1 - x_2)h(x'') \\ &\leq h(x) \\ &< x_1 \ , \end{aligned}$$

a contradiction.

□

Although several upper bounds for $\prod_{i=1}^n x_i$ with $x \in [0,1]^n$ are detailed above, we note that bounding $\prod_{i=1}^n x_i$ *below* by a nontrivial positive-valued concave function is not possible, if $n \geq 2$.

Theorem 5. *Let $n \geq 2$ and $h : \mathbb{R}^n_{++} \to \mathbb{R}_+$ be a concave function such that*

$$h(x) \leq \prod_{i=1}^n x_i$$

for all $x \in [0,1]^n$. Then $h(x)$ is identically equal to zero.

Proof. If x has a component which is zero then $h(x) \leq 0$ and hence $h(x) = 0$ since $h(x) \geq 0$. Choosing $\theta \in (0,1)$ and $x \in (0,1]^n$ yields that

$$\theta h(x) + (1-\theta)h(0) \leq h(\theta x) \leq \theta^n \prod_{i=1}^n x_i \ ,$$

and we note that the left hand side above is equal to $\theta h(x)$. Dividing both sides by θ we next have

$$h(x) \leq \theta^{n-1} \prod_{i=1}^n x_i \ .$$

Letting $\theta \to 0$ in the last equation we get $h(x) \leq 0$. Since we also have $h(x) \geq 0$, we now have $h(x) = 0$ for any $x \in [0,1]^n$, as needed.

□

The main takeaway of the above is that positive valued concave envelopes for $\prod_{i=1}^n x_i$ are limited to *upper* envelopes such as those provided by f_p in Definition 8. Indeed, among *all* upper envelopes we can choose, the min is the tightest.

4.2 A Family of Convex IBM Model 2 Alternatives

From our earlier discussion, the first relaxations of IBM Model 2 were called I2CR-1 and I2CR-2. Since the methods presented here are a generalization of the previous results, we use I2CR to denote the general optimization problem class that arises by using a special concave h instead of $x_1 x_2$ in IBM Model 2; see Figure 2. I2CR-3 and I2CR-4 are based on the particular concave

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3.

A positive-valued concave function $h : \mathbb{R}^2_{++} \rightarrow \mathbb{R}_+$ such that

$$x_1 x_2 \leq h(x_1, x_2) \ ,$$

$\forall \ (x_1, x_2) \in [0,1]^2$.

Parameters: Same as IBM Model 2.

Constraints: Same as IBM Model 2.

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} h(t(f_j^{(k)} | e_i^{(k)}), d(i|j)) \tag{4.3}$$

with respect to the $t(f|e)$ and $d(i|j)$ parameters.

Figure 4.1: The I2CR (IBM 2 Convex Relaxation) Problem. For any function h that is concave, the resulting optimization problem is a convex problem. I2CR-1 results from using $h(x_1, x_2) = f_{-\infty}(x_1, x_2) = \min\{x_1, x_2\}$ in the above while I2CR-3 arises from using $h(x_1, x_2) = f_0(x_1, x_2) = x_1^{\beta} x_2^{1-\beta}$ with $\beta \in [0, 1]$.

function $f_0(x_1, x_2) = x_1^{\beta} x_2^{1-\beta}$ (with $\beta \in [0, 1]$) from Definition 8.[1] Although the focus is on the special case I2CR-3, the convexity proof we present is general and will imply that I2CR is a family of convex optimization problems. For a fixed h, any new relaxation of IBM Model 2 could then be optimized using a mini-batch EG method as discussed in Chapter 3 [33]. Because of the convexity of the problems that result, the optimization methods above are guaranteed to converge to a global solution.

4.2.1 The I2CR-3 Problem

The I2CR-3 problem is a special case of I2CR shown in Figure 4.1 using $h = f_0$. The key difference between this model and IBM Model 2 is that in the objective of IBM Model 2 we have replaced

[1]Note that there is some similarity of the resulting objective function to methods that use deterministic annealing for EM ([39]; [32]) In annealing approaches the objective would be $(x_1 x_2)^{\beta}$ where β is initially close to 0, and is then progressively increased to a value of 1. This prior work does not make the connection to convex objectives when $\beta = 1/2$, and unlike our approach varies β between 0 and 1 within their algorithm.

terms of the type $t(f_j|e_i) \times d(i|j)$ by $t^{\beta}(f_j|e_i) \times d^{1-\beta}(i|j)$, where $\beta \in [0,1]$. We now state the main result needed to show that the objective of I2CR-3 is concave:

Lemma 2. *Let $\mathcal{T}$ be a subset of $[n]$ and consider $h : \mathbb{R}^n_{++} \to \mathbb{R}_+$ given by*

$$h(x_1, \ldots, x_n) = \prod_{i \in \mathcal{T}} x_i^{\alpha_i} \ ,$$

where $\alpha_i \in (0,1)$ $\forall i \in \mathcal{T}$ and $\sum_{i \in \mathcal{T}} \alpha_i = 1$. Then h is concave.

Proof. Let $g : \mathbb{R}^{|\mathcal{T}|}_{++} \to \mathbb{R}_+$ be given by

$$g(x_1, \ldots, x_{|\mathcal{T}|}) = \prod_{i=1}^{|\mathcal{T}|} x_i^{\alpha_i}$$

and note that g is concave by Theorem 4.1. Next we note that $h(x) = g(Ax + b)$ where $b = 0$ and $A \in \mathbb{R}^{n \times |\mathcal{T}|}$ is a suitably chosen matrix which projects down from dimension n to $|\mathcal{T}|$. By the composition rule of a concave function with a linear transformation, h is a concave function [7]. □

Using the above Lemma, we can prove that functions such as

$$h(x_1, x_2, x_3) = \sqrt{x_1 x_2} + \sqrt{x_2 x_3}$$

are concave since they are the sum of two concave functions. We use this observation in the following theorem.

Theorem 6. *The objective of I2CR-3 is concave.*

Proof. Fix a specific training pair index k and target word position j within the objective of I2CR-3 given by Eq. 4.3. We first note that the log is an increasing concave function (we define $\log(x)$ to be $-\infty$ if $x \leq 0$). Using Lemma 2 repeatedly the sum inside the logarithm in the objective of I2CR-3 (Eq. 4.3) is a sum of concave functions, and is hence itself concave. It is a well known rule that composing a concave increasing function (such as the logarithm) and a concave function yield a concave function [7]. Hence, for a fixed k and j, the objective of I2CR-3 is concave. Since the objective in Eq. 4.3 is a sum of concave functions, the result now follows. □

Theorem 6 implies that I2CR-3 is a convex optimization problem since its objective is concave and the constraints form a polytope. In fact, note that an analogous Lemma 2 would hold for any concave function h. With this observation we now have a recipe that can be carried out for any positive-valued concave function h thus yielding our main result: I2CR is a family of convex relaxations for IBM Model 2. In particular, this recipe is more general than the linearization technique in Chapter 3 and can be carried out for any concave function h in Figure 4.1. By using $h = f_{-\infty} = \min$ and applying Theorem 2 we have the tightest such relaxation: I2CR-1 [33]. Interestingly, we will see later that a tighter relaxation does not necessarily give better alignment quality. Specifically, if we set $\alpha_i = 1/n \ \forall i$, we have that

$$f_{1/2}(x) = \left(\sum_{i=1}^{n} \frac{\sqrt{x_i}}{n} \right)^2 \tag{4.4}$$

and the *harmonic* mean

$$f_{-1}(x) = n \left(\sum_{i=1}^{n} 1/x_i \right)^{-1} \tag{4.5}$$

are both concave and using analogous versions of Lemma 2 and Theorem 6 we then have a family of relaxations for latent variable models involving the product of probabilities. Moreover, we note that for $x_1, x_2 \in (0, 1]$:

$$x_1 x_2 \le \min\{x_1, x_2\} \le \frac{2x_1x_2}{x_1 + x_2} \le \sqrt{x_1 x_2} \ , \tag{4.6}$$

so that the usage of the min as in [33] is a better approximation of the product under our general setup. Finally, we note that we could use other concave functions outside of the generalized means to create relaxations. For example, we have that $h(x_1, x_2) = \sqrt{(1 - e^{-x_1})(1 - e^{-x_2})}$ is concave [7] on $\mathbb{R}^2_+$ and is not in the generalized means family. There are infinitely many nontrivial relaxations to consider.

As a final comment, we remark that the new relaxation is not *strictly* convex for *all* datasets. However, similar to IBM Model 2, our sense is that the symmetries in the data that would result in non-strict convexity will be rare in real datasets — much more rare than the case of IBM Model 1, for which it is well known that the objective is not strictly convex for real-world datasets [41]. We leave further study of this to future work.[2]

[2]Noting that for $(\alpha_1, \alpha_2) \in (0, 1)^2$ with $\alpha_1 + \alpha_2 < 1$ $f_0(x_1, x_2) = x_1^{\alpha_1} x_2^{\alpha_2}$ is strictly concave ([44]), there is an easy remedy to guarantee strict convexity. In particular, using a degenerate f_0 we get the same EM algorithm as in Figure 3 (change $(\beta, 1 - \beta)$ to (α_1, α_2)), but now have a strictly convex relaxation. Besides this, we could also use an l_2 regularizer.

4.2.2 The I2CR-4 Problem

Our initial experiments with I2CR-3 lead to better performance than IBM Model 1, but did not yield results as good as those of Model 2. In Chapter 3 we obtained better performance by appending an IBM Model 1 objective to the original convex relaxation I2CR-1 that we derived, and we felt that this might work for I2CR-3 as well. To this end we call our new model I2CR-4 and note that its objective is the sum of one likelihood which places all its importance on the lexical terms (IBM 1) and another (I2CR-3) that distributes weight on the lexical and distortion term via the geometric weighted mean:

$$\begin{aligned} & \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} t(f_j^{(k)}|e_i^{(k)}) \\ + \quad & \frac{1}{2n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} t^{\beta}(f_j^{(k)}|e_i^{(k)})d^{1-\beta}(i|j)\ , \end{aligned}$$

This new model is still a convex optimization problem since its objective is concave (the sum of two concave functions is concave).

4.3 An EM Algorithm for I2CR-4

We describe an EM algorithm for optimizing the I2CR-4 problem in Figure 4.2, and note that the memory and time requirements are the same as those of IBM Model 2's EM algorithm. We find it appealing to introduce a relaxation based on the weighted geometric mean specifically because a simple EM algorithm can be derived. In particular, we note that that the likelihood function we study is of the form

$$L(\mathbf{t},\mathbf{d}) = \frac{\sum_{k=1}^{n}\log p_{IBM1}(f^{(k)}|e^{(k)})p_{I2CR-3}(f^{(k)}|e^{(k)})}{n}\ ,$$

where p_{IBM1} and p_{I2CR-3} are the likelihoods of IBM Model 1 and I2CR-3. Under these models, denote the alignment posteriors by $q_{IBM1}(a^{(k)}|e^{(k)}, f^{(k)})$ and $q_{I2CR-3}(a^{(k)}|e^{(k)}, f^{(k)})$, as usual. Using Jensen's inequality we have the above likelihood is equal to the sum of

$$\sum_{a^{(k)}} q_{IBM1}(a^{(k)}|e^{(k)}, f^{(k)})\log\frac{p_{IBM1}(f^{(k)}, a^{(k)}|e^{(k)})}{q_{IBM1}(a^{(k)}|e^{(k)}, f^{(k)})}$$

and

$$\sum_{a^{(k)}} q_{I2CR-3}(a^{(k)}|e^{(k)}, f^{(k)}) \log \frac{p_{I2CR-3}(f^{(k)}, a^{(k)}|e^{(k)})}{q_{I2CR-3}(a^{(k)}|e^{(k)}, f^{(k)})} \ .$$

In an M-step iteration of EM, we fix the posteriors and maximize the p terms for the lexical and distortion parameters. For IBM Model 1, the EM algorithm is known. For I2CR-3, the EM algorithm follows the same essentially path as that of IBM Model 2 the only difference being that we insert β or $1-\beta$ to modify the balance between counts (this latter fact follows from the known property of the logarithm: $\log(t^{\beta}d^{1-\beta}) = \beta\log(t) + (1-\beta)\log(d)$). Since both these optimization problems have the same multinomial flavor, the Lagrange multipliers act as normalizing constants and we can just combine the gotten counts. The above discussion leads to the main upshot: for the I2CR-4 lexical parameter updates we collect counts arising from IBM Model 1 and I2CR-3 and renormalize as needed. This last bit of logic is what we have summarized in Figure 4.2.

4.4 Decoding with I2CR-3 and I2CR-4

To obtain the highest probability alignment of a pair (e, f) using an IBM Model we need to find the $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_m)$ which yields the highest probability $p(f, \hat{a}|e)$. There are various ways to use the estimated parameters from the IBM Models in decoding. For one, we could find the optimal alignment for I2CR-4 using IBM Model 2's rule (this is the optimal rule for I2CR-3 as well). On the other hand, using the same methods as presented in Chapter 3 we can find the optimal vector $\hat{a}$ by splitting the maximization over the components of $\hat{a}$ and focusing on finding $\hat{a}_j$ given by

$$\hat{a}_j = \text{argmax}_{i=0}^{l}\{t^{1+\beta}(f_j|e_i)d^{1-\beta}(i|j)\} \ .$$

Finally, as a check of the model's validity, we also decode using IBM Model 1's rule. Since the EM updates for IBM Model 1 do not take position at all into account, any reasonable convex relaxation of IBM Model 2 should always beat IBM Model 1 in lexical parameter quality.

4.5 Experiments

In this section we describe experiments using the I2CR-3 and I2CR-4 optimization problems combined with the EM algorithm for these problems. For our experiments we only used $\beta = \frac{1}{2}$, but note that β can be cross-validated for optimal performance.

```
1: **Input**: Define $E$, $F$, $L$, $M$, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3. An integer $T$
   specifying the number of passes over the data. A weighting parameter $\beta \in [0, 1]$.
2: **Parameters:**
   • A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
   • A parameter $d(i|j)$ for each $i, \in [L]_0, j \in [M]$.
3: **Initialization:**
   • $\forall e \in E,\ f \in D(e)$, set $t(f|e) = 1/|D(e)|$.
   • $\forall i \in [L]_0,\ j \in [M]$, set $d(i|j) = 1/(L+1)$.
4: **EM Algorithm:**
5: **for all** $t = 1 \dots T$ **do**
6:     $\forall e \in E,\ f \in D(e),\ count(f, e) = 0$
7:     $\forall e \in E,\ count(e) = 0$
8:     $\forall i \in [L]_0, j \in [M]$, $count(i, j) = 0$
9:     $\forall j \in [M]$, $count(j) = 0$
10:    **EM Algorithm: Expectation**
11:    **for all** $k = 1 \dots n$ **do**
12:        **for all** $j = 1 \dots m_k$ **do**
13:            $\delta_1[i] = \delta_2[i] = 0\ \forall i \in [l_k]_0$
14:            $\Delta_1 = \Delta_2 = 0$
15:            **for all** $i = 0 \dots l_k$ **do**
16:                $\delta_1[i] = t(f_j^{(k)}|e_i^{(k)})$
17:                $\delta_2[i] = t^{\beta}(f_j^{(k)}|e_i^{(k)})d^{1-\beta}(i|j)$
18:                $\Delta_1 \mathrel{+}= \delta_1[i]$
19:                $\Delta_2 \mathrel{+}= \delta_2[i]$
20:            **for all** $i = 0 \dots l_k$ **do**
21:                $\delta_1[i] = \frac{\delta_1[i]}{\Delta_1}$
22:                $\delta_2[i] = \frac{\delta_2[i]}{\Delta_2}$
23:                $count(f_j^{(k)}, e_i^{(k)}) \mathrel{+}= \delta_1[i] + \beta\delta_2[i]$
24:                $count(e_i^{(k)}) \mathrel{+}= \delta_1[i] + \beta\delta_2[i]$
25:                $count(i, j) \mathrel{+}= (1-\beta)\delta_2[i]$
26:                $count(j) \mathrel{+}= (1-\beta)\delta_2[i]$
27:    **EM Algorithm: Maximization**
28:    **for all** $e \in E$ **do**
29:        **for all** $f \in D(e)$ **do**
30:            $t(f|e) = \frac{count(e,f)}{count(e)}$
31:    **for all** $\forall i \in [L]_0, j \in [M]$, **do**
32:        $d(i|j) = \frac{count(i,j)}{count(j)}$
33: **Output:** $t$, $d$ parameters.
```

Figure 4.2: Pseudocode for T iterations of the EM Algorithm for the I2CR-4 problem.

4.5.1 Data Sets

For our alignment experiments, we used a subset of the Canadian Hansards bilingual corpus with 247,878 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data [26]. As a second corpus, we considered a training set of 48,706 Romanian-English sentence-pairs, a development set of 17 sentence pairs, and a test set of 248 sentence pairs [26]. For our SMT experiments, we choose a subset of the English-German Europarl bilingual corpus, using 274,670 sentences for training, 1,806 for development, and 1,840 for test.

4.5.2 Methodology

For each of the models we follow convention in applying the following methodology: first, we estimate the t and d parameters using models in both source-target and target-source directions; second, we find the most likely alignment for each development or test data sentence in each direction; third, we take the intersection of the two alignments as the final output from the model.

For our experiments, we report results in both AER (lower is better) and F-Measure (higher is better) [28]. There is evidence [18] that F-Measure is better correlated with translation quality when the alignments are used in a full system.

In training IBM Model 2 we first train IBM Model 1 for 5 iterations to initialize the t parameters, then train IBM Model 2 for a further 15 iterations [28]. For the I2CR models, we use 15 iterations over the training data and seed all parameters to uniform probabilities. Since the development data we use is rather small, for all models considered we report F-Measure and AER results for each of the 15 iterations, rather than picking the results from a single iteration. Table 4.1 contains our results for the Hansards data. For the Romanian data, we obtained similar behavior, but we leave out these results due to space limitations.

From our experiments, we see that both I2CR-4 and I2CR-3 converge to solutions which give better alignment quality than those of IBM Model 1. Moreover, I2CR-3 is strictly speaking worse than IBM Model 2 and its performance lies in-between that of IBM Model 1 and IBM Model 2. On the other hand, extracting the alignments from I2CR-4 with its natural decoding rule (using $t \times \sqrt{t \times d}$) produces better F-Measure scores than those of IBM Model 2. We feel that even though our convex models are not superior in every way to IBM Model 2, their relatively easy structure and similarity to IBM Model 2 offer some deep insights into what can be accomplished with a convex relaxation. Lastly, we note that it is possible that the balance between the t and d parameters in I2CR-3 should be more carefully chosen within the weighted geometric mean (recall that we used $\beta = 1/2$) to produce the optimal results. Indeed, if we had set $\beta = 1$ in I2CR-3 we get IBM Model 1; on the other hand, setting $\beta = 0$ gives a model that ignores lexical parameters and has weak performance.

So as to better understand the need for an IBM Model 1 objective within our convex relaxation, we also compared I2CR-3 with I2CR-1 trained via the setup in [33]. Our analysis found that I2CR-1 got AER and F-Measure scores that were very close to those of IBM Model 1 (using the same setup as [33], I2CR-1 has AER and F-Measure numbers that hover around .19 and .71, respectively, while IBM Model 1 has AER and F-Measure numbers close to .21 and .70, respectively). Since I2CR-3 performs better than I2CR-1, what this says is that even though the min is a stronger relaxation of the product of two probabilities than the square root (c.f. Theorem 2), the objective (value) difference between a convex relaxation and the original problem it estimates is not the most important feature when picking between various relaxations.

Lastly, we also conducted SMT experiments using the cdec system [17] on a subset of the Europarl English-German data using BLEU as our metric [29] along with the "grow-diagonal-final" heuristic [28]. In computing BLEU, we ran cdec three times over the data and report the average test BLEU score achieved. Using alignments generated by IBM Model 2 and I2CR-4 we respectively obtained BLEU scores of **0.175202** and **0.1751417**. With the default FastAlign system cdec obtained **0.177983** BLEU.

4.6 A Strictly Concave IBM Model 1

As an application of the relaxation methods we discussed, we now detail a very simple method to make IBM Model 1 strictly concave with a unique optimal solution without the need for appending an l_2 loss. This application addresses the problem discussed in (Toutanova et al., 2011) [41], where it show that although IBM Model 1 is convex, it has multiple optima that, although each have the same log-likelihood cost, differ in F-Measure and AER significantly. Moreover, our goal here is to offer a more structured method to improve IBM Model 1 in a more structured manner than the very successful but heuristic set of methods addressed in [27].

Theorem 7. *Consider IBM Model 1 and modify its objective to be*

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) \tag{4.7}$$

where $h_{i,j,k} : \mathbb{R}_+ \to \mathbb{R}_+$ is strictly concave. With the new objective and the same constraints as IBM Model 1, this new optimization problem is strictly concave.

Proof. To prove the result, we now show that the new likelihood function

$$L(\mathbf{t}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) \ ,$$

is strictly concave (concavity follows in the same way trivially). Suppose by way of contradiction that there are feasible solutions (t) $\neq$ (t$'$) and $\theta \in (0,1)$ such that equality hold for Jensen's inequality. Since (t) $\neq$ (t$'$) we must have that there must be some (k,j,i) such that $t(f_j^{(k)}|e_i^{(k)}) \neq t'(f_j^{(k)}|e_i^{(k)})$ and so since $h_{i,j,k}$ is strictly concave we have that Jensen's inequality is strict:

$$\begin{aligned}&h_{i,j,k}(\theta t(f_j^{(k)}|e_i^{(k)}) + (1-\theta)t'(f_j^{(k)}|e_i^{(k)}))\\ &\quad > \theta h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) + (1-\theta)h_{i,j,k}(t'(f_j^{(k)}|e_i^{(k)}))\end{aligned}$$

Using Jensen's inequality, the monotonicity of the log, and the above strict inequality we have

$$\begin{aligned}&L(\theta\mathbf{t} + (1-\theta)\mathbf{t}')\\ &= \sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(\theta t(f_j^{(k)}|e_i^{(k)}) + (1-\theta)t'(f_j^{(k)}|e_i^{(k)}))\\ &> \sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} \theta h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) + (1-\theta)h_{i,j,k}(t'(f_j^{(k)}|e_i^{(k)}))\\ &\geq \theta\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)}))\\ &+ (1-\theta)\sum_{k=1}^{n}\sum_{j=1}^{m_k}\log\sum_{i=0}^{l_k} h_{i,j,k}(t'(f_j^{(k)}|e_i^{(k)}))\\ &= \theta L(\mathbf{t}) + (1-\theta)L(\mathbf{t}')\end{aligned}$$

Input: Define E, F, L, M, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \ldots n$, $D(e)$ for $e \in E$ as in Section 2.3. A set of strictly concave functions $h_{i,j,k} : \mathbb{R}_+ \to \mathbb{R}_+$.

Parameters:

• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), \qquad t(f|e) \geq 0 \tag{4.8}$$

$$\forall e \in E, \qquad \sum_{f \in D(e)} t(f|e) = 1 \tag{4.9}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \log \sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) \tag{4.10}$$

with respect to the $t(f|e)$ parameters.

Figure 4.3: The IBM Model 1 strictly concave optimization problem.

□

In Theorem 7 it is crucial that each $h_{i,j,k}$ be strictly concave within $\sum_{i=0}^{l_k} h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)}))$. For example, we have that $\sqrt{x_1} + x_2$ is concave but not strictly concave and the proof of Theorem 7 would break down. To see this, we can consider $(x_1, x_2) \neq (x_1, x_3)$ and note that equality holds in Jensen's inequality. We should be clear: the main reason why Theorem 7 works is that we have $h_{i,j,k}$ are strictly concave (on $\mathbb{R}_+$) and all the lexical probabilities that are arguments to L are present within the log-likelihood.

4.6.1 Parameter Estimation via EM

In this section we detail a particular choice of $h_{i,j,k}$ that will enable us to derive an easy EM algorithm. We first need the following:

Lemma 3. *Consider $h : \mathbb{R}_+ \to \mathbb{R}_+$ given by $h(x) = x^\beta$ where $\beta \in (0, 1)$. Then h is strictly concave.*

Proof. The proof of this lemma is elementary and follows since the second derivative given by $h''(x) = \beta(\beta-1)x^{\beta-2}$ is strictly negative. $\square$

For our concrete experiments, we picked a model based on Lemma 1 and used $h(x) = \alpha x^{\beta}$ with $\alpha, \beta \in (0,1)$ so that

$$h_{i,j,k}(t(f_j^{(k)}|e_i^{(k)})) = \alpha(f_j^{(k)}, e_i^{(k)}) t^{\beta(f_j^{(k)}, e_i^{(k)})}(f_j^{(k)}|e_i^{(k)}) \ .$$

Using this setup, parameter estimation for the new model can be accomplished via a slight modification of the EM algorithm for IBM Model 1. In particular, we have that the posterior probabilities of this model factor just as those of the standard Model 1 and we have an M step that requires optimizing

$$\sum_{a^{(k)}} q(a^{(k)}|e^{(k)}, f^{(k)}) \log p(f^{(k)}, a^{(k)}|e^{(k)})$$

where

$$q(a^{(k)}|e^{(k)}, f^{(k)}) \propto \prod_{j=1}^{m_k} h_{a_j^{(k)},j,k}(t(f_j^{(k)}|e_{a_j^{(k)}}^{(k)}))$$

are constants gotten in the E step. This optimization step is very similar to the regular Model 1 M step since the β drops down using $\log t^{\beta} = \beta \log t$; the exact same count-based method can be applied. The details of this algorithm are in Fig. 4.4.

4.6.2 Choosing β and α

The performance of our new model will rely heavily on the choice of $\alpha(e_i^{(k)}, f_j^{(k)})$, $\beta(e_i^{(k)}, f_j^{(k)}) \in (0,1)$ we use. In particular, we could make β depend on the association between the words, or the words' positions, or both. One classical measure of word association is the dice coefficient [28] given by

$$dice(e,f) = \frac{2c(e,f)}{c(e)+c(f)} \ .$$

In the above, the count terms c are the number of training sentences that have either a particular word or a pair of of words (e,f). As with the other choices we explore, the dice coefficient is a fraction between 0 and 1, with 0 and 1 implying less and more association, respectively. Additionally, we make use of positional constants like those of the IBM Model 2 distortions given by

$$d(i|j,l,m) = \begin{cases} \frac{1}{(l+1)Z(j,l,m)} & : i = 0 \\ \frac{le^{-\lambda|\frac{i}{l}-\frac{j}{m}|}}{(l+1)Z(j,l,m)} & : i \neq 0 \end{cases}$$

```
1: **Input**: Define $E$, $F$, $L$, $M$, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3. An integer $T$ specifying the number of passes over the data. A set of weighting parameter $\alpha(e,f), \beta(e,f) \in (0,1)$ for each $e \in E, f \in D(e)$. A tuning parameter $\lambda > 0$.
2: **Parameters:**
   • A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
3: **Initialization:**
   • $\forall e \in E,\ f \in D(e)$, set $t(f|e) = 1/|D(e)|$.
4: **EM Algorithm:**
5: **for all** $t = 1 \dots T$ **do**
6:     $\forall e \in E,\ f \in D(e),\ count(f,e) = 0$
7:     $\forall e \in E,\ count(e) = 0$
8:     **EM Algorithm: Expectation**
9:     **for all** $k = 1 \dots n$ **do**
10:        **for all** $j = 1 \dots m_k$ **do**
11:            $\delta_1[i] = 0\ \forall i \in [l_k]_0$
12:            $\Delta_1 = 0$
13:            **for all** $i = 0 \dots l_k$ **do**
14:                $\delta_1[i] = \alpha(f_j^{(k)}, e_i^{(k)}) t^{\beta(f_j^{(k)}, e_i^{(k)})}(f_j^{(k)}|e_i^{(k)})$
15:                $\Delta_1 \mathrel{+}= \delta_1[i]$
16:            **for all** $i = 0 \dots l_k$ **do**
17:                $\delta_1[i] = \frac{\delta_1[i]}{\Delta_1}$
18:                $count(f_j^{(k)}, e_i^{(k)}) \mathrel{+}= \beta(f_j^{(k)}, e_i^{(k)})\delta_1[i]$
19:                $count(e_i^{(k)}) \mathrel{+}= \beta(f_j^{(k)}, e_i^{(k)})\delta_1[i]$
20:    **EM Algorithm: Maximization**
21:    **for all** $e \in E$ **do**
22:        **for all** $f \in D(e)$ **do**
23:            $t(f|e) = \frac{count(e,f)}{count(e)}$
24: **Output:** $t$ parameters
```

Figure 4.4: Pseudocode for T iterations of the EM Algorithm for the strictly convex IBM Model 1 problem.

In the above, $Z(j,l,m)$ is the partition function discussed in [16]. The previous measures all lead to potential candidates for $\beta(e,f)$, we have $t(f|e) \in (0,1)$, and we want to enlarge competeting values when decoding (we use αt^{β} instead of t when getting the highest probability alignment). The above then implies that we'll have the word association measures inversely proportional to β, and so we set $\beta(e,f) = 1 - dice(e,f)$ or $\beta(e,f) = 1 - d(i|j,l,m)$. In our experiments we picked $\alpha(f_j^{(k)}, e_i^{(k)}) = d(i|j,l_k,m_k)$ or 1. Lastly, we note that for the distortions $d(i|j,l,m)$ we hold λ to a constant of 16 and do not estimate this variable (if optimal performance is needed, $\lambda = 16$ can be chosen by cross-validation on a small trial data set).

4.6.3 Experiments

For this section we use the same Canadian Hansards bilingual corpus with 247,878 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data [26]. Below we report results in both AER (lower is better) and F-Measure (higher is better) [28] for the English to French translation direction.

In the above, we note that when using

$$h(t(f_j|e_i)) = d(i|j,l,m)t(f_j|e_i)$$

with d constant we cannot use Theorem 7 since h is linear. Most likely, the strict concavity of the model will hold because of the asymmetry introduced by the d term; however, there will be a necessary dependence on the data set.

Table 4.2 contains our results for the Hansards data. Our experiments show that using

$$h(t(f_j|e_i)) = (t(f_j|e_i))^{1-d(i|j,l,m)}$$

yields the best F-Measure performance and is not far off in AER from the “fake" IBM Model 2 (gotten by setting $(\alpha, \beta) = (d, 1)$) whose results are in column 2. Moreover, we note that dice does not lead to quality β exponents and that, unfortunately, combining methods as in column 5 ($(\alpha, \beta) = (d, 1-d)$) does not necessarily lead to addictive gains in AER and F-Measure performance.

4.7 Conclusions and Future Work

This section described the main work presented in [35]. This last research generalizes the work [33] and introduces a class of convex relaxations for the unsupervised learning of alignments in statistical machine translation with performance comparable to the commonly-used IBM Model 2. Extending the convexity results of [33] allows us to better understand the old results and develop further applications. With reference to IBM Model 2, future work would consider different relaxations within the class we have introduced, and apply our method to other NLP tasks and problems beyond alignment tasks.

Lastly, as an application of the main presented in [35], we also showed how IBM Model 1 can be made into a strictly convex optimization problem. In this final part, we take a moment to also compare our work with the classical IBM 1 work of [27]. Summarizing [27], we note that this work improves substancially upon the classical IBM Model 1 by introducing a set of heuristics, among which are to (1) modify the lexical parameter dictionaries (2) introduce an initialization heuristic (3) modify the standard IBM 1 EM algorithm by introducing smoothing (4) tune additional parameters. However, we stress that the main concern of this work is not just heuristic-based empirical improvement, but also structured learning. In particular, although using an regularizer l_2 and the methods of [27] would yield a strictly concave version of IBM 1 as well (with improvements), it is not at all obvious how to choose the learning rate or set the penalty on the lexical parameters. The goal of our work was to offer a new, alternate form of regularization. Moreover, since we are changing the original log-likelihood, our method can be thought of as way of *bringing the l_2 regularizer inside the log likelihood*. Like [27], we also achieve appreciable gains but have just one tuning parameter (when $\beta = 1 - d$ we just have the centering λ parameter) and do not break the probabilistic interpretation any more than appending a regularizer would (our method modifies the log-likelihood but the simplex constrains remain).

For the strictly convex IBM 1 family we studied, we looked at a specific member within the class that allows for an easy EM algorithm and we conducted experiments showing 30% improvement over the standard IBM Model 1 algorithm. For further research, we note that picking the optimal $h_{i,j,k}$ is an open question, so provably finding and justifying this model is one topic of interest.

Model	IBM2	IBM2	I2CR-3	I2CR-3	I2CR-4	I2CR-4	I2CR-4
Decoding Rule	t	$t \times d$	t	$t \times d$	t	$t \times d$	$t \times \sqrt{t \times d}$
Iteration				AER			
0	0.2141	0.2141	0.9273	0.9273	0.9273	0.9273	0.9273
1	0.2128	0.1609	0.3697	0.3786	0.3669	0.3790	0.3569
2	0.2013	0.1531	0.2614	0.2235	0.2408	0.2090	0.2038
3	0.1983	0.1477	0.2333	0.1879	0.2209	0.1769	0.1754
4	0.1950	0.1458	0.2116	0.1783	0.2153	0.1668	0.1646
5	0.1941	0.1455	0.2088	0.1753	0.2067	0.1632	0.1592
6	0.1926	**0.1436**	0.2063	0.1739	0.2058	0.1600	0.1559
7	0.1912	**0.1436**	0.2048	0.1726	0.2046	0.1566	0.1551
8	0.1904	0.1449	0.2044	0.1730	0.2044	0.1549	0.1540
9	0.1907	0.1454	0.2041	0.1727	0.2047	0.1527	0.1534
10	0.1913	0.1451	0.2042	0.1721	0.2045	0.1524	0.1524
11	0.1911	0.1452	0.2042	**0.1718**	0.2039	0.1515	0.1520
12	0.1901	0.1454	**0.2040**	0.1722	0.2035	0.1513	0.1514
13	0.1899	0.1462	0.2041	0.1721	0.2032	0.1510	0.1511
14	**0.1898**	0.1471	0.2041	0.1724	0.2032	0.1509	0.1508
15	0.1900	0.1474	0.2041	0.1727	**0.2031**	**0.1505**	**0.1505**
Iteration				F-Measure			
0	0.7043	0.7043	0.0482	0.0482	0.0482	0.0482	0.0482
1	0.7049	0.7424	0.5610	0.5446	0.5664	0.5455	0.5712
2	0.7127	0.7468	0.6603	0.6910	0.6818	0.7059	0.7149
3	0.7116	0.7489	0.6838	0.7201	0.6977	0.7302	0.7385
4	**0.7130**	**0.7501**	0.7036	0.7255	0.7020	0.7369	0.7471
5	0.7124	0.7495	0.7060	0.7252	0.7102	0.7394	0.7515
6	0.7121	0.7501	0.7079	0.7257	0.7103	0.7411	0.7531
7	0.7132	0.7493	0.7084	0.7260	0.7111	0.7443	0.7531
8	0.7132	0.7480	**0.7085**	0.7252	0.7113	0.7457	0.7541
9	0.7127	0.7473	0.7084	0.7254	0.7115	0.7476	0.7547
10	0.7116	0.7474	0.7082	**0.7261**	0.7113	0.7482	0.7559
11	0.7113	0.7466	0.7080	**0.7261**	0.7117	0.7493	0.7563
12	0.7123	0.7463	0.7081	0.7256	0.7118	0.7496	0.7568
13	0.7119	0.7460	0.7081	0.7257	0.7121	0.7497	0.7571
14	0.7122	0.7451	0.7081	0.7253	0.7121	0.7497	0.7575
15	0.7122	0.7447	0.7081	0.7250	**0.7122**	**0.7501**	**0.7577**

Table 4.1: Intersected results on the English-French data for IBM Model 2, I2CR-3, and I2CR-4 trained for 15 EM using either the IBM1 (t), IBM2 ($t \times d$), or I2CR-4 ($t \times \sqrt{t \times d}$) decoding.

(α, β)	$(1,1)$	$(d,1)$	$(1,1-dice)$	$(1,1-d)$	$(d,1-d)$
Iteration			AER		
0	0.8716	0.6750	0.6240	0.6597	0.5570
1	0.4426	0.2917	0.4533	0.2738	0.3695
2	0.3383	0.2323	0.4028	0.2318	0.3085
3	0.3241	0.2190	0.3845	0.2252	0.2881
4	0.3191	0.2141	0.3751	0.2228	0.2833
5	0.3175	0.2118	0.3590	0.2229	0.2812
6	0.3160	0.2093	0.3566	0.2231	0.2793
7	0.3203	0.2090	0.3555	0.2236	0.2783
8	0.3198	0.2075	0.3546	0.2276	0.2777
9	0.3198	0.2066	0.3535	0.2323	0.2769
10	0.3177	0.2065	0.3531	0.2352	0.2769
Iteration			F-Measure		
0	0.0427	0.1451	0.2916	0.1897	0.2561
1	0.4213	0.5129	0.4401	**0.5453**	0.4427
2	0.5263	0.5726	0.4851	**0.5940**	0.5014
3	0.5413	0.5852	0.5022	**0.6047**	0.5199
4	0.5480	0.5909	0.5111	**0.6085**	0.5255
5	0.5500	0.5939	0.5264	**0.6101**	0.5273
6	0.5505	0.5959	0.5282	**0.6101**	0.5286
7	0.5449	0.5965	0.5298	**0.6096**	0.5296
8	0.5456	0.5977	0.5307	**0.6068**	0.5300
9	0.5451	0.5985	0.5318	**0.6040**	0.5309
10	0.5468	0.5984	0.5322	**0.6024**	0.5311

Table 4.2: Results on the English-French data for various (α, β) settings. The standard IBM Model 1 is column 1 and corresponds to a setting of $(1,1)$. The not necessarily strictly concave model with $(d,1)$ setting gives the best AER, while the strictly concave model given by the $(1,1-d)$ setting has the highest F-Measure.

Chapter 5

A Convex Alternative for the HMM Alignment Model

5.1 Introduction

In this section we outline the construction of a convex HMM alternative for word alignment. In particular, the model we try to relax is the HMM word alignment model of (Vogel et al., 1996) [43]. A standard model, the HMM model is implemented in GIZA++ and performs very well, much better than IBM Model 2 [28]. Although the new convex model's performance does not surpass that of the HMM, it nevertheless archives strong empirical results. In particular, the new convex model improves upon the former convex relaxations of IBM Model 2 by more than 30% and also performs better than the improved IBM2 FastAlign model of (Dyer et al., 2013) [16].

5.2 Distortion and Transition parameter structure

The structure of IBM Model 2's distortion parameters and the HMM's transition parameters is important and used in our model as well, so we detail this here. The main idea for using these particular distortions is introduced in (Dyer et al., 2013) [16] where a new IBM Model 2 is proposed and detailed. The distortions of our model are parametrized by forcing the model to concentrate its alignments on the diagonal, while the HMM style transitions favor jumps to the next adjacent source word.

5.2.1 Distortion Parameters for IBM2

Let $\lambda > 0$. For the IBM Model 2 distortions we set the NULL word probability as $d(0|j, l, m) = p_0$, where $p_0 = \frac{1}{l+1}$ and note that this will generally depend on the source sentence length within a bitext training pair that we are considering. For $i \neq 0$ we set

$$d(i|j, l, m) = \frac{(1 - p_0)e^{-\lambda|\frac{i}{l} - \frac{j}{m}|}}{Z_\lambda(j, l, m)} ,$$

where $Z_\lambda(j, l, m)$ is a normalization constant as in [16].

5.2.2 Transition Parameters for HMM

Let $\theta > 0$. For the HMM transitions we first set the NULL word generation to $d(0|i, l) = p_0$, with $p_0 = \frac{1}{l+1}$. For source word position $i, i' \neq 0$, we set

$$d(i'|i, l) = \frac{(1 - p_0)e^{-\theta|\frac{i'-1}{l} - \frac{i}{l}|}}{Z_\theta(i, l, m)} ,$$

where $Z_\theta(i, l, m)$ is a suitable normalization constant. Lastly, if $i = 0$ so that we are jumping from the NULL word onto a possibly different word we set $d(i'|0, l) = p_0$.

5.3 Combining IBM Model 2 and the HMM

In formulating the new alignment model, our main goal is to mimic IBM Model 2's structure while at the same time allowing the current word position to know as much about the previous alignment variable. Ultimately, the idea here is to formulate a model that has HMM alignment dependency and still has log-likelihood that factors as that of IBM Model 2 so that we may relax product terms using the geometric mean mechanism of Chapter 4. To this end, we combine IBM Model 2 and the HMM by incorporating the generation of words using the structure of both models. Consider a sentence pair (e, f) with $|e| = l$ and $|f| = m$. Then, for target positions j and $j + 1$ we have target words f_j and f_{j+1} and we assign a joint probability involving the alignments a_j and a_{j+1} as:

$$q(j, a_j, a_{j+1}, l, m) = \tag{5.1}$$

$$t(f_j|e_{a_j})d(a_j|j, l, m)t(f_{j+1}|e_{a_{j+1}})d(a_{j+j}|a_j, l) . \tag{5.2}$$

From the equation above, we use the IBM Model 2's word generation method for position j and the HMM generative structure for position $j+1$. The generative nature of the above procedure introduces dependency between adjacent words two at a time. Since we want to mimic the HMM's structure as much as possible, we devise our likelihood function to mimic the HMM's dependency between alignments using q. The model we introduce, IBM2-HMM, is displayed in Fig 5.2. Essentially, we move the target word position j from 1 to $m_k - 1$ and generate sentences via two likelihoods, one that starts at $j = 1$ and generates alignment two at a time while another starts at $j = 2$ and does same. In what follows, we describe this representation in detail.

We have that the likelihood in Eq. 5.14 is actually the sum of two likelihoods which use equations Eq. 5.1 and 5.2 repeatedly. To this end, we will discuss how our objective is actually

$$\frac{1}{n}\sum_{k=1}^{n}\log\sum_{a^{(k)},b^{(k)}} p(f^{(k)},a^{(k)},b^{(k)}|e^{(k)}) \ , \tag{5.3}$$

where $a^{(k)}$ and $b^{(k)}$ both are alignment vectors whose components are independent and can take on any values in $[l_k]_0$. To see how $p(f,a,b|e)$ comes about, note that we could generate the sentence f by generating pairs $(1,2),(3,4),(5,6)\ldots$ using equations Eqs. 5.1 and 5.2 for each pair. If m is odd, the above alignment generation method misses the last word f_m and so we do not have the term

$$t(f_m|e_{a_m})d(a_m|m,l,m) \tag{5.4}$$

in our model. Due to the above lack of last-word generation, we need to specify a way by which we also generate this pair, and we will resolve this issue below. Taking all this together, the upshot of our discussion is that generating the pair (e,f) in this way gives us that the likelihood for an alignment a would be:

$$p_1(f,a|e) = \prod_{j \text{ odd}} q(j,a_j,a_{j+1},l,m) \ . \tag{5.5}$$

A similar argument to the above also allows us to skip the first target word position and generate pairs $(2,3),(4,5),\ldots$ using Eqs. 5.1 and 5.2. For this generation scheme the probability for alignment b is roughly:

$$p_2(f,b|e) = \prod_{j \text{ even}} q(j, b_j, b_{j+1}, l, m) \ , \tag{5.6}$$

In the above, we note that $p_2(f,b|e)$ misses generating the term

$$t(f_1|e_{a_1})d(a_1|1,l,m) \tag{5.7}$$

associated with f_1.

To rectify the issues with generating sentences via either p_1 or p_2, we now note that using p_1 *and* p_2 in a combined fashion generates all alignment and word pairs since one or the other of these models generates the terms associated with f_1 and f_m. Specifically, using $p(f,a,b|e) = p_1(f,a|e)p_2(f,b|e)$ and factoring the log-likelihood as in IBM Model 1 and 2, we get the log-likelihood in Fig 5.2. Finally, we note that our model's log-likelihood could be viewed as the sum of the log-likelihoods of a model which generates (e,f) using p_1 and another model which generates sentences using p_2. These models share parameters but generate words using different recipes as discussed above.

5.4 Parameter estimation for IBM2-HMM

To optimize our new model, we can use an EM algorithm in the same fashion as (Dyer et al., 2013) [16]. In particular, for the model in question the EM algorithm still applies but we have to use a gradient-based algorithm within the maximization step because we need to optimize for θ and λ and for these parameters deriving expected counts via a standard multinomial EM algorithm does not apply. Alternatively, we could just have θ and λ be two tuning parameters so that we only need to optimize for the multinomial lexical t parameters. In this work, we pursued the latter option as it allows us to derive a cleaner multinomial EM algorithm for the t parameters. Using either optimization method, we note that we are still only approximately solving the main optimization problem since IBM2-HMM is, like the HMM and IBM Model 2, a non-convex optimization.

5.5 A Convex HMM Alternative

We now derive a convex relaxation for the new model we introduced. As a first step, notice that one possible convexification path would follow via the methods developed in Chapter 4: we would be

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3.

Parameters:

• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

• A distortion centering parameter $\lambda > 0$.

• A transition centering parameter $\theta > 0$.

Constraints:

$$\forall e \in E, f \in D(e), t(f|e) \geq 0 \tag{5.8}$$

$$\forall e \in E, \sum_{f \in D(e)} t(f|e) = 1 \tag{5.9}$$

$$\forall i \in [l_k]_0, j \in [m_k], d(i|j, l_k, m_k) \geq 0 \tag{5.10}$$

$$\forall j \in [m_k], \sum_{i \in [l_k]_0} d(i|j, l_k, m_k) = 1 \tag{5.11}$$

$$\forall i, i' \in [l_k]_0, d(i'|i, l_k) \geq 0 \tag{5.12}$$

$$\forall i \in [l_k]_0, \sum_{i' \in [l_k]_0} d(i'|i, l_k) = 1 \tag{5.13}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k - 1} \log \sum_{i=0}^{l_k} \sum_{i'=0}^{l_k} q(j, i, i', l_k, m_k) \tag{5.14}$$

with respect to the parameters $t(f|e)$, $d(i'|i, l)$ $d(i|j, l, m)$, and $q(j, i, i', l_k, m_k)$ set as

$$q(j, i, i', l_k, m_k) = t(f_j^{(k)}|e_i^{(k)}) d(i|j, l, m) t(f_{j+1}^{(k)}|e_{i'}) d(i'|i, l) \tag{5.15}$$

Figure 5.1: The IBM2-HMM Optimization Problem. We use equation (5.1) within the likelihood definition.

to let $d(i|j, l, m)$ and $d(i'|i, l)$ be multinomial probabilities and replace all the terms $q(j, i', i, l, m)$ in (5.14) by $(q(j, i', i, l, m))^{\frac{1}{4}}$. Although this method is feasible, pilot experiments showed that the gotten relaxation is not very competitive and performs worse than IBM Model 2. Further analysis with these type of models explains why this is the case. Specifically, when relaxing product terms $\prod_{j=1}^{n} x_j$ with $(\prod_{j=1}^{n} x_j)^{\frac{1}{n}}$ we have that the approximation becomes weaker and weaker as n gets larger. In particular, we have that a two terms product is approximated well by a geometric mean of two terms, but approximating three term products by a cube root is not a good idea, and the problem only gets worse as we move to more terms. Indeed, as we raise each power of x_j to a smaller and smaller power (as n increases) we encourage variables to be nothing more than uniform and the resulting relaxation loses its goal of being close to the original model objective.

To rectify the above degradation in our convex relaxation, we left in place the structure discussed in Section 5.2 and made λ and θ two tuning parameters which we cross-validated for on a small held out data set. This last modification effectively removed the distortion and transition parameters from the model but we still maintained the structural property of these parameters: namely, we maintained their favoring the diagonal and adjacent alignment. We thus replaced $q(j, i', i, l, m)$ by

$$p(j, i, i, l, m) \propto \sqrt{t(f_j^{(k)}|e_i^{(k)})t(f_{j+1}^{(k)}|e_{i'})}$$

and set the proportionality constant to be $d(i|j, l, m)d(i'|i, l)$. At the risk of not spelling this out exactly, we note that we are using the Chapter 4 mechanism with $h(x_1, x_2) = \sqrt{x_1 x_2} y_1 y_2$ instead of $f(x_1, x_2) = x_1 x_2 y_1 y_2$ where x_1, x_2 would be the lexical t parameters and y_1, y_2 would be the fixed distortion and transition terms. Using this setup we now have a convex objective which approximates the product of two terms by their square root and has the other terms estimated via cross-validation (both θ and λ are not parameters here - they are tuned). Although not as terse as the I2CR models or Chapters 3 and 4, our relaxation can be viewed in the same light as other convex optimization problems in Machine Learning (such as, for example, the SVM) that include parameters to be cross-validated over.

5.6 An EM algorithm for The Convex HMM Alternative

The EM algorithm for the convex relaxation of our alternative is given in Fig 5.3. As the model's objective is the sum of the objectives of two models generated by a multinomial rule, we can get a very succinct EM algorithm. Specifically, we once again have that the log-likelihood of our model is the sum of two independent log-likelihoods and we can use Jensen's inequality and the posterior probabilities to easily derive the expected count updates for the lexical t terms. We can use this since the $\frac{1}{2}$ will drop down and the distortion and transition probabilities are constants. For more details on this and a similar derivation, please refer to Chapter 4 and [35]. For this algorithm, we

Input: Define E, F, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3. Pick $\lambda, \theta > 0$ as in Section 5.2 via cross-validation.

Parameters:

• A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.

Constraints:

$$\forall e \in E, f \in D(e), t(f|e) \geq 0 \tag{5.16}$$

$$\forall e \in E, \sum_{f \in D(e)} t(f|e) = 1 \tag{5.17}$$

Objective: Maximize

$$\frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k - 1} \log \sum_{i=0}^{l_k} \sum_{i'=0}^{l_k} p(j, i, i', l_k, m_k) \tag{5.18}$$

with respect to the parameters $t(f|e)$ and $p(j, i, i', l_k, m_k)$ set as

$$p(j, i, i', l_k, m_k) = \sqrt{t(f_j^{(k)}|e_i^{(k)})} d(i|j, l, m) \sqrt{t(f_{j+1}^{(k)}|e_{i'})} d(i'|i, l)$$

Figure 5.2: The IBM2-HMM convex relaxation optimization problem. Note that the distortions $d(i|j, l, , m)$ and transitions $d(i'|i, l)$ are constants held fixed and parameterized by cross-validated parameters λ and θ as in Section 5.2.

again note that the distortion and transition parameters are constants so that the only estimation necessary is on the lexical t terms Moreover, unlike the original IBM2-HMM model which need not have λ and θ constant (we can then use a more complicated EM algorithm), we *need* these terms to be turning parameters for the objective we work with to be concave.

5.7 Decoding methods for the IBM2-HMM problem

When computing the optimal alignment we wanted to compare our model with the HMM as closely as possible. Because of this, the most natural method of evaluating the quality of the parameters would be to use the same rule as the HMM. Specifically, for a sentence pair (e, f) with $|e| = l$ and

```
1: **Input**: Define $E$, $F$, $(e^{(k)}, f^{(k)}, l_k, m_k)$ for $k = 1 \dots n$, $D(e)$ for $e \in E$ as in Section 2.3. Two parameters $\lambda, \theta > 0$ picked by cross-validation so that the distortions and transitions are constants obeying the structure in Section 5.2. An integer $T$ specifying the number of passes over the data.
2: **Parameters:**
   • A parameter $t(f|e)$ for each $e \in E, f \in D(e)$.
3: **Initialization:**
   • $\forall e \in E,\ f \in D(e)$, set $t(f|e) = \frac{1}{D(e)}$.
4: **EM Algorithm: Expectation**
5: for all $k = 1 \dots N$ do
6:    for all $j = 1 \dots m_k$ do
7:       $\delta = 0$
8:       $\Delta = 0$
9:       for all $i = 0 \dots l_k$ do
10:         for all $i' = 0 \dots l_k$ do
11:            $\delta[i, i'] = p(j, i', i, l_k, m_k)$
12:            $\Delta + = \delta[i, i']$
13:      for all $i = 0 \dots l_k$ do
14:         for all $i' = 0 \dots l_k$ do
15:            $\delta[i, i'] = \frac{\delta[i, i']}{\Delta}$
16:            $counts(f_j^{(k)}, e_i^{(k)}) + = \delta[i, i']$
17:            $counts(e_i^{(k)}) + = \delta[i, i']$
18:            $counts(f_{j+1}^{(k)}, e_{i'}^{(k)}) + = \delta[i, i']$
19:            $counts(e_{i'}^{(k)}) + = \delta[i, i']$
20: **EM Algorithm: Maximization**
21: for all $e \in E$ do
22:    for all $f \in D(e)$ do
23:       $t(f|e) = \frac{counts(e,f)}{counts(e)}$
24: **Output:** $t$ parameters.
```

Figure 5.3: Pseudocode for the EM algorithm of the IBM2-HMM's convex relaxation. As the distortion and transition parameters are constants, the algorithm is very similar to that of IBM Model 1.

$|f| = m$, in HMM decoding we aim to find $(\hat{a}_1, \ldots, \hat{a}_m)$ which maximizes

$$\max_{a_1,\ldots,a_m} \prod_{j=1}^{m} t(f_j|e_{a_j})d(a_j|a_{j-1}, l).$$

As is standard, dynamic programming can now be used to find the Viterbi alignment. Although there are a number of ways we could define the optimal alignment, we felt that the above would be the best since it tests dependance between alignment variables and allows for easy comparison with the GIZA++ HMM, which is our goal. Finding the optimal alignment under the HMM setting is labelled “HMM" in Table 5.1.

We can also find the optimal alignment by taking the objective literally and computing

$$\max_{a_1,\ldots,a_m} p_1(f, a|e)p_2(f, a|e).$$

In this case, we are asking for the optimal alignment that yields the highest probability alignment through generating technique p_1 and p_2. This method of decoding is a lot like the HMM style and also relies on dynamic programming. In this case we have the recursion for Q_{Joint} given by

$$Q_{Joint}(1, i) = t(f_1|e_i)d^2(i|1, l, m) \ ,$$

$\forall i \in [l]_0$, and

$$Q_{Joint}(j, i') = t^2(f_j|e_{i'})d(i'|j, l, m)M_{Joint}(j-1, i') \ ,$$

where $M_{Joint}(j-1, i')$ is

$$M_{Joint}(j-1, i') = \max_{i=0}^{l} \{d(i'|i, l)Q_{Joint}(j-1, i)\} \ ,$$

$\forall \ 2 \leq j \leq m, \forall \ i' \in [l]_0$. Although his is the natural decoding rule, the alignment results gotten by decoding with this method proved to yield weaker results than decoding using standard HMM rules. In some sense, the above empirical results imply that the IBM Model 2 distortion parameters are not as strong (or informative) as the transition parameters of the HMM.

5.8 Experiments

In this section we describe experiments using the IBM2-HMM optimization problem combined with the EM algorithm for parameter estimation. The experiments conducted here use a similar setup to those in [33]. We first describe the data we use, and then describe the experiments we ran.

5.8.1 Data Sets

We use data from the bilingual word alignment workshop held at HLT-NAACL 2003 [26]. We use the Canadian Hansards bilingual corpus, with 743,989 English-French sentence pairs as training data, 37 sentences of development data, and 447 sentences of test data (note that we use a randomly chosen subset of the original training set of 1.1 million sentences, similar to the setting used in [27]). The development and test data have been manually aligned at the word level, annotating alignments between source and target words in the corpus as either "sure" (S) or "possible" (P) alignments, as described in [28]. As is standard, we lower-cased all words before giving the data to GIZA++ and we ignored NULL word alignments in our computation of alignment quality scores.

5.8.2 Methodology

We test several models in our experiments, including experiments of our model, the GIZA++ IBM Model 3 and HMM, as well as the FastAlign IBM Model 2 implementation of (Dyer et al., 2013) [16]. For each of the models we estimate the t and d parameters using models in the English-French source-target direction and present the gotten alignments. Although there are several methods for combining alignments, we felt that the presented direct comparisons would offer the most clear presentation of relative model performance. In training, we employ the standard practice of initializing nonconvex alignment models with simpler nonconvex models. In particular, we initialize, the GIZA++ HMM with IBM Model 2, IBM Model 2 with IBM Model 1, and IBM2-HMM with IBM Model 2 preceded by Model 1, and IBM Model 3 with IBM Model 2.

We measure the performance of the models in terms of *Precision*, *Recall*, *F-Measure*, and *AER* using only sure alignments in the definitions of the first three metrics and sure and possible alignments in the definition of AER, as in [33] and [14]. For our experiments, we report results in both AER (lower is better) and F-Measure (higher is better).

Table 5.1 shows the alignment summary statistics for the 447 sentences present in the Hansard test data. We present alignments quality scores using either the FastAlign IBM Model 2, the GIZA++ HMM, and our model and its relaxation using either the "HMM" or "Joint" decoding. First, we note that in deciding the decoding style for IBM2-HMM, the HMM method is better than the Joint method. We expected this type of performance since HMM decoding introduces positional dependance among the entire set of words in the sentence, which is shown to be a good modeling assumption [43].

From the results in Table 5.1 we see that the HMM outperforms all other models, including IBM2-HMM and its convex relaxation. On the other hand, IBM2-HMM is not far in AER performance from the HMM and both it and its relaxation do better than FastAlign or IBM Model 3 (the results for IBM Model 3 are not presented in Table 5.1; however, a run of $1^5 2^5 3^{15}$ gave AER and F-Measure numbers of 0.1768 and 0.6588, respectively).

Training	$1^5 2H^{10}$	$1^5 2H^{10}$	$2HC^{10}$	$2HC^{10}$	FA^{10}	$1^5 2^5 H^{10}$
Decoding	HMM	Joint	HMM	Joint	IBM2	HMM
Iteration			AER			
1	0.1640	0.1587	0.2001	0.2327	0.5316	0.2715
2	0.1524	0.1546	0.1655	0.1891	0.2237	0.1521
3	0.1475	0.1527	0.1592	0.1794	0.1840	0.1320
4	0.1448	0.1519	0.1564	0.1764	0.1745	0.1231
5	0.1436	0.1516	0.1555	0.1746	0.1674	0.1161
6	0.1436	0.1512	0.1559	0.1749	0.1660	0.1130
7	0.1411	0.1505	0.1555	0.1745	0.1642	0.1116
8	0.1399	0.1500	0.1553	0.1743	0.1619	0.1112
9	0.1390	0.1496	0.1555	0.1741	0.1617	0.1123
10	0.1390	0.1500	0.1555	0.1739	0.1610	0.1121
Iteration			F-Measure			
1	0.6329	0.4831	0.5858	0.5605	0.2854	0.5709
2	0.6398	0.6391	0.6189	0.6003	0.5731	0.6701
3	0.6422	0.6390	0.6252	0.6106	0.6201	0.6903
4	0.6437	0.6394	0.6277	0.6125	0.6340	0.6967
5	0.6441	0.6389	0.6280	0.6138	0.6390	0.6972
6	0.6446	0.6388	0.6279	0.6140	0.6396	0.6986
7	0.6459	0.6395	0.6279	0.6141	0.6402	0.6989
8	0.6470	0.6397	0.6279	0.6144	0.6422	0.6981
9	0.6470	0.6397	0.6279	0.6144	0.6422	0.6964
10	0.6464	0.6400	0.6279	0.6144	0.6428	0.6961

Table 5.1: Alignment quality results for the IBM2-HMM (2H) Model, its convex relaxation, FastAlign, and the HMM. For mode 2H we decode via dynamic programming using either HMM-style decoding or "Joint" decoding. FA above refers to the improved FastAlign IBM Model 2 model that makes use of a Digamma prior.

Finally, we also tested our model in the full SMT pipeline using the cdec system [17]. For our experiments, we compared our model's alignments (gotten by training $1^5 2 H^5$) against the alignments gotten by the HMM ($1^5 2^5 H^5$), IBM Model 4 ($1^5 2^5 3^5 H^5 4^5$), and FastAlign. Unfortunately, we found that all 4 systems led to roughly the same BLEU score of 40 on a Spanish to English training set of size 250000 which was a subset of version 7 of the Europarl dataset [16]. For our development and test sets, we used data each of size roughly 1800 and we preprocessed all data by considering only sentences of size less than 80 and filtering out sentences which had a very large (or small) ratio of target and source sentence lengths (this is standard for cdec). Although the SMT results did not not produce significant gains, we feel that the experiments at least highlight that our model does not degrade downstream even though the structure of our model is much more local and arguably simpler than either that of the HMM or IBM Model 4.

5.9 Conclusions and Future Work

In this section we have presented some of the details of a new model which combines the structure of IBM Model 2 the alignment HMM model. We've shown that this new model, which has a log-likelihood that can be expressed as a product of terms like the log-likelihood of IBM Model 2, performs about the same as the standard GIZA++ implementation of the HMM. Although the GIZA++ HMM is a celebrated model, bridging the gap between it and convex models proves difficult for a number of reasons, among which the inability to efficiently write out the HMM's log-likelihood. Indeed, although the literature has a plethora of applications for convex optimization, the convex relaxation of the HMM has only been studied though semi-definite programming, and such optimization problems would not be amenable to large datasets like the ones found in SMT [21]. Using the new surrogate, we derived its convex relaxation and showed that the performance of the new model is better than any pervious convex model we studied in Chapters 3 and 4. Moreover, the new convex model performs better than FastAlign [16] and IBM Model 3. Thus, although we do not beat our target goal of the HMM, we do advance the convex models we developed in Chapters 3 and 4 substantially.

Chapter 6

Conclusion

In this work we've looked at several new convex alignments models whose performance is either equal to or very close to some of the non-convex models present in the literature.

The first part of this work looked at the first convex relaxation of IBM Model 2 [8]. The main outcome of our research was two fold as we outlined a model that performed very close to IBM Model 2 and specified an algorithm for its optimization. Several other experiments were conducted and showed that the model can again better F-Measure than IBM Model 2 and, moreover, it could be used as a seed for IBM Model 2; using I2CR-2 as a seed to IBM 2 takes the log-likelihood to another region which might be more natural than the location where IBM Model 1's lexical parameters are located.

Having developed a relaxation for IBM2, the next step in our research was to find a convex surrogate for the alignment HMM model of [43]. While doing this, however, we also discovered a new way to think about relaxations pertaining to IBM Model 2 and we generalized the previous research [33] to [35]. The methods presented in (Simion et al., 2015) [35] can be used to develop multiple relaxation of IBM Model 2. Each of these new relaxations offers a new tradeoff between the lexical t parameters and the distortion d parameters, and we could study each of these new models via the EG algorithm we developed for I2CR-2. However, using a model based on the geometric mean is very advantages because an easy EM algorithm can be developed. The new EM algorithm is just as easy modification to the the EM algorithm for IBM Model 2 and its integration into GIZA++ is seamless. Lastly, as an application of this material, we also looked at a new strictly convex version of IBM Model 1.

The final part of this thesis focuses on a new convex HMM surrogate. The method we use to develop this new model was indirect. To this end, we first discussed a new powerful IBM 2 variant which combined the structure of IBM 2 and the HMM and performs about as well as the HMM. Next, using the methods we developed for IBM Model 2, we study a new convex version of the model. Although the alignment performance of our model is not at the level of the HMM, the SMT experiments are promising and, moreover, the new convex model does better than the popular (and non convex) FastAlign IBM Model 2 variant [16].

Having developed some new convex alignment models, there are several directions where this research might be further pushed. Firstly, we note that we have not provably discovered the best IBM Model 2 relaxation. And, interestingly, it does not seem like the tightest relaxation is the best, so finding this relaxation provably is an open question of interest. Moreover, improving the new convex HMM surrogate to the level of the classical HMM model is an open question. Finally, because the methods we develop are used essentially find a relaxation to a generative probabilistic model, it might be the case that such research can be also applied to other models in other domains such as parsing or word clustering. These questions are not answered yet, but we've made some headway.

Bibliography

[1] Noah A. Smith Andre F. T. Martins and Eric P. Xing. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the EMNLP*, 2010.

[2] John DeNero Aria Haghighi, John Blitzer and Dan Klein. Better word alignments with supervised itg models. In *Proceedings of the ACL*, 2009.

[3] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. In *Operations Research Letters*, 2003.

[4] D. Bertsekas. Nonlinear optimization. Athena Press, 1999.

[5] D. Bertsimas. Optimization over integers. Dynamic Ideas, 2005.

[6] D. Bertsimas and J. Tsitsiklis. Introduction to linear programming. Athena Press, 1997.

[7] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge University Press, 2004.

[8] P. Brown, V. Della-Pietra, S. Della-Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics*, 1993.

[9] J. Brunning. Alignment models and algorithms for statistical machine translation. In *PhD Thesis*. Cambridge University Engineering Department and Jesus College, 2010.

[10] P. Bullen, D. Mitrinovic, and M. Vasic. Means and their inequalities. Springer, 1987.

[11] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL*, 2005.

[12] M. Collins. Statistical machine translation: Ibm models 1 and 2. In *Lecture Notes on NLP*, 2015.

[13] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. 2008.

[14] Abdessamad Echihabi Daniel Marcu, Wei Wang and Kevin Knight. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of the EMNLP*, 2006.

[15] A. Dempster, N. Laird, and D.Rubin. Maximum likelihood from incomplete data via the em algorithm. In *Journal of The Royal Statistical Society*, 1977.

[16] C. Dyer, V. Chahuneau, and N. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL*, 2013.

[17] C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL*, 2010.

[18] A. Fraser and D. Marcu. Measuring word alignment quality for statistical machine translation. In *Journal Computational Linguistics*, 2007.

[19] K. Ganchev, J. V. Graca, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. In *Journal of Machine Learning*, 2010.

[20] J. V. Graca, K. Ganchev, and B. Taskar. Learning tractable word alignment models with complex constraints. In *Computational Linguistics*, 2010.

[21] Y. Guo and D. Schuurmans. Convex relaxations of latent variable training. In *Proceedings of NIPS*, 2007.

[22] S. Kakade. Exponentiated gradient descent. In *Statistical Learning Theory: Lecture Notes*, 2011.

[23] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. In *Information and Computation*, 1997.

[24] P. Koehn. In *Statistical Machine Translation*. Cambridge University Press, 2010.

[25] P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Proceedings of HLT-NAACL*, 2006.

[26] R. Michalcea and T. Pederson. An evaluation exercise in word alignment. In *HLT-NAACL 2003: Workshop in building and using Parallel Texts: Data Driven Machine Translation and Beyond*, 2003.

[27] R. Moore. Improving ibm word-alignment model 1. In *Proceedings of the ACL*, 2004.

[28] F. Och and H. Ney. A systematic comparison of various statistical alignment models. In *Journal of Computational Linguistics*, 2003.

[29] K. Papineni, S. Roukos, T. Ward, and W. Zhu. "bleu: A method for automatic evaluation of machine translation". In *Proceedings of the ACL*, 2002.

[30] Lawrence A Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of IEEE, 1989.

[31] D. Riley and D. Gildea. Impoving the ibm alignment models using variational bayes. In *Proceedings of ACL*, 2012.

[32] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, 1998.

[33] A. Simion, M. Collins, and C. Stein. A convex alternative to ibm model 2. In *Proceedings of EMNLP*, 2013.

[34] A. Simion, M. Collins, and C. Stein. Some experiments with a convex ibm model 2. In *Proceedings of EACL*, 2014.

[35] A. Simion, M. Collins, and C. Stein. A family of latent variable convex relaxations for ibm model 2. In *Proceedings of AAAI*, 2015.

[36] A. Simion, M. Collins, and C. Stein. On a strictly convex ibm model 1. In *submitted*, 2015.

[37] A. Simion, M. Collins, and C. Stein. Towards a convex alignment hmm surrogate. In *submitted*, 2015.

[38] Dan Klein Simon Lacoste-Julien, Ben Taskar and Michael Jordan. Word alignment via quadratic assignment. In *Proceedings of the NAACL*, 2006.

[39] N. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proceedings of ACL*, 2004.

[40] B. Taskar, S. Lacoste-Julien, and D. Klein. A discriminative matching approach to word alignment. In *Proceedings of EMNLP*, 2005.

[41] K. Toutanova and M. Galley. Why initialization matters for ibm model 1: Multiple optima and non-strict convexity. In *Proceedings of ACL*, 2011.

[42] A. Vaswani, L. Huang, and D. Chiang. Smaller alignment models for better translations: Unsupervised word alignment with the $l0$-norm. In *Proceedings of ACL*, 2012.

[43] S. Vogel, H. Ney, and C. Tillman. Hmm-based word alignment in statistical translation. In *Proceedings of COLING*, 1996.

[44] C. Zalinescu. Convex analysis in general vector spaces. World Scientific, 2002.

[45] S. Zhao and D. Gildea. A fast fertility hidden markov model for word alignment using mcmc. In *Proceedings of EMNLP*, 2010.