SEARCH AND DECODING STRATEGIES FOR COMPLEX LEXICAL MODELING IN LVCSR

by

Anoop Deoras

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

July, 2011

Abstract

The language model (LM) in most state-of-the-art large vocabulary continuous speech recognition (LVCSR) systems is still the n-gram. A major reason for using such simple LMs, besides the ease of estimating them from text, is computational complexity.

It is also true, however, that *long-span* LMs, be they due to a higher n-gram order, or because they take syntactic, semantic, discourse and other long-distance dependencies into account, are much more accurate than low-order n-grams. The standard practice is to carry out a first pass of decoding using, say, a 3-gram LM to generate a *lattice*, and to rescore only the hypotheses in the lattice with a higher order LM. But even the search space defined by a lattice is intractable for many long-span LMs. In such cases, only the N-best full-utterance hypotheses from the lattice are extracted for evaluation. However, the N-best lists so produced, tend to be "baised" towards the model producing them, making the re-scoring sub-optimal, especially if the re-scoring model is complementary to the initial n-gram model. For this reason, we seek ways to incorporate information from long-span LMs by searching in a more unbiased search space.

In this thesis, we first present strategies to combine many complex long and short span

ABSTRACT

language models to form a much superior unified model of language. We then show how this unified model of language can be incorporated for re-scoring dense word graphs, using a novel search technique, thus alleviating the necessity of sub-optimal N-best list re-scoring. We also present an approach based on the idea of variational inference, virtue of which, long-span models are efficiently approximated by some tractable but faithful models, allowing for the incorporation of long distance information directly into the first-pass decoding.

We have validated the methods proposed in this thesis on many standard and competitive speech recognition tasks, sometimes outperforming state-of-the-art results. We hope that these methods will be useful for research with long span language models not only in speech recognition but also in other areas of natural language processing such as machine translation, where even there the decoding is limited to n-gram language models.

Advisors: Prof. Frederick Jelinek (deceased), Prof. Kenneth Church

Readers: Prof. Kenneth Church and Prof. Sanjeev Khudanpur

Thesis Committee: Prof. Kenneth Church, Prof. Sanjeev Khudanpur, Prof. Trac Tran, Prof. Mounya Elhilali, Prof. Damianos Karakos and Prof. Aren Jansen

Acknowledgments

I am truly indebted to my advisor, late Prof. Frederick Jelinek, for giving me a chance and opportunity to work with him towards my PhD. Prof. Jelinek arranged for my scholarship by securing funding from the very prestigious Partnership for International Research and Education (PIRE) grant and then later from Human Language Technology - Center of Excellence (HLT-COE). The grant allowed me to visit universities in Europe and collaborate with some of the best researchers in the area of Speech and Language. Apart from many values that I learned from him, the two most important ones were punctuality and respect for everyone. His enthusiasm for learning new stuff, be it a new concept or be it a new technology, was very contegious. I still remember that afternoon, just a few hours before he passed away, when he was fumbling with keyboard keys of his new iMac to find an equivalent of Ctrl C + Ctrl V. When I told him the right keys on his keyboard, he quickly tried it and upon success, jumped with excitement shouting 'Yes ! I learned a new thing today. You made my day.' It was hard to comprehend that a person with so much zeal and passion to learn new things, would leave us just like that.

Prof. Kenneth Church unconditionally took the responsibility of mentoring me after

ACKNOWLEDGMENTS

Prof. Jelinek's death. Prof. Church not only understood the situation but he also gave me enough time to come to terms with it. Prof. Church's deep background in language processing and speech recognition helped me tremendously to shape some of the ideas I was pursuing at the time of transition. His crystal clear views on organizing and presenting the content and material, importance of citing old but important work, among many others, has been extremely useful in not only preparing this work but also in appreciating the importance of the same. His mantra of saying everything 3 times is very appealing. He says that everything has to be said thrice in any research paper or presentation: First you say what you want to say, then you say what you have to say and then you say what you had to say.

Prof. Jelinek had pioneered many ideas at IBM, while Prof. Church had innovated many concepts at AT&T labs and then at MSR. This gave me an opportunity to compare and contrast the solid thinking styles of two giants of this field and learn from it. I am really fortunate to spend so many hours in their offices, sometimes discussing ideas, sometimes refuting (took a lot of courage to do that !) and sometimes simply nodding with a sheer submissive thought.

I am also grateful to Prof. Jansen for spending so much of his time in helping me with the transition after Prof. Jelinek's death. Prof. Karakos and Prof. Khudanpur helped me shape many nascent ideas during many TTO meetings. A truly resourceful person, Prof. Karakos always had answers to my questions and I never had to wait more than a few minutes to receive a reply to my email. I have been lucky to collaborate with Prof. Harper

and one of her students, Denis Filimonov. It was an enriching experience to discuss some of my ideas with her. Her deep background in linguistics gave a different perspective to the problems I was focusing on. Collaboration with Tomáš Mikolov from Brno University of Technology - CZ, turned out to be an extremely productive one and many of the ideas presented in this thesis were possible due to this collaboration.

Substantial and meaningful research was possible due to generous sharing of tools and models by IBM. Brian Kingsbury and Bhuvana Ramabhadran shared with us top-notch software and best statistical models to form a solid and state-of-the-art baseline foundations upon which we tried and tested our own ideas to advance the science of statistical speech recognition.

My student life at the Center for Language and Speech Processing (CLSP) was made enjoyable by my many friends and colleagues. CLSP and ECE staff members especially Desiree Clives, Monique Folk, Justin Martin, Debbie Race, Felicia Roane and Barbara Sullivan helped me with any administrative issues I had. Thanks to them for being so kind.

I am grateful to my parents for providing me with the best opportunities there were. My very close friends from my undergraduate school made me realize my own potential, for which I will always be thankful to them. Finally, I would like to thank my wife, Rucha, for enduring with me the hardship of 5 years of long distance commuting and sacrificing so many long weekends by staying at home with me so that I could run a few more experiments!

Dedication

To Rucha.
In memory of my beloved advisor: Frederick Jelinek (18 Nov. 1932 - 14 Sep. 2010).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Problem

The language model (LM) in most state-of-the-art large vocabulary continuous speech recognition (LVCSR) systems is still the n-gram, which assigns probability to the next word based on only the $n-1$ preceding words. A major reason for using such simple LMs, besides the ease of estimating them from text, is computational complexity. The search space (time) in LVCSR decoding is governed by the number of distinct "equivalent" histories, i.e. the number of unique probability distributions needed to account for all possible histories, and it grows nearly exponentially with n-gram order. So it is customary to limit n to 3 or 4.

It is also true, however, that *long-span* LMs, be they due to a higher n-gram order, or because they take syntactic, semantic, discourse and other long-distance dependencies into account, are much more accurate than low-order n-grams. Long-span LMs therefore are employed when accuracy is a priority. The standard practice is to carry out a first pass of decoding using, say, a 3-gram LM to generate a *lattice*, and to rescore only the hypotheses in the lattice with a higher order LM, such as a 4- or 5-gram. But even the search space defined by a lattice is intractable for many long-span LMs. In such cases, only the N-best full-utterance hypotheses from the lattice are extracted for evaluation by the long-span LM. Typically, N is a few thousand, if not a few hundred.

Using an n-gram for generating the N-best list however comes at a price. The n-gram LM may assign such a low score to good hypotheses that they fail to appear among the N-best. If such hypotheses would have eventually surfaced to the top due to the long-span LM, their loss is attributable to the "bias" of the N-best list towards the first pass LM. For this reason, we seek ways to incorporate information from long-span LMs by searching in a more unbiased search space.

In this thesis, we first present strategies to combine many complex long and short span language models to effectively form a much superior unified model of language. We then propose and demonstrate a novel search technique, to show how this unified model of lan-

guage, capturing local and long distance dependencies, can be incorporated for re-scoring dense word graphs, thus alleviating the necessity of sub-optimal N-best list re-scoring. We also present an approach based on the idea of variational inference, virtue of which, long-span models are efficiently approximated by some tractable but faithful models, allowing for the incorporation of long distance information directly into the first-pass decoding.

In the next two sections, we discuss the main contributions of this thesis followed by the organization of chapters.

1.2 Contribution

The main contribution of the thesis is efficient decoding schemes to solve some of the computationally complex decoding problems occurring in speech recognition systems. While some problems become intractable due to the long span nature of the language model, some become computationally complex due to the nature of the task and the limitation of the decoder. This thesis has contributed by investigating into these decoding issues and propose and present novel solutions. The key contribution of the thesis can be summarized as follows:

- It presents empirical evidence of the fact that by combining many powerful statistical language models, the net performance of the system improves significantly [1]. It also demonstrates an efficient training framework for combining many complex language models for their eventual use in decoding in speech recognition systems [2].
- It discusses a decoding scheme which enables the use of long span language models or for that matter combination of many complex long and short span language models to carry out re-scoring of big word graphs [3; 4], which are the output of first pass recognition systems.
- It presents a variational inference scheme for inferring a tractable model from a long span language model so that it could be used directly in the first pass decoding of speech recognition system [5] thus alleviating the time consuming post processing step.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- **Chapter 2 - Background:** In this chapter, statistical speech recognition pipeline with an emphasis on language models and the impact of their sizes and complexities on the efficiency of decoding (search) is discussed. Some of the very popular

language models and smoothing schemes are described. This chapter also illustrates some of the well known decoding schemes and commonly used search space topology structures. Having described the necessary background, problems central to our thesis are discussed and an outline of the proposed solutions is briefly discussed.

- **Chapter 3 - Model Combination:** The efficacy of model combination for its eventual use in decoding in speech recognition systems is presented in this chapter. Various state-of-the-art statistical language models are combined together and for a text processing task, it is shown that the model combination outperforms the best single model performance. The idea is extended further to demonstrate the efficacy of this method for applications such as automatic speech recognition in which it is shown that the model combination framework not only outperforms single best model result, but also the consensus of independent system outputs i.e. system combination output.

- **Chapter 4 - Iterative Decoding:** Although long span model and the combination of many language models achieve substantial reduction in word errors, their incorporation in speech recognition systems become computationally challenging. To solve this problem, a hill climbing approach for decoding big word graphs is presented in this chapter. Some word graph structures naturally define local neighborhoods while for others they have to be explicitly defined. Once these local neighborhoods are defined, exhaustive search is performed in there and the solution is improved in each successive step by analyzing the neighborhoods one after the other. Although hill climbing methods are quite popular and widely known in general in Artificial Intelligence, the choice of local neighborhood is very crucial and it impacts the efficiency and accuracy of the overall search algorithm. Ideas based on finding near-independent *islands of confusability* to form local neighborhoods are proposed and demonstrated.

- **Chapter 5 - Variational Approximations:** A different take on the problem of incorporating long span and complex language model is discussed in this chapter. By searching for a tractable substitute of complex model such that it satisfies some consistency requirement, the long span models are replaced by these tractable counterparts and first-pass decoding is carried out with them instead. These models are found out such that they are closer to the long span models when the distance is measured using an appropriate metric. Ideas from *variational inference* are used for searching these models.

- **Chapter 6 - Conclusions and Future Directions:** This chapter summarizes the major contributions and findings of this dissertation and concludes with future directions.

Chapter 2

Background

The purpose of this chapter is to provide sufficient background information on various aspects and components of statistical speech recognition so that readers will be able to follow and appreciate the ideas presented in this thesis. We will begin discussing the pipeline of the statistical speech recognition system and then in subsequent sections, we will explore the fundamentals and working of its each component. While some components will not be discussed beyond merely citing the appropriate references, some components will be allocated significant portions of this chapter. We have done so in order to keep the flow of ideas consistent with the overall theme of this thesis, which has got to do with language modeling and its impact on the decoding capabilities of speech recognition.

Automatic Speech Recognition (ASR) is the *art* of verbatim transcription of the audio signal. We are interested in its statistical aspects i.e. the methodology used for speech recognition discussed in this chapter will be based on techniques heavily borrowed from mathematical statistics, machine learning and statistical pattern recognition. While digital signal processing (DSP) is an equally important subject and this science is necessary for extracting useful features from audio signal, which in turn are necessary for pattern recognition algorithms to work well, it is, however, not treated in any detail as it is beyond the scope of this thesis.

2.1 Statistical Learning

In the Maximum-A-Posteriori (MAP) decoding framework for statistical speech recognition, one aims to find the most likely word string, $W^* \in \mathcal{W}$ such that its likelihood given the observed acoustics, $\mathbf{A}$, is maximized. Mathematically it can be written as:

$$W^* = \underset{W \in \mathcal{W}}{\operatorname{argmax}} P(W|\mathbf{A}) \tag{2.1}$$

Using Bayes decomposition rule, and noting that the input acoustics are fixed and given

to us, the above objective function can be rewritten as:

$$W^* = \underbrace{\underset{W \in \mathcal{W}}{\operatorname{argmax}}}_{\text{Search}} \underbrace{P(\mathbf{A}|W)}_{\text{AM}} \underbrace{P(W)}_{\text{LM}} \tag{2.2}$$

where the first probability distribution, $P(\mathbf{A}|W)$, becomes the characterization of an Acoustic Model (AM) while the second probability distribution, $P(W)$, becomes the characterization of a Language Model (LM). The optimal hypothesis is found out by searching (argmax) through all word sequences possible in the language.[1]

Bayes decision theory tells us that such a formulation, under true posterior distribution, will guarantee the minimization of the $1/0$ loss function on an average. For speech recognition systems, $1/0$ loss corresponds to the Sentence Error Rate (SER)[2]. However, the metric we are interested in is Word Error Rate (WER)[3], which is more forgiving than a rather harsh metric of SER. The decoding framework hence needs to be modified such that WER of the system is explicitly minimized (rather than implicitly by way of minimizing SER). The Minimum Bayes Risk (MBR) [6, pp. 161] decoding framework, which guarantees minimization of the general loss (WER for ASR) rather than $1/0$ loss (SER for ASR), aims to find the hypothesis, $W^* \in \mathcal{W}$ such that this hypothesis has the least expected loss when such an expectation is computed under true posterior distribution over a sample space of all word sequences possible in our language. Mathematically it can be written as:

$$W^* = \underset{W' \in \mathcal{W}}{\operatorname{argmin}} E_{P(\mathbf{W}|\mathbf{A})}[L(\mathbf{W}, W')] \tag{2.3}$$

The expectation is computed as follows:

$$E_{P(\mathbf{W}|\mathbf{A})}[L(\mathbf{W}, W')] = \sum_{W \in \mathcal{W}} L(W, W') P(W|\mathbf{A}) \tag{2.4}$$

where the loss function, $L(W, W')$, computes the edit distance of one hypothesis from the other. Note that the choice of $1/0$ loss function for $L(\cdot, \cdot)$, makes Eqn. 2.3 and 2.1 equivalent to each other.

We can re-write the MBR formulation by decomposing the posterior probability using Bayes theorem (similar to what we did for MAP decoding in Equation 2.1):

$$W^* = \underbrace{\underset{W' \in \mathcal{W}}{\operatorname{argmin}}}_{\text{Search}} \frac{1}{Z} \sum_{W \in \mathcal{W}} L(W, W') \underbrace{P(\mathbf{A}|W)}_{\text{AM}} \underbrace{P(W)}_{\text{LM}} \tag{2.5}$$

[1] Of course, appropriate approximations are needed to make the search tractable.

[2] SER is the percentage of speech utterances correctly transcribed by the recognizer.

[3] WER is defined as the total number of minimum word edits (substitution (S), insertion (I) and deletion (D)) required to transform the recognizer hypotheses into true transcripts, divided by the total number of words in reference transcripts (N):

$$\text{WER} = \frac{\#\text{S} + \#\text{D} + \#\text{I}}{\text{N}}$$

where $Z = \sum_{W \in \mathcal{W}} P(\mathbf{A}|W)P(W)$

In the next few sections, we will discuss each component of the MAP / MBR decoding framework, starting with Acoustic Model (concerning with $P(\mathbf{A}|W)$), followed by Language Model (concerning with $P(W)$) and finally concluded by search (concerning with argmax or argmin depending upon the decoding paradigm).

2.2 Acoustic Model

In this section, we will very briefly discuss a method to construct acoustic model that the recognizer may use to compute the probability $P(\mathbf{A}|W)$ for any acoustic string $\mathbf{A} = a_1, \ldots, a_m$ and hypothesized word string $W = w_1, \ldots, w_n$. The acoustic model in state-of-the-art recognizers today is based on hidden Markov model (HMM) concepts (see Rabiner's paper [7] for a detailed discussion about HMMs and their applications in speech recogniton). Each basic sound in an LVCSR system is represented by a HMM. It consists of sequence of hidden states connected to each other by probabilistic transitions. A transition to the same state or to some other state leads to the production of acoustics which are observed. This is modeled by the emission probability distribution. The transition probabilities model the durational variability in real speech while the emission probabilities model the spectral variability [8].

HMM phone models consists of three left to right states (also called as Bakis model [9]). The first and last state of the phone model are entry and exit states respectively. The exit state can be *glued* to the entry state of another phone for forming word models. These states are typically non-emitting since their sole purpose is to glue the models together. If we represent a set of internal states by s_i, outputting an acoustic vector $\mathbf{o}_j$, then the likelihood of such an emission, $l_i(\mathbf{o}_j)$, is modeled using Gaussian Mixture Models (GMM):

$$l_i(\mathbf{o}_j) = \sum_{m=1}^{M} \lambda_{i,m} \mathcal{N}(\mathbf{o}_j; \mu_{i,m}, \Sigma_{i,m})$$

where $\lambda_{i,m}$ is the mixture weight associated with the m^{th} gaussian of the GMM model of the i^{th} state.

Typically context dependent phone models are learned instead of context independent phone models. Tri-phone model and/or quin-phone models are considered to be the state-of-the-art. Tri-phone models take into account the effect of one neighboring phone on the left and right of the phone to be modeled, while quin-phone models take into account two neighboring phones. However, even for a small inventory of phones, say 45, the total number of possible tri-phones turn out to be huge ($= 45^3$). Since each tri-phone needs to be modeled by a GMM with possibly hundreds of mixtures, the total number of parameters to be trained blows up. Decision tree based state clustering[4]is hence employed to remove the redundancy and cluster tri-phones. This technique is called as *state tying*, in which

[4]State is context dependent phone structure.

states which are acoustically indistinguishable (or have the same acoustic properties) are tied together. This allows all the data associated with each individual state to be pooled and thereby gives more robust estimates for the parameters of the tied-state. Readers are encouraged to read Steve Young's paper on acoustic modeling for LVCSR [8], for thorough details.

2.3 Language Model

A language model assigns probability distribution over word string, W, possible in our language, $\mathcal{W}$. If we think of the word string as composed of sequence of words: $W \equiv w_1, \ldots, w_m$, then the probability of the word string can be obtained using the chain rule as follows:

$$P(W) = \prod_{i=1}^{m} P(w_i | w_{i-1}, \ldots, w_1). \tag{2.6}$$

where $P(w_i|w_{i-1}, \ldots, w_1)$ is the probability that w_i will be spoken given that words $w_1, \ldots, w_{i-1}$ were previously said. The past $w_1, \ldots, w_{i-1}$ is frequently referred to as *history* and will be denoted by $\mathbf{h}_i$. The recognizer thus should be able to determine the estimates of these probabilities. However, even for a reasonably sized vocabulary, $\mathcal{V}$, the number of arguments for $P(w_i|w_1, \ldots, w_{i-1})$ turns out to be $|\mathcal{V}|^i$. To put it in perspective, for a system with vocabulary size $20,000$ and $i = 3$, the number of arguments is close to 8×10^{12}. This is just prohibitively large and in any real system, we won't ever have enough training data to robustly estimate these many probability values. Hence the history is mapped to an equivalence class using 'many to one' mapping function. This facilitates reduction in the number of arguments. If we represent the equivalence mapping function by $\Phi(\cdot)$, then the probability of the word string can be obtained as:

$$P(W) = \prod_{i=1}^{m} P(w_i | \Phi(w_{i-1}, \ldots, w_1)) \tag{2.7}$$

The art of Language Modeling then consists of determining the properties of these equivalence classes and ways of estimating them.

When the equivalence mapping maps the history $\mathbf{h}_i$ to previous $n-1$ words, the LM is called as an n-gram Language Model. The n-gram LM is discussed in subsection 2.3.1. The LM in which the choice of equivalence class is based on the identity of the leaf which in turn is reached by percolating the decision tree after asking series of questions about the history (truncated), is called a Decision Tree Language Model (DTLM). Ensembles of many *randomly* grown decision trees form Random Forest Language Model (RFLM). DTLM and RFLM is discussed in subsection 2.3.2. The choice of an equivalence class to which the history belongs, fulfills the requirement of having fewer parameters to estimate, which becomes possible from the limited data one has access to. In Maximum Entropy based LM (MELM), a joint probability model, $P(w, \mathbf{h})$, is estimated such that it not only

satisfies certain linear constraints but also organizes itself in all other respects in accordance with our ignorance about everything these constraints do not specify [10, pp. 219]. MELM is discussed in subsection 2.3.3.

Data sparseness can also be combated by forming equivalence classes of words. Class based language models cluster the words either using syntactic information such as Part-of-Speech tags or using statistics derived from the training data itself. Class based LMs are discussed in subsection 2.3.4.

Another way of fighting data sparsity is to first map each word to a vector of dimension, say, $d(\ll |\mathcal{V}|)$, of continuous features, and then learn the probability of various outputs as a function of these continuous features. Neural network based language models project words into a lower dimensional continuous space before learning probability distributions of words given the context. Feed Forward Neural Network based LM (FFNNLM) truncates the history to $n-1$ words, similar to n-gram LM, while Recurrent Neural Network based LM (RNNLM) maps the entire context into the continuous space. Neural Network based LMs are discussed in subsection 2.3.6.

RFLM, MELM and FFNNLM, like n-gram LM, utilize only n-gram features for prediction and hence do not capture longer dependencies.[5] While RNNLM uses only the statistics derived from the data for estimating the probability distributions capturing longer dependencies, parser based syntactical language models capture these long span dependencies by parsing the input sentence and finding syntactical dependencies such as between subject and its direct or indirect objects, among others. Structured (syntactic) LMs are discussed in subsection 2.3.5.

There are many other novel LMs introduced to the speech community, however, in our thesis, we have considered and used only the above mentioned ones and hence in this chapter we discuss only these. We refer interested readers to Goodman's survey on language modeling techniques [11] for more details.

2.3.1 n-gram LM

In this form of language model, the context or the history is truncated to contain only the previous $n-1$ words i.e. the equivalence classification on the history maps the context to previous $n-1$ words. The probability of a word sequence is then given as:

$$P(W) = \prod_{i=1}^{m} P(w_i|\Phi(w_{i-1},\ldots,w_1)) = \prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1}) \tag{2.8}$$

The probability values are estimated using the maximum likelihood principle i.e. a non-parametric model is fit such that the it maximizes the likelihood of the training data. If we let $C(w_{i-n+1},\ldots,w_{i-1})$ represent the number of occurrences of $w_{i-n+1},\ldots,w_{i-1}$

[5] In principle, one can use the entire context for predicting the next word, however, the number of parameters in these LMs grow proportionately with the size of the context. Due to data sparsity, these LMs hence use only truncated context for prediction.

in the training data and similarly for $C(w_{i-n+1}, \ldots, w_i)$, then the probability estimate of $P(w_i|w_{i-n+1}^{i-1})$ is found out as:

$$P(w_i|w_{i-n+1}^{i-1}) \approx \frac{C(w_{i-n+1}, \ldots, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1})} \tag{2.9}$$

Even for small values of n, say 3, it is only obvious that we won't be able to find enough samples (sometimes any sample) in our training corpus for some n-tuple, leading to poor estimates. To avoid assigning 0 probability or extremely low probability to events not seen in the training data or seen very few times, probability estimates need to be smoothed. Good Turing smoothing and Kneser-Ney smoothing are two popular smoothing methods, among few others, which smooth the probability distribution by taking some probability mass away from some events (occurring frequently) and distributing it to events which either occur very rarely or do not occur at all. In subsections 2.3.1.1 and 2.3.1.2, we will briefly discuss the mathematical formulation of these two smoothing methods. We refer interested readers to a very thorough survey on various smoothing techniques by Chen and Goodman [12].

2.3.1.1 Good Turing Smoothing

The Good Turing estimate [13] states that for any n-gram that occurs r times, we should *discount* it by pretending that it occurs r^* times:

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{2.10}$$

where n_r is the number of n-grams such that they occur r times in the training data. In LM, the discounted count, r^*, will always be less than r.[6] This will leave a certain amount of probability mass for rare or unseen events. Letting N represent the total size of the training data, the left-over probability mass will be equal to $\frac{n_1}{N}$; this represents the amount of probability to be allocated for events that were *never* seen. Goodman quotes in [11]: 'This is really quite an amazing and generally useful fact, that we can predict how often we expect something to happen that has never happened before, by looking at the proportion of things that have occurred once'.

Katz smoothing [14] is based on Good Turing discount formula and in this model, the probability of a word given the context of previous $n-1$ words is given as:

$$P_{\text{Katz}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{C^*(w_{i-n+1},\ldots,w_i)}{C(w_{i-n+1}^{i-1})} & \text{if } C(w_{i-n+1}^{i}) > 0 \\ \alpha(w_{i-n+1}^{i-1}) \times P_{\text{Katz}}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \end{cases} \tag{2.11}$$

[6] However, the only exception is when the n-gram has zero-counts in which case, a non-zero mass is assigned to it.

where $C^*(\cdot)$ represents the discounted count according to the Good Turing formula given in Eqn. 2.10.

There is another well known smoothing method by Church et.al. [15], which, like Katz's method, combines the Good-Turing estimate with a method for merging the information from lower and higher order models.

2.3.1.2 Kneser-Ney Smoothing

Backoff Kneser-Ney (BKN) smoothing [16; 17] and its variants such as interpolated (IKN) and modified Kneser-Ney (MKN) smoothing, are considered to be the state-of-the-art in smoothing methods for n-gram LMs [12]. The difference between Katz smoothing and Backoff Kneser-Ney smoothing method is that while the former uses number of *occurrences* of the backed off terms for computing backoff probability mass, the latter smoothing method uses number of *contexts* the backed off terms occur with. Goodman [11] illustrates the importance of this change with an example. Let us say that we want to estimate the probability of word *Francisco* given the word *on* using a bigram model. Since the word *San Francisco* is very common, the unigram probability $\frac{C(Francisco)}{\sum_w C(w)}$ will also be fairly high. The Katz smoothing model will then assign the probability for *on Francisco* according to following formula:

$$P_{\text{Katz}}(Francisco|on) = \begin{cases} \frac{C^*(onFrancisco)}{C(on)} & \text{if } C(onFrancisco) > 0 \\ \alpha(on) \times P_{\text{Katz}}(Francisco) & \text{otherwise} \end{cases} \tag{2.12}$$

If *on Francisco* does not occur at all in the training data, then the Katz model will end up giving it a very high probability due to high unigram probability on the word *Francisco*. Backoff Kneser Ney model corrects this by modifying the backoff distribution to depend upon the number of contexts the backoff word occurs with. Also, instead of using Good Turing discount formula, Kneser Ney formula uses an absolute discounting term, D. The use of absolute discounting for smoothing was first described by Ney et.al. in [18]. The bigram probability of a word given a context of one previous word using Backoff Kneser-Ney (BKN) method is given as:

$$P_{\text{BKN}}(w_i|w_{i-1}) = \begin{cases} \frac{\max\left(C(w_{i-1},w_i)-D,0\right)}{C(w_{i-1})} & \text{if } C(w_{i-1}w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{|\{v|C(vw_i)>0\}|}{\sum_w |\{v|C(vw)>0\}|} & \text{otherwise} \end{cases} \tag{2.13}$$

Chen and Goodman [12] proposed *Interpolated* Kneser-Ney smoothing method, in which instead of backing off to lower order distribution, the higher-order and lower order distributions are interpolated. The formula for bigram probability of word given the context of one previous word using Interpolated Kneser-Ney (IKN) method is given below:

$$P_{\text{IKN}}(w_i|w_{i-1}) = \frac{\max\big(C(w_{i-1}, w_i) - D, 0\big)}{C(w_{i-1})} + \lambda(w_{i-1})\frac{|\{v|C(vw_i) > 0\}|}{\sum_w |\{v|C(vw) > 0\}|} \tag{2.14}$$

where $\lambda(w_{i-1})$ is a normalization constant such that the left hand side is a valid probability distribution, i.e. it sums to 1. The Interpolated Kneser-Ney smoothing formula for n-gram probability of a word given the context of previous $n-1$ words under this smoothing scheme is given by:

$$P_{\text{IKN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\big(C(w_{i-n+1}^{i}) - D, 0\big)}{\sum_{w_i} C(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1})P_{\text{IKN}}(w_i|w_{i-n+2}^{i-1}) \tag{2.15}$$

where $\gamma(w_{i-n+1}^{i-1})$ makes sure that $P_{\text{IKN}}(w_i|w_{i-n+1}^{i-1})$ is a valid probability distribution. Its value is given by:

$$\gamma(w_{i-n+1}^{i-1}) = \frac{DN_{1+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} C(w_{i-n+1}^{i})} \tag{2.16}$$

where $N_{1+}(w\bullet) = |\{v|C(wv) > 0\}|$.

Chen and Goodman [12] proposed one more modification to Kneser-Ney smoothing. They proposed the use of multiple discounts: one discount dedicated for counts with value 1, another dedicated for counts with value 2 and a third one dedicated for counts with value 3 or more. They call it Modified Kneser-Ney (MKN) smoothing. The n-gram probability of a word given the context of previous $n-1$ words under this smoothing scheme is given by:

$$P_{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\big(C(w_{i-n+1}^{i}) - D\big(C(w_{i-n+1}^{i})\big), 0\big)}{\sum_{w_i} C(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1})P_{\text{MKN}}(w_i|w_{i-n+2}^{i-1}) \tag{2.17}$$

where:

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_3 & \text{if } c \geq 3 \end{cases} \tag{2.18}$$

and where $\gamma(w_{i-n+1}^{i-1})$ makes sure that $P_{\text{MKN}}(w_i|w_{i-n+1}^{i-1})$ is a valid probability distribution. Its value is given as:

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1N_1(w_{i-n+1}^{i-1}\bullet) + D_2N_2(w_{i-n+1}^{i-1}\bullet) + D_3N_3(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} C(w_{i-n+1}^{i})} \tag{2.19}$$

where $N_i(w\cdot) = |\{v|C(wv) > i\}|$. The bi-gram probability under Modified Kneser-Ney smoothing follows the same intuition as applied in the Interpolated Kneser-Ney smoothing in Eqn. 2.14, except of course, this version uses multiple discount values depending upon the number of counts of w_{i-1}, w_i. In our thesis, we have extensively used Modified Kneser Ney smoothing for n-gram models. Ney et.al. [18] proposed setting the discounting factor D through deleted estimation on the training data. They arrived at the following estimate:

$$D = \frac{n_1}{n_1 + 2n_2} \tag{2.20}$$

Chen and Goodman [12] extended this idea to derive formula for discount parameters for the Modified Kneser-Ney smoothing as follows:

$$D_1 = 1 - 2D\frac{n_2}{n_1}$$
$$D_2 = 2 - 3D\frac{n_3}{n_2}$$
$$D_3 = 3 - 4D\frac{n_4}{n_3}$$

where the value of D is found out using Eqn. 2.20.

2.3.2 Random Forest LM

A Random Forest Language Model (RFLM) proposed by Xu et.al. [19] is a collection of randomized Decision Tree Language Models (DTLMs), which define equivalence classification of histories. The RFLM generalizes the DTLM by averaging multiple DTLMs, which, in turn, generalizes the n-gram LM by having a sophisticated equivalence classification. The LM probability of a RFLM is defined as follows:

$$\begin{aligned} P_{RF}(w_i|\mathbf{h}_i) = & \quad \tfrac{1}{M}\textstyle\sum_{j=1}^{M} P_{DT_j}(w_i|\mathbf{h}_i) \\ = & \quad \tfrac{1}{M}\textstyle\sum_{j=1}^{M} P(w_i|\Phi_{DT_j}(\mathbf{h}_i)), \end{aligned} \tag{2.21}$$

where h is the history and $\Phi_{DT_j}(\cdot)$ is a decision tree. The questions that have been used so far care only about the identity of the words in a history position. If w_i is the word we want to predict, then the general question takes the following form:
Is the word $w_{i-k}, k \geq 1$ in a set of words $\mathcal{S}$?
Because in the normal n-gram situation we know no more than the words in the history, these questions are almost all we can ask.

The decision tree training procedure proposed in [19], consists of a *growing* and a *pruning* stage. The growing stage starts from a single node which contains all n-gram histories from the training text. Exchange Algorithm [20] is then used to recursively split every node until further splitting does not decrease the perplexity of this training text. In

the pruning stage, the *potential* of each node is computed. The potential is defined as the possible increase in held out data likelihood from growing the node into a sub-tree. Nodes whose potential fall below a threshold are pruned, bottom up. Thus the structure of the tree (i.e. the depth of the tree, number of nodes, number of leaves) and the parameters of the tree (questions), are tuned to optimize two objective functions: perplexity of the training data and perplexity of the held out data.

Due to the greediness of the exchange algorithm, restricting the candidate questions at each node-splitting step to a *random subset* of all available questions helps find a better tree. The same argument holds for random initialization of the exchange algorithm.

Xu et.al. [19] used Interpolated Kneser-Ney (IKN) smoothing, to compute the LM probabilities[7]:

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{\max(C(\Phi(w_{i-n+1}^{i-1}),w_i)-D,0)}{C(\Phi(w_{i-n+1}^{i-1}))} + \lambda(\Phi(w_{i-n+1}^{i-1}))P_{KN}(w_i|w_{i-n+2}^{i-1}), \quad (2.22)$$

where D is a *constant discount* and $P_{KN}(\cdot)$ is the Backoff Kneser-Ney (BKN) probability distribution (computed using Equation 2.13) for lower order model.

Recently, we showed that RFLMs are also very effective in adapting language models [21]. In this work it was shown, on a moderately sized task, that if we grow the decision trees on huge amounts of out of domain data and then prune them such that the likelihood of in domain data is maximized, then the decision trees get adapted sufficiently enough to outperform n-gram LM interpolation and count merging adaptation [22] techniques.

2.3.3 Maximum Entropy LM

Maximum Entropy models [23], first introduced by Rosenfeld for language modeling [24] in the form of Maximum Entropy Language Model (MELM), are exponential models allowing easy inclusion of number of interesting features within a unified framework. The underlying probability distribution of the LM is found out such that it has the maximum entropy while satisfying some constraints. MELMs, till now, have mainly used only n-gram features, however, features concerning with topic and syntax can also be easily incorporated.[8] The probability distribution should satisfy the linear constraint that the expectation of some feature under this distribution should match the feature's empirical expectation as estimated from the training data.[9] The probability of a word w_i given the context $\mathbf{h}_i$ is given as follows:

[7] The authors also used random forests for backoff lower order estimation, but turns out that an n-gram based backoff model works better.

[8] Rosenfeld in fact used *triggers*, which can be thought of as collocation word pairs, in the maximum entropy framework. Wu & Khudanpur [25] and Khudanpur & Wu [26] used topic information in the maximum entropy framework for estimating a language model. Recently Xu et.al. [27] used MELM framework for estimating a discriminative language model using semi-supervised techniques.

[9] It is not hard to see that maximizing the entropy while satisfying the linear constraints such as to do with the expectation of feature, results in the the probability distribution of an exponential form.

$$\begin{aligned} P(w_i|\mathbf{h}_i;\Theta) &= \frac{1}{Z(\Theta,\mathbf{h}_i)}\prod_{j=1}^{J}\exp^{\theta_j\phi_j(\mathbf{h}_i,w_i)} \\ &= \frac{1}{Z(\Theta,\mathbf{h}_i)}\exp^{\sum_{j=1}^{J}\theta_j\phi_j(\mathbf{h}_i,w_i)} \end{aligned} \tag{2.23}$$

where Z is the normalization constant for $\mathbf{h}_i$, J is the total number of features, $\Theta = (\theta_1,\ldots,\theta_J)$ are the parameters to be estimated and ϕ_j are the features.[10] For instance, a feature for an unigram *a* will be of the following kind:

$$\phi_a(\mathbf{h},w) = \begin{cases} 1 & \text{if } w = a \\ 0 & \text{otherwise;} \end{cases} \tag{2.24}$$

Similarly, feature for trigram *a b c* will be of following type:

$$\phi_{abc}(\mathbf{h},w) = \begin{cases} 1 & \text{if } w = c \text{ and } \mathbf{h} \text{ ends in } a\ b \\ 0 & \text{otherwise;} \end{cases} \tag{2.25}$$

The parameters of the MELM model (Eqn. 2.23) are estimated using maximum likelihood techniques i.e. the parameters are found out such that the resulting probability distribution maximizes the likelihood of the training data[11]:

$$\begin{aligned} \Theta^* &= \operatorname*{argmax}_{\Theta} \log\prod_{i=1}^{m} P(w_i|\mathbf{h}_i;\Theta) \\ &= \operatorname*{argmax}_{\Theta} \sum_{i=1}^{m} \log P(w_i|\mathbf{h}_i;\Theta) \end{aligned} \tag{2.26}$$

where the training data consists of m words: $w_1,\ldots,w_m$. Typically generalized iterative scaling algorithm [23] is used for optimization. However, training of maximum entropy models is extremely slow and this is the main reason why this model has not become popular in-spite of its many convenient features.[12]

[10] These features although in principle can take values in $\mathbb{R}$, they are restricted to be of binary kind for the ease of training.

[11] Duality theory presented by Darroch et.al. [23] states that the distribution of the form of (2.23) maximizing the likelihood of the training data is equivalent to a distribution (without constraining it to any parametric family) with maximum entropy such that it satisfies some linear constraints concerning with the equality of expectation of features under this distribution with the empirical estimate computed on the same training data.

[12] Wu [28] proposed an optimization trick to speed up the training procedure, however even with this optimization, the overall training time remains considerably slower. It is worthwhile to mention the recent work of Xu, Gunawardana and Khudanpur [29] in which authors replaced the probability distribution over the whole vocabulary by combination of several 'word v/s rest' distributions. Considerable speed up is reported when substantial negative samples are weeded out in training each binary classifier. This pruning, leading to considerable speedups, is shown to have no negative effect on performance.

2.3.4 Class Based LM

As previously said, another way of fighting data sparsity is to form equivalence classes of words i.e. cluster the words together. Class based LMs were first described by Brown et.al. [30] in which the authors employed a bottom-up approach to form classes i.e. initially as many classes are formed as there are words in the vocabulary. At each step classes are merged to form a super class such that the merging results in the minimum loss of likelihood compared to the previous classing. In its simplest form, class based trigram word probability decomposes as follows:

$$P(w_i|w_{i-1}, w_{i-2}) \approx P(w_i|g(w_i)) \times P(g(w_i)|g(w_{i-1}), g(w_{i-2})) \tag{2.27}$$

where $g(\cdot)$ is the function which maps words to their deterministic class labels. At any step of the class formation procedure, the above decomposition is used to obtain the likelihood of the training data, which then guides the formation of the classes for words. Note that while the class mapping information (range of function $g(\cdot)$) changes from one step to another during the training procedure, it converges to some local optimal configuration and is then kept fixed during the evaluation phase. Also note that we have used approximation sign in the equation above because some conditional independence assumptions are used to simplify the decomposition. Martin et.al. [20] proposed a different approach for forming classes while keeping the class based formulation same as that in Brown's approach i.e. the probability decomposition of (2.27) remained same in Martin et.al. approach too. However, unlike Brown et.al.'s approach of bottom up clustering, Martin et.al.'s approach uses an exchange algorithm [31] (which is in some sense a top-down approach). Algorithm (1) describes the exchange algorithm for forming classes.

Algorithm 1 Exchange Algorithm

Require: Total number of classes
  Start with initial mapping $\mathcal{M} : V \rightarrow C$
  for each iteration **do**
    for each word $w \in V$ **do**
      move w from $g(w)$ tentatively to all classes
    end for
    Note the class, C^*, for which the likelihood is locally maximum
    set $g(w)$ to C^*
  end for

Emami et.al.'s class based language model [32] can be thought of as the generalization of Martin et.al.'s class based LM. In Emami et.al.'s proposed LM, instead of using fixed class mapping, the authors used different mapping depending upon the position of the word in the n-gram tuple. Under this scheme, the class based word trigram probability decomposes as follows:

$$P(w_i|w_{i-1}, w_{i-2}) \approx P(w_i|g_{(0)}(w_i)) \times P(g_{(0)}(w_i)|g_{(1)}(w_{i-1}), g_{(2)}(w_{i-2})) \tag{2.28}$$

where $g_{(i)}(\cdot)$ is the class mapping function for the i^{th} position in the trigram tuple.[13]

Algorithm (2) describes the modified exchange algorithm proposed by Emami et.al.

Algorithm 2 Generalized Exchange Algorithm

Require: Total number of classes for *each* position
  Start with initial mapping $\mathcal{M}_{(i)} : V_{(i)} \rightarrow C_{(i)}$, $i = 0, 1, 2$
  for each iteration **do**
    for each word position $i = 0, 1, 2$ **do**
      for each word $w \in V_{(i)}$ **do**
        move w from $g_{(i)}(w)$ tentatively to all classes in $C_{(i)}$
      end for
      Note the class, $C^*_{(i)}$, for which the likelihood is locally maximum
      set $g_{(i)}(w)$ to $C^*_{(i)}$
    end for
  end for

2.3.5 Syntactic LM

The first successful language model employing syntax was demonstrated by Chelba and Jelinek [33]. Authors made use of dependency parsers to get the exposed head words which then were used as conditioning terms for predicting the next word. The model assigns a probability $P(W, T)$ to every sentence W and every possible binary parse T. The terminals of T are the words of W with part of speech (POS) tags, and the nodes of T are annotated with phrase headwords and non-terminal labels. The joint probability can be broken up as follows:

$$P(W,T) = \prod_{k=1}^{n+1} \Big[P(w_k|W_{k-1}T_{k-1}) \times P(t_k|W_{k-1}T_{k-1}, w_k) \times \prod_{i=1}^{N_k} P(p_i^k|W_{k-1}T_{k-1}, w_k, t_k, p_1^k \ldots p_{i-1}^k) \Big] \quad (2.29)$$

where:

- $W_{k-1}T_{k-1}$ is the word phrase $(k-1)-$prefix.
- w_k is the word predicted by `WORD-PREDICTOR`
- t_k is the tag assigned to w_k by the `TAGGER`

[13] 0^{th} position corresponds to the predicted word, while the first and second position corresponds to the most recent and second most recent word in the history respectively.

- $N_k - 1$ is the number of operations the CONSTRUCTOR executes at sentence position k before passing control to the WORD-PREDICTOR.
- p_i^k denotes the i^{th} CONSTRUCTOR operation carried out at position k in the word string.

The language model probability of a word string is then obtained by summing over all the binary parses:

$$P(W) = \sum_T P(W, T) \tag{2.30}$$

Several conditional independence assumptions were invoked to simplify Eqn. 2.29. Readers are encouraged to refer to [34; 35; 33] for a thorough discussion on initial attempts to put syntax into language modeling.

These models are also called as *joint* models since they essentially predict joint events of words and some random variable(s). While model of Chelba et.al. used POS tags in combination with parser instructions for constructing a full parse tree in a left-to-right manner, model of Wang et.al. [36] and Filimonov et.al. [37] used SuperARVs and other fine grained tags (they call them *heads* and *parent*) capturing dependency and constituency information, without actually parsing the sentence, thus called *almost parsing*. In the models of Wang et.al. and Filimonov et.al. a random variable is associated with every surface form word and hence to obtain the probability of the next word, one needs to sum over all the assignments of the stochastic variable as shown below:

$$\begin{aligned} P(w_i|w_1^{i-1}) &= \sum_{t_1,\ldots,t_i} P(w_i t_i|w_1^{i-1} t_1^{i-1}) \\ &= \frac{\sum_{t_1,\ldots,t_i} P(w_i t_i|w_1^{i-1} t_1^{i-1}) P(w_1^{i-1} t_1^{i-1})}{\sum_{t_1,\ldots,t_{i-1}} P(w_1^{i-1} t_1^{i-1})} \end{aligned} \tag{2.31}$$

Of course, again, many conditional independence assumptions need to be invoked in order to simplify the probability formulation of Eqn. 2.31. Readers are encouraged to refer to [36; 37] for details on syntactical language model involving the above mentioned fine grained syntactic tags.

2.3.6 Neural Network LM

There is a long history of using neural networks to model sequences. Elman et.al. [38] used recurrent neural network for modeling sentences of words generated by an artificial grammar. Work on statistical language modeling of real natural language data, together with an empirical comparison of performance to standard techniques was done by Bengio et.al. in [39]. His work has been followed by Schwenk et.al. [40], who has shown that neural network language models actually work very well in the state-of-the-art speech

recognition systems. The underlying neural network of the LM proposed by Schwenk et.al. was formulated as a feedforward network. Mikolov et.al. proposed its generalized version in the form of recurrent neural network LM (RNNLM) [41]. The main idea behind neural network LMs is to form a distributed representation of words of language. This enables projecting words into a continuous space where generalization becomes much more easier thus alleviating the necessity of smoothing. While feedforward neural network is still an n-gram kind of language model[14], recurrent neural network language model generalizes this form by projecting the entire context to continuous space. Below, we will describe both forms of neural network LMs.

2.3.6.1 Feedforward Neural Network LM

The original feedforward neural network LM proposed by Bengio et.al. has 3 layers. The first is the input layer, the second is the hidden layer and the third is the output layer. The probability of a word, w_i, given the context of previous $n-1$ words, $w_{i-1}, \ldots, w_{i-n+1}$, is obtained in the output layer at the index corresponding to that of the predicted word, after normalizing the values of output layer. The exact procedure of how information is propagated from one layer to another is described below.

All words of the vocabulary are given unique indices starting with 1 and ending with $|\mathcal{V}|$, where $\mathcal{V}$ is the vocabulary. The words of the context (history) are then represented as 1 of $|\mathcal{V}|$ coding vector. This vector has 0s in all position except one, which corresponds to the index of the word, where the value is 1. These vectors are then linearly projected onto a lower dimension of size m. The linear transformation can be done by multiplying the word feature vector by a $|\mathcal{V}| \times m$ matrix $\mathbf{C}$. This matrix is shared across all the positions of n-gram tuple for all n-grams. For any word $w \in \mathcal{V}$, we will represent this lower dimensional representation by $\mathbf{C}(w)$. Note that while w is represented by a $|\mathcal{V}| \times 1$ vector, $\mathbf{C}(w)$ is represented by a $m \times 1$ vector ($m \ll |\mathcal{V}|$). There is no non-linearity in this projection. The concatenation of projected vectors for each word of the context i.e. $[\mathbf{C}(w_{i-1})^T \cdot \mathbf{C}(w_{i-2})^T \cdot \ldots \cdot \mathbf{C}(w_{i-n+1})^T]^T$, represented hereafter by $\mathbf{x}$, is then transformed linearly by multiplying it with a $(n-1)m \times h$ matrix, $\mathbf{H}$. This projected vector becomes input to the hidden layer of size h. A non-linearity in the form of $\tanh(\cdot)$ is then applied to each element of this vector. The resulting output is then linearly transformed again by multiplying it with a $h \times |\mathcal{V}|$ matrix $\mathbf{U}$. This forms the output layer and a softmax on the output layer gives the normalized probability distribution over words given the context. If we represent the output layer by $\mathbf{y}$, then the above operations can be compactly represented as:

$$\mathbf{y} = \mathbf{b} + \mathbf{U} \tanh(\mathbf{d} + \mathbf{H}\mathbf{x}) \tag{2.32}$$

where $\mathbf{b}$ and $\mathbf{d}$ are the bias terms.

[14] In the conventional n-gram LM of type discussed in Sec. 2.3.1, the probabilities are function of discrete events, while in the n-gram LM of the feedforward neural network LM, the probabilities are function of continuous variables.

The probability of a word w_i given the context is then given by:

$$P(w_i|w_{i-1},\ldots,w_{i-n+1}) = \frac{e^{y_{w_i}}}{\sum_j e^{y_{w_j}}} \tag{2.33}$$

where y_{w_j} is the value in the output layer, $\mathbf{y}$, at the position equal to index of the word w_j.

The trainable parameter set, Θ, are the entries of the bias vectors $\mathbf{b}$, $\mathbf{d}$ and elements of matrices, $\mathbf{H}$ and $\mathbf{U}$:

$$\Theta = \{\mathbf{b}, \mathbf{d}, \mathbf{H}, \mathbf{U}\}$$

Stochastic gradient ascent on the neural network is performed by following iterative updates after presenting the i^{th} word of the training corpus:

$$\Theta \leftarrow \Theta + \epsilon \frac{\partial \log P(w_i|w_{i-1}^{i-n+1})}{\partial \Theta}$$

where $P(w_i|w_{i-1}^{i-n+1})$ is obtained using equation 2.33 and ϵ is the learning rate. Usually backpropagation algorithm is used to find out the gradients.

2.3.6.2 Recurrent Neural Network LM

Recurrent neural network based language models (RNN-LMs) [41] improved the ability of the original model i.e. feedforward neural network LM, to capture patterns in the language without using any additional features (such as part of speech, morphology etc) i.e. other than lexical ones. The RNN-LM was shown to have superior performance than the original feedforward neural network and many advanced language modeling techniques [41]. We hence decided to work with this model in our thesis. This model uses whole history to make predictions, thus it lies outside the family of n-gram models. Power of the model comes at a considerable computational cost. Due to the requirement of unlimited history, many optimization tricks for rescoring with feedforward-based NNLMs as presented by [40] cannot be applied during rescoring with RNN LM. Thus, this model turns out to be a good candidate to show effectiveness and importance of our work.

The basic RNN-LM is shown in Fig. 2.1. The network has an input layer $\mathbf{x}$, a hidden layer $\mathbf{s}$ (also called state or context layer) and an output layer $\mathbf{y}$. Input to the network at time t is denoted $\mathbf{x}(t)$, output $\mathbf{y}(t)$, and the hidden state $\mathbf{s}(t)$. The input $\mathbf{x}(t)$ is formed by concatenating a vector $\mathbf{w}(t)$ that represents the current word while using 1 of $|\mathcal{V}|$ coding (thus the size is equal to the size of the vocabulary $\mathcal{V}$) and vector $\mathbf{s}(t-1)$ that represents output values in the hidden layer from the previous time step. Values in these layers are computed as follows:

$$\mathbf{x}(t) = [\mathbf{w}(t)^T \mathbf{s}(t-1)^T]^T \tag{2.34}$$

$$s_j(t) = f\Big(\sum_i x_i(t) u_{ji}\Big) \tag{2.35}$$

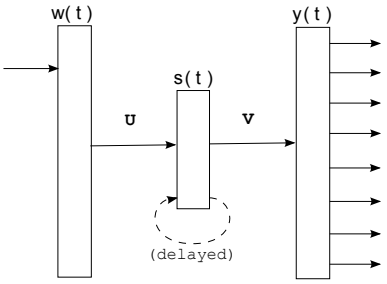

Figure 2.1: Schematic Representation of Recurrent Neural Network Language Model. The network has an input layer $\mathbf{w}$, a hidden layer $\mathbf{s}$ and an output layer $\mathbf{y}$. Matrices U and V represent synapses.

$$y_k(t) = g\Big(\sum_j s_j(t) v_{kj}\Big) \tag{2.36}$$

where $f(z)$ is a sigmoid activation function: $f(z) = \frac{1}{1+e^{-z}}$

$$f(z) = \frac{1}{1+e^{-z}} \tag{2.37}$$

and $g(z_m)$ is a softmax function (to make sure that the outputs form a valid probability distribution, i.e. all outputs are greater than 0 and their sum is 1):

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{2.38}$$

Both the feedforward and the recurrent architecture of the neural network based language model can be trained by the simple backpropagation algorithms (BP). However, for better performance, the so-called backpropagation through time (BPTT) algorithm can be used to train the recurrent architectures. With simple BP, the recurrent model can perform poorly in some cases. The BPTT algorithm is also described in, and a good description for a practical implementation is in [42]. The training algorithm is on-line gradient descent, thus the weights are updated after propagating every example. The cross entropy criterion is used to obtain an error vector in the output layer, which is then back propagated to the hidden layer etc. The training algorithm uses validation data for early stopping and to control the learning rate. Training iterates over all the training data in several epochs before convergence is achieved - usually $8-20$ epochs are needed. We refer the interested readers to [41; 43] for more details.

2.4 Search

In state-of-the-art speech recognition systems today, two types of decoding methods are used: *static decoding* and *dynamic decoding*.

In static decoding, a full cross word search network is built off line which is then searched through for finding the most likely word sequence. The structure of the search network results due to effect of context either at the state level (tri-phone) or word level (introduced by language model). Typically, copies of the phonetic tree[15] are maintained in the search network. As the leaves of these phonetic trees determine the word identity, the probability of transition between the copies of these trees is determined according to the n-gram language model. The state or context dependent phone of a phonetic tree is modeled by a HMM and the probability of emitting an output acoustic observation is obtained using GMM. For an n-gram LM, $|\mathcal{V}|^{n-1}+1$ copies are needed to be maintained at any time, where $\mathcal{V}$ is the vocabulary used. For instance, if the language model is a bigram language model ($n = 2$), then corresponding to each leaf of the first phonetic tree (this phonetic tree would correspond to the *predicting* word), one separate copy of the phonetic tree (these phonetic trees will correspond to the hypothesized *predicted* word) needs to be maintained. Most probably word sequence is then found out in this expanded network.

Expanding the whole network in a manner presented above, has been for long the most natural approach to LVCSR decoding until increase in the vocabulary size, order of language model and the phonetic context (tri-phone to quin-phones to even septa-phones) rendered this method impossible, due to memory limitations. Optimization techniques for compacting the network, approximations to the objective function and even a total change in the decoding paradigm were considered to make the static decoding approach feasible.

The network is usually compacted by exploiting the sparsity of knowledge sources and detecting and taking advantages of the redundancies. Unlikely state sequences are pruned early on to avoid state space blowing up issues (this type of pruning is referred to as *beam pruning* in speech literature). Also, instead of finding the joint probability of acoustic observation, $\mathbf{A}$, and word sequence, W'[16], (Eqn. 2.40), which is the sum over the joint probability of acoustic observation and all the possible state sequences, S, such that they give rise to the same same word sequence, $W(S) = W'$, (Eqn. 2.41), approximations are made by replacing the sum by the maximum joint distribution possible under one of the state sequences of W' (Eqn. 2.43). Word sequence, whose state sequence maximizes the joint distribution, is then outputted (this is called as *viterbi approximation* to MAP decoding.). Note that $P(\mathbf{A}|S)$ is obtained with the aid of an acoustic model while $P(S)$ can be obtained simply by getting the language model probability of the corresponding word sequence (usually the inter state transition probabilities do not matter much).

[15] Phonetic tree has one root node and the edges between any two connected nodes represent the context dependent phones such as tri-phones. The leaves determine the word identity whose phonetic composition can be obtained by percolating the tree from root node all the way down to the leaf. Thus there are as many leaves as there are words in the vocabulary of the system.

[16] This is required for MAP decoding. MBR decoding is intractable and impossible in the decoders of today's state-of-the-art recognizers.

$$
\begin{aligned}
W^* &= \underset{W'}{\operatorname{argmax}}\, P(\mathbf{A}|W')P(W') && (2.39)\\
&= \underset{W'}{\operatorname{argmax}}\, P(\mathbf{A}, W') && (2.40)\\
&= \underset{W'}{\operatorname{argmax}} \sum_{S:W(S)=W'} P(\mathbf{A}, S) && (2.41)\\
&= \underset{W'}{\operatorname{argmax}} \sum_{S:W(S)=W'} P(\mathbf{A}|S)P(S) && (2.42)\\
&\approx \underset{W'}{\operatorname{argmax}} \max_{S:W(S)=W'} P(\mathbf{A}|S)P(S) && (2.43)
\end{aligned}
$$

In-spite of huge optimizations and compactness, static decoders cannot make use of long span language models such as 4-grams and 5-grams.[17] Inability to use complex and long span language models in the static decoder gave rise to dynamic decoding approach. In dynamic decoding, instead of building the whole cross word network off line, it is built as and when required i.e. directly on the fly. There are two basic approaches for dynamically exploring an n-gram tree-structured network: (a) the *re-entrant tree* where the word history remains attached to each path coming back to the root and (b) the *start-synchronous tree* which generates the next hypotheses of all paths having simultaneous word endings. This type of decoding, as against static decoding, requires far less memory during run time, but then runs very slowly too.

The approach involving the re-entrant tree structure, follows closely with the static decoding method. In this approach, however, a single phonetic tree is maintained, but then each node of this tree maintains linguistic theories. For a correct use of n-gram LM, it is necessary to keep track of the individual $n-1$ word linguistic theories of the active paths, until the recombination step can take place at the next word ending.

The approach involving start-synchronous tree has been used in multi-stack decoders [44]. Stack decoding implements a best-first tree search which proceeds by extending word by word, one or several selected hypotheses without the constraint that they all end at the same time. Running hypotheses are handled using a stack which is a priority queue sorted on the likelihood scores and possibly on time. Use of heuristics (known as A^* principle) guide in selecting theories for further extensions and termination of the algorithm [10, pp. 93].

We refer interested readers to a very thorough survey article, written by Aubert [45], on decoding techniques for LVCSR systems.[18]

When compared to static decoders, dynamic decoder can handle relatively higher order language models (upto 4 or 5 grams). However, even for dynamic decoders, use of long span models such as RNNLM or MELM with triggers as features or some other complex

[17] Higher order models have to be pruned before they could be used.

[18] In order to be precise and concise, we have used description of some technique directly from Aubert's article [45].

LM, the decoding becomes computationally challenging. It is the aim of this thesis to discuss and present ideas and techniques which enables incorporation of long span and complex linguistic models for efficient decoding.

Chapter 3

Model Combination

The focus of this chapter is to investigate if there lies any extra power by combining various statistical knowledge sources for the eventual use (in some form) for decoding in speech recognition systems. While decoding schemes are separately discussed in Cchapters 4 and 5, in this chapter, we discuss an efficient training platform for combining various statistical models.

Over the past three decades, many fascinating and novel language modeling techniques have been introduced. We have discussed some of these language models in chapter 2. While some of these language models have shown significant improvements over the standard backoff n-gram language models, it is not clear if after combining some of these, we will do any better? To find out, we investigated this problem by combining many state-of-the-art language models on a very standard corputhes on which results of most of these models were already reported. Generous sharing of language models and appropriate toolkits by the researchers gave us a chance to find out if there lies any power in combining various statistical models and if their combination can outperform the best single model results.

The rest of the chapter is organized as follows: In the next section titled 'Language Model Combination', we will investigate the performance of some of the very powerful language models when combined together. This performance will be based solely on the basis of computing the perplexity (a function of the distribution characterized by the underlying language models) on some clean evaluation text data. In section titled 'Model Combination', we will take the concept further and investigate if combining language models has any impact on the accuracy of speech recognition systems.

3.1 Language Model Combination

Whenever a new language modeling technique is introduced to the speech community, its performance on the Wall Street Journal's Penn Tree Bank section is usually reported. With consistent vocabulary and partitioning of the corpus into train, dev and test, it then

becomes very convenient to compare and contrast various methods solely based on their performances. In this section, we will not only compare and contrast various language modeling techniques but also investigate the power of combining some of these language models and find out which of them provide complementary knowledge when combined together.

For the purpose of this study we worked on sections $0-24$ of Penn Tree Bank Corpus (PTB). Sections $0-20$ (930K word tokens) were used for training different models; Sections $21-22$ (74K word tokens) were used for tuning various interpolation weights and sections $23-24$ (82K word tokens) were used for evaluating the models either individually or after combining. The vocabulary was set to top 10K most frequent words found in the training data. We investigated the following language models:

- Backoff n-gram LMs with Good Turing discount smoothing [13; 14]
- Backoff n-gram LMs with modified interpolated Kneser-ney smoothing [17; 16; 12]
- Random clustering and class based LM [32; 20; 30]
- Cache LM [46]
- Within and across sentence LM [47]
- Maximum Entropy LM [24; 48]
- Structured LM [33]
- Syntactic LM [49; 37]
- Random Forest LM [19]
- Feedforward Neural Network LM [39; 50; 51; 52]
- Recurrent Neural Network LM [41; 43]

We refer interested readers to chapter 2 for a thorough discussion on some well known smoothing techniques for backoff n-gram models and all of the above mentioned advanced language modeling techniques. For the purpose of language model combination, we have combined the models at the word level. There are two forms of such combination: *linear interpolation* and *log-linear interpolation* [53]. Although, log-linear interpolation sometimes performs better than its linear counterpart, its immense computational complexity prohibits combining more than a certain number of models. We have hence used linear interpolation for combining models. If probability distribution over words, $w \in \mathcal{V}$ conditioned on the past context, $\mathbf{h}$, as characterized by some m^{th} language model is denoted by: $P_m(w|\mathbf{h})$, where $\mathcal{V}$ is the vocabulary used, then the linear interpolation of M different models is simply obtained as:

Model	Perplexity		
	individual	+KN5	+KN5+cache
3-gram with Good-Turing smoothing (GT3)	165.2	-	-
5-gram with Good-Turing smoothing (GT5)	162.3	-	-
3-gram with Kneser-Ney smoothing (KN3)	148.3	-	-
5-gram with Kneser-Ney smoothing (KN5)	**141.2**	-	-
5-gram with Kneser-Ney smoothing + cache	**125.7**	-	-
Maximum entropy model with 5-gram features [24; 48]	142.1	138.7	124.5
Random clusterings LM [32; 20]	170.1	126.3	115.6
Random forest LM [19]	131.9	131.3	117.5
Syntactic LM [37]	146.1	125.5	114.4
Within and across sentence boundary LM [47]	116.6	110.0	108.7
Feedforward neural network LM [51]	140.2	116.7	106.6
Feedforward neural network LM [52]	141.8	114.8	105.2
Syntactical neural network LM [50]	131.3	110.0	101.5
Recurrent neural network LM [41; 43]	124.7	105.7	97.5
Adaptive RNNLM [43]	123.2	102.7	98.0
Combination of static RNNLMs	102.1	95.5	89.4
Combination of adaptive RNNLMs	101.0	92.9	90.0

Table 3.1: Perplexity of individual models alone and after combination with backoff n-gram language model (with and without cache model). Results are reported on the Penn Treebank corpus evaluation set.

$$P(w|\mathbf{h}) = \sum_{m=1}^{M} \lambda_m P_m(w|\mathbf{h}) \tag{3.1}$$

where λ_m is the weight assigned to m^{th} model. The interpolation weight takes values between 0 and 1 and the total sum of all the interpolation weights should be 1. These weights are tuned so that the resulting probability distribution maximizes the likelihood on some held out data. Typically Expectation Maximization (EM) algorithm is used to find out these weights. See Appendix A for the exact use of EM algorithm for finding these optimal weights.

Table 3.1 shows the performance of various language models on the evaluation section of PTB. It can be seen from the top 4 rows that the Kneser-Ney smoothed 5-gram (3-gram) language model outperforms the corresponding n-gram models smoothed using Good Turing estimation. We will refer to n-gram Good Turing smoothed models by GTn and n-gram Kneser-Ney smoothed models by KNn. Cache model further improves the performance of n-gram models. Interpolation of KN5 with a cache model reduces the perplexity from 141.2 to 125.7.

Maximum Entropy based LM (ME-LM) using n-gram features achieves about the same performance as by KNn models. This is an interesting result mainly because there is no explicit smoothing in the ME-LMs. [1] ME-LMs trained using n-gram features outperform the GTn models. Thus in some sense, ME-LMs with n-gram features establishes the upper bound on the performance of backoff n-gram models based solely on the smoothing methodologies. Since the ME-LMs do not use any features other than the n-grams, they are not expected to have any complementary information to the KNn models (which are supposed to be state-of-the-art among the backoff models). Although the linear interpolation of KN5 and ME-LM does reduce the overall perplexity, this reduction is very small. Interpolation of this combined model with a cache model gives the perplexity of 124.5 which is again just slightly less than that of linear interpolation of KN5 and cache model, which gives the perplexity of 125.7.

Emami et.al.'s random clustering LM [32], which is in some sense a generalization of Martin et.al.'s class based LM [20] performs worse than KNn model. This is expected mainly because class based LMs by themselves do not capture local information as well as any n-gram model. Class based models cluster words into equivalence classes and hence are complementary to n-gram models especially when the latter models are insufficient to generalize on less frequently seen n-gram context. Linear interpolation of class based models with KNn model reduces the perplexity to 126.3 which is better than the performance of any of the individual models. Random Forest LM [19] puts an equivalence classification on the context i.e. the history conditioned on which the next word is predicted. These models inherently use n-grams models when they have to backoff. These models perform better than KNn models. Interpolation of these models with cache models reduces the perplexity further to 117.5.

Filimonov et.al.'s syntactic LM [37] which although is a left to right model but uses the entire context for prediction, achieves relatively much better performance. It achieves the perplexity of 125.5 when combined with KNn model. This further reduces to 114.4 after interpolating it with the cache model. Slightly better results were reported on this dataset with yet another syntactic LM - SuperARV LM [49]. Perplexity of 118.4 was reported by using SuperARV language model combined with KNn model. Chelba et.al.'s structured LM [33] which was one of the very first LM utilizing syntactic information, achieved relatively poor performance. The perplexity of the model after interpolating with KNn model was 146. We did not use this model in our study mainly because over the years better syntactic LMs were introduced.

Momtazi et.al.'s within and across sentence language model [47], which is a combination of several simpler models (including n-gram models, skip n-gram models, cache-like models and class based models) outperforms all of the LMs discussed so far. When combined with KNn model, the performance improves further to yield a perplexity of 110.0.

Exact implementation of Bengio's feedforward neural network language model [52; 51] achieves about the same perplexity as the KNn model. Interpolation of this neural network

[1] ME-LMs tend to assign uniform distribution from the left over probability mass to all the unseen events.

Model	Weight	PPL
3-gram with Good-Turing smoothing (GT3)	0	165.2
5-gram with Kneser-Ney smoothing (KN5)	0.0237	141.2
5-gram with Kneser-Ney smoothing + cache	0.08	125.7
Maximum entropy model	0	142.1
Random clusterings LM	0	170.1
Random forest LM	0.089	131.9
Structured LM	0.021	146.1
Within and across sentence boundary LM	0.082	116.6
Log-bilinear LM	0.0113	144.5
Feedforward NNLM	0.010	140.2
Syntactical NNLM	0.083	131.3
Combination of static RNNLMs	0.3	102.1
Combination of adaptive RNNLMs	0.3	101.0
ALL	1	83.5

Table 3.2: Results on Penn Treebank corpus (evaluation set) after combining all models. The weight of each model is tuned to minimize perplexity of the final combination.

based LM with KNn model reduces the perplexity to 116.7 which still is worse than within and across sentence LM. While Bengio's feedforward neural network LM utilized only lexical information, Emami et.al.'s extension utilized syntactic features [50]. Use of syntactic features (which were obtained using Chelba et.al.'s structured LM [33]) obtains a perplexity of 131.3 which then reduces to 110.0 after interpolating with KNn model.

Mikolov et.al.'s recurrent neural network LM [41; 43] achieves the lowest perplexity when compared to all of the LMs discussed so far. Interpolation of this LM with KNn model reduces the perplexity further to 105.7. As far as we know, this is the lowest perplexity number ever reported on this setup.

Table 3.2 shows the effect of combining all the above mentioned language models. The LMs were combined at the word level using Expectation Maximization (EM) technique. Appendix A illustrates the EM algorithm for finding optimal interpolation weights for language model. It can be seen from the table that out of 13 models that were combined together, only 7 got a significant non-zero weight. The remaining 6 models got extremely low weight (some also got a weight of 0), indicating that these models did not add any complementary information.

It is not surprising to see a weight of 0 assigned to GTn model because the models KNn (+cache) contain all the necessary information. It is very interesting to see that the ME-LM gets a weight of 0. In Table 3.1, we observed that the performance of both KNn model and ME-LM is very similar and since ME-LM uses only n-gram features, it is not surprising that a well smoothed n-gram model would be able to capture all the necessary information. The fact that KN smoothing is the state-of-the-art smoothing method for n-gram LMs, it is reasonable to see a weight of 0 to ME-LMs.

The reason why random clustering LM got a weight of 0 is not entirely clear. Probably some of this information is already captured by some other LM.

It is very interesting to see that the feedforward LM too got a very low weight of 0.01. Recurrent neural network LMs, which can be thought of as the generalization of the feedforward LM, got the highest weight. It thus seems that feedforward neural network LM does not offer any complementary information to the recurrent neural network LM.

The combination of all the LMs obtains a perplexity of $\mathbf{83.5}$ which is considerably lower than that of any individual system. The conclusion of this empirical study is that often different language models built from the same data capture complementary information and hence upon optimal combination, the performance of the combined model is considerably superior than the best individual model's performance.

In the next section, we will take this idea further and investigate if combination of some of the state-of-the-art language models have any effect on the reduction of WER for speech recognition system. For text processing, model combination parameters can be tuned to minimize the perplexity on some held out data. Perplexity is a function of the probability distribution characterizing the underlying language model, hence for linear or log-linear interpolation of models, Expectation Maximization can be used to find out the model weights (see Appendix A). Since WER cannot be written as a function of the probability distribution alone, its minimization puts a restriction on the form in which models are combined. Moreover, WER needs to be computed on sentence level and hence minimization of overall WER (training error) necessitates the computation of probability of the entire sentence hypothesis. It thus becomes easy to combine models on the sentence level rather than word level. In our work too, we combine models on sentence level rather than word level. The incorporation of sentence specific language models thus inherently becomes possible in this setup. More discussion on the exact objective function and training criteria is discussed in following sections.

3.2 Model Combination for ASR

The idea behind combining various knowledge sources is to tap on the complementary information provided by various models and use it to perform significantly better than any single best model. However, when it comes to applications such as speech recognition or machine translation, there are two ways to achieve the combination of various knowledge sources – via *System Combination* or *Model Combination*.

- In **System Combination** the knowledge sources are combined at the system level i.e. the outputs from systems using each individual knowledge source independently, is combined together. Although the methods based on *bagging* (voting) were already introduced in Machine Learning [54], they were recasted in specific and precise form for speech recognition community by Jonathan Fiscus. His now widely used method, ROVER (recognizer output voting error reduction) [55] was the first demonstrable

technique in which it was shown that combination of outputs of various single best systems yield an improved performance.[2] The intuition behind this technique is that by exploiting the differences in the errors caused by each system and by way of collective voting, a more refined and improved output can be obtained.[3] Thus the inadequacies of one system can be taken care by other systems etc. The outputs from each individual system (mostly 1-best output) are aligned together to form a word transition network. The network is usually created using modifications of the dynamic programming alignment. The word receiving highest votes at each branching point of the word network i.e. the one which has highest *consensus* among the different systems is then selected. J. Fiscus discusses some specific weighting schemes essential for the assignment of proper scores to any word in the word network. Voting schemes based on a) frequency of occurrence, b) frequency of occurrence and average word confidence and c) frequency of occurrence and maximum confidence is used for weighting words output by the systems. Although contextual information was not used during re-scoring in this original work, Schwenk et.al. [56] later extended the basic ROVER technique to incorporate the language modeling information during re-scoring. These methods are now very popular even in domains other than speech recognition. For instance, Karakos et.al. uses the system combination technique followed by language model rescoring on various Machine Translation tasks [57].

Goel et.al. [58] too extended the ROVER technique but the extension was done to generalize the method for arbitrary recognizer output. They proposed eROVER (extended-ROVER) to accommodate the alignment of word lattices outputted by various individual systems and they demonstrated an improved performance on switchboard task. Consensus decoding technique [59] developed by Mangu et.al. too had a ROVER like flavor in the sense that a multiple alignment of all the hypotheses in a word lattice was carried out to obtain consensus among the outputs produced by one single system. Posterior probabilities on words served as weights for weighted voting.

A choice of different recognition paradigm or choice of different language models and acoustic models can bring in diversity for system combination to work very well. One interesting line of work in this direction was by Siohan et.al. [60] in which a randomness induced in the formation of phonetic decision tree brought huge diversity in the output of the recognizer (which remained essentially the same except for the phonetic decision trees that indirectly defined the form of acoustic model). Change of language models during the decoders too can produce enough diversity in the output for system combination to work better than the best individual system. In this chapter, we have looked at this specific problem of using different language models

[2] In fact it was shown that the even if the single system performances were not very good, the system combination performance was remarkably better.

[3] The fact that diverse systems make uncorrelated errors, makes the ROVER technique so powerful.

to induce randomness for effective system combination, while comparing it to model combination technique, which we discuss next.

- **Model Combination** on the other hand is the technique in which different knowledge sources are unified under one framework and then the single system (which now incorporates various knowledge sources) produces just one output. Peter Beyerlein [61] demonstrated discriminative model combination (DMC) technique in which he unified various statistical models, including acoustic model and various n-gram language models, in one log-linear framework. An interesting work on these lines was done by D. Vergyri et.al. [62] in which the authors combined multiple acoustic models, obtained using training corpora from different languages, in order to improve ASR performance in languages for which large amounts of training data are not available. Static weights and dynamic weights based on different phonetic classes [63] were used to combine different knowledge sources.

  In this chapter, we focus on this problem too, however, with a different recipe in mind. The objective function of [61; 62; 63] aimed at minimizing the training error by choosing the 1-best output using MAP decoding. However, as we will describe in detail in the remainder of the chapter, such an objective function does not guarantee minimization of expected word error rate (the loss function we are after). Hence a change in the paradigm is necessary to guarantee minimization of the chosen loss function (or atleast come close to it). In this chapter we are proposing the modification to Beyerlein's framework so that under some limiting conditions it guarantees minimization of word error rate on an average. The framework that we will be using will be referred to as Empirical Bayes Risk minimization.

As discussed in Sec. 3.1, due to the limitation posed by the minimization of WER, which is not a function of the probability distribution alone (being characterized by the underlying language model), it becomes convenient to combine various models at the sentence level. We have thus focused on the *log-linear combination* (detailed in Section 3.4) of several different models, using, in particular, the Deterministic Annealing framework (described in Section 3.6) to carry out non-convex optimization based on the Empirical Bayes Risk objective function (described in Section 3.5). Rao et.al.[64] trained an *isolated word* speech recognition system using Deterministic Annealing (DA) to minimize the smoothed error function. Smith and Eisner previously showed the efficacy of using annealing techniques in training log linear models for machine translation and dependency parsing tasks [65]. In our work, we have focused on Large Vocabulary Continuous Speech Recognition (LVCSR) task using combinations of various n-gram, syntactic and other forms of language models. We also use this technique to combine various recurrent neural network based language models and n-gram models. The focus of this chapter is to demonstrate an efficient training procedure for model combination. Chapter 4 and 5 will focus on the decoding schemes using this complex combination of long and short span language models.

We begin by discussing the optimal objective functions necessary to guarantee minimization of the chosen loss function (WER for speech recognition setting). Starting with the standard setting of classification problem in machine learning, we will establish the ground work and then go into more details for a general speech recognition setup.

3.3 Inductive Learning

The aim of any inductive learning technique should not be to minimize the training error of the classifier, but to minimize its generalization error. If we denote the class labels by the random variable $\mathbf{Y}$ taking values in $\mathcal{Y}$ and observations by $\mathbf{X}$ taking values in $\mathcal{X}$, then the classifier $f(\cdot)$ should be trained to minimize the generalization error $e_f = P(\mathbf{Y} \neq f(\mathbf{X}))$. Under $1/0$ loss function (denoted by $L(\cdot)$) i.e. which assigns the loss of 1 for any misclassification and 0 for correct classification, the Expected loss between the true class labels and hypothesized class labels, $E_{P(\mathbf{X},\mathbf{Y})}(L(\mathbf{Y}, f(\mathbf{X})))$ is equal to the generalization error, $e_f = P(\mathbf{Y} \neq f(\mathbf{X}))$. The proof is simple and is given below for completeness purposes:

$$
\begin{aligned}
E_{P(\mathbf{Y},\mathbf{X})}(L(\mathbf{Y}, f(\mathbf{X}))) &= \sum_y \int_x P(x,y) L(y, f(x)) \\
&= \sum_y \Big(\int_{x:f(x)=y} P(x,y) L(y, f(x)) \\
&\quad + \int_{x:f(x)\neq y} P(x,y) L(y, f(x)) \Big) \\
&= \sum_y \Big(\int_{x:f(x)=y} P(x,y) \cdot 0 \\
&\quad + \int_{x:f(x)\neq y} P(x,y) \cdot 1 \Big) \\
&= P(\mathbf{Y} \neq f(\mathbf{X}))
\end{aligned}
$$

Since in this special case of $1/0$ loss function, generalization error is equal to the expected loss, the training criteria can be made to minimize the expected loss instead of generalization error. However, for any arbitrary loss function, $L(\cdot) \in \mathbb{R}$, it is not guaranteed that by way of minimizing the Expected Loss, the generalization error, e_f, will also be minimized. Under the assumption that the loss function for the task at hand can take only discrete values and hence there $\exists \epsilon > 0$ such that the range of loss function is bounded below by ϵ, then using Markov's inequality, we can see that the generalization error is in-fact very loosely bounded above by some scaled value of Expected Loss.

$$\begin{aligned} e_f &= P(\mathbf{Y} \neq f(\mathbf{X})) \\ &= P(L(\mathbf{Y}, f(\mathbf{X})) \neq 0) \\ &= P(L(\mathbf{Y}, f(\mathbf{X})) \geq \epsilon), \epsilon > 0 \end{aligned}$$

If we treat $L(\mathbf{Y}, f(\mathbf{X}))$ as a random variable distributed under the joint distribution $P(\mathbf{Y}, \mathbf{X})$, then from Markov's inequality, we get:

$$P(L(\mathbf{Y}, f(\mathbf{X})) \geq \epsilon) \leq \frac{E_{P(\mathbf{Y},\mathbf{X})}[L(\mathbf{Y}, f(\mathbf{X}))]}{\epsilon}$$

This bound is very loose and is useless for any practical purposes. Thus there is no guarantee that minimization of Expected Loss would also minimize the generalization error, however, since minimizing latter is not that straightforward, it is the former which is minimized in any practical system. Under the *true* joint distribution $P(\mathbf{Y}, \mathbf{X})$, the aim then would be find out that classifier which minimizes the Expected Loss i.e.:

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} E_{P(\mathbf{Y},\mathbf{X})}[L(\mathbf{Y}, f(\mathbf{X}))]$$

Under the above mentioned objective criteria, the optimal class label for any given $X = x$ is given as[4]:

$$f(x) = \underset{y' \in \mathcal{Y}}{\operatorname{argmin}} E_{P(\mathbf{Y}|x)}[L(\mathbf{Y}, y')] \tag{3.3}$$

Typically the classifier, $f(x)$, is some function of the conditional distribution i.e. $P(\mathbf{Y}|\mathbf{X} = x)$. However, we rarely have access to the true distribution and hence we will parameterize the conditional distribution $P(\mathbf{Y}|x)$ by $\lambda \in \Lambda$ and search for optimal λ instead i.e.:

[4]If, however, the loss function, $L(\cdot, \cdot)$, is a $1/0$ loss function, then the optimal class label for any given $X = x$ is simply the label $y \in \mathcal{Y}$ which has the Maximum A-Posterior (MAP) probability given $X = X$, as shown below:

$$\begin{aligned} f(x) &= \underset{y' \in \mathcal{Y}}{\operatorname{argmin}} E_{P(\mathbf{Y}|x)}[L(\mathbf{Y}, y')] \\ &= \underset{y' \in \mathcal{Y}}{\operatorname{argmin}} \sum_{y \in \mathcal{Y}} L(y, y') P(y|x) \\ &= \underset{y' \in \mathcal{Y}}{\operatorname{argmin}} \sum_{y \in \mathcal{Y}} (1 - \delta(y, y')) P(y|x) \\ &= \arg\max_{y' \in \mathcal{Y}} P(y'|x) \end{aligned} \tag{3.2}$$

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}} E_{P(\mathbf{Y},\mathbf{X})}[L(\mathbf{Y}, f_\lambda(\mathbf{X}))] \tag{3.4}$$

Note that we have parameterized the classifier function since it is implicitly a function of our parameterized conditional distribution. The optimal class label for the input x is then given by:

$$f_{\lambda^*}(x) = \underset{y' \in \mathcal{Y}}{\operatorname{argmin}} E_{P_{\lambda^*}(\mathbf{Y}|x)}[L(\mathbf{Y}, y')] \tag{3.5}$$

Note that we cannot parameterize the true distribution i.e. $P(\mathbf{Y}, \mathbf{X})$ in Equation 3.4, because by doing so we may end up getting some degenerate classifier. We can atmost approximate the Expectation evaluated under this true joint distribution in Eqn. (3.4) by time average over some huge training data (assuming it to be generated by the underlying distribution $P(\mathbf{Y}, \mathbf{X})$).

Let us represent the labelled training data by $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{|\mathcal{T}|}, y_{|\mathcal{T}|})\}$. From the weak law of large numbers (WLLN) [66] we can approximate the Expectation under the true joint distribution by the time average over the training data i.e. that:

$$\forall \lambda \in \Lambda \quad \lim_{|\mathcal{T}| \to \infty} \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} L(y_t, f_\lambda(x_t)) \quad = \quad E_{P(\mathbf{X},\mathbf{Y})}[L(\mathbf{Y}, f_\lambda(\mathbf{X})] \tag{3.6}$$

Then the optimal parameter can be found out using the following approximation instead:

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} L(y_t, f_\lambda(x_t)) \tag{3.7}$$

Thus it is reasonable to assume that if we have large amount of labelled training data, then optimal model parameters (λ) can be found out using Eqn. (3.7). Substituting the form of the classifier from Eqn. (3.5), we see that the optimal parameters are found out as follows:

$$\lambda^* \quad = \quad \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} L\Big(y_t, \underbrace{\underset{y' \in \mathcal{Y}}{\operatorname{argmin}} E_{P_\lambda(\mathbf{Y}|x_t)} L(\mathbf{Y}, y')}_{\text{Optimal Classifier for general loss.}}\Big) \tag{3.8}$$

In practice, however, the choice of such a classifier turns out to be extremely cumbursome. With parameter λ taking values in $\mathbb{R}^d$, objective function becomes extremely difficult to optimize. Due to such pragmatic considerations, optimal classifier with respect to $1/0$

loss is used for minimizing the training error, i.e. that, the optimal parameters are found out using the following criteria:

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} L\Big(y_t, \underbrace{\arg\max_{y' \in \mathcal{Y}} P_\lambda(y'|x_t)}_{\text{Optimal Classifier for } 1/0 \text{ loss.}}\Big) \tag{3.9}$$

The computation of $\arg\max(\cdot)$ of Eqn 3.9 is cheaper than the computation of $\operatorname{argmin}(\cdot)$ of Eqn 3.8 and hence the standard practice is to use the objective function of Eqn 3.9 during training. However, this objective function is not justified even in the limit as the training data goes to infinity. Hence, if the aim of our inductive learning is to minimize some arbitrary loss, on an average, then objective function as described in Equation 3.8 is well suited and has theoretical justifications especially under the limit as the training data becomes larger.

Even with simpler objective function form (3.9), the optimization problem can become extremely difficult with higher dimensional parameter vector. For small dimensional parameter vector, exhaustive search could still be carried out, but the parameter search problem becomes prohibitive with increasing parameter vector size. For certain applications such as Machine Translation and Speech Recognition, this objective function turns out to be piece wise constant (due to the specific nature of the loss function metric used (BLEU [67] and WER respectively)). Due to this reason, gradient descent like techniques cannot be used for solving the optimization function.

To address this issue, a simpler objective function is proposed below Eqn. (3.10). This objective function is a smooth function of model parameters (λ) and hence facilitates the use of gradient descent like techniques. This objective function has previously been proposed by Smith and Eisner [65] and Och [68] in Machine Translation and we in this chapter focus on its applicability for large vocabulary speech recognition task under the model combination framework.

$$\lambda^* = \underset{\lambda \in \Lambda}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} \sum_{y' \in \mathcal{Y}} L(y_t, y') P_\lambda(y'|x_t) \tag{3.10}$$

For the special case of speech recognition application, we will try to investigate if the above mentioned objective function is in any sense related to the optimal objective function given by Eqn. 3.4.

3.3.1 Speech Recognition

Let us establish some notation. Let us denote our training data-set by $\mathcal{T}$, and let there be $|\mathcal{T}|$ speech utterances to decode and to rescore the recognizer's corresponding

first pass search space. We will index these speech utterances by the letter $\mathbf{o}_t$, where $t = \{1, 2, \ldots, |\mathcal{T}|\}$. Let the search space output by the recognizer be N-best lists. Thus for each speech utterance $\mathbf{o}_t$, we will have top N-best hypotheses according to some baseline model. Let these N-best hypotheses[5] be denoted by $W_{t,1}, \ldots, W_{t,N}$. For each speech utterance, $\mathbf{o}_t$, let us denote the corresponding true transcript by $W_{t,0}$. Let us represent the true, but unknown, joint distribution of transcript and acoustic vector by $P(\mathbf{W}, \mathbf{O})$.[6] Since our classifier is a function of this distribution, we can approximate the true distribution by learning a parameterized version of it: $P_\lambda(\mathbf{W}, \mathbf{O})$ and then parameterize the classifier function with λ. The training criterion (following Equation (3.4)) becomes:

$$\lambda^* = \operatorname*{argmin}_{\lambda \in \Lambda} E_{P(\mathbf{W},\mathbf{O})}[L(\mathbf{W}, f_\lambda(\mathbf{O})]$$

where $L(W, W')$ is the edit distance between word string W and W' and $f_\lambda(\mathbf{O} = O)$ is the parameterized classifier, which takes the form:

$$f_\lambda(O) = \operatorname*{argmin}_{W' \in \mathcal{W}} E_{P_\lambda(\mathbf{W}|O)}[L(\mathbf{W}, W')]$$

If we represent the space of word strings by $\mathcal{W}$ and space of acoustic strings by $\mathcal{O}$, then the above Expectation and hence the objective function can be written as:

$$\lambda^* = \operatorname*{argmin}_{\lambda \in \Lambda} \int_{O \in \mathcal{O}} \sum_{W \in \mathcal{W}} L(W, f_\lambda(O)) P(W, O) \partial O \tag{3.11}$$

$P(W, O)$ is the joint probability of W and O.

If we now denote the *true* but unknown classifier by $g(\cdot)$, then under the training regime we know what is the correct output of such a true classifier (since for any given acoustic, O, it is possible to get the true transcript using human annotators).

Then if the following inequality holds true:

$$\begin{aligned} \forall \lambda \in \Lambda, \quad & \int_{O \in \mathcal{O}} \sum_{W \in \mathcal{W}} L(W, f_\lambda(O)) P(W, O) \\ \leq & \int_{O \in \mathcal{O}} \sum_{W \in \mathcal{W}} L(W, g(O)) P_\lambda(W, O) \end{aligned} \tag{3.12}$$

we can search for the optimal $\hat{\lambda}$ (upto a certain approximation) by considering the following objective function instead:

[5] For simplicity, we assume that N is the same for all speech utterances, although it is not required.

[6] We will use bold symbols for random variables.

$$\hat{\lambda} \;=\; \underset{\lambda\in\Lambda}{\operatorname{argmin}} \int_{O\in\mathcal{O}} \sum_{W\in\mathcal{W}} L(W, g(O)) P_\lambda(W, O) \tag{3.13}$$

i.e. that, the parameters so found will minimize some upper bound on the Expected Loss we are after.[7]

Using Bayes's decomposition rule, we can decompose the posterior probability to get:

$$\hat{\lambda} \;=\; \underset{\lambda\in\Lambda}{\operatorname{argmin}} \int_{O\in\mathcal{O}} \sum_{W\in\mathcal{W}} L(W, g(O)) P_\lambda(W|O) P(O) \partial O \tag{3.14}$$

The above Expectation can be approximated by quantizing the space of acoustic observations thus leading to:

$$\hat{\lambda} \approx \underset{\lambda\in\Lambda}{\operatorname{argmin}} \sum_{O\in\mathcal{O}} \sum_{W\in\mathcal{W}} L(W, g(O)) P_\lambda(W|O) P(O) \tag{3.15}$$

We can further approximate the search by restricting the second summation, for every $O \in \mathcal{O}$, to a sub-space of word strings $\mathcal{W}_O (\subseteq \mathcal{W})$ such that word strings in $\mathcal{W}_O$ are often put out by the recognizer when it is given the task of recognizing the audio O. Thus for every $O \in \mathcal{O}$, set $\mathcal{W}_O$ is a typical set i.e. that $\sum_{W\in\mathcal{W}_O} P(W|O) \approx 1$, where $P(\cdot)$ denotes the *true* posterior probability.

$$\hat{\lambda} \approx \underset{\lambda\in\Lambda}{\operatorname{argmin}} \sum_{O\in\mathcal{O}} \sum_{W\in\mathcal{W}_O} L(W, g(O)) P_\lambda(W|O) P(O) \tag{3.16}$$

Finally, we can replace the outer sum (which is a sum over alphabet $\mathcal{O}$) with the sum over the training corpus. Thus $P(O)$ gets substituted by the normalizing term $\frac{1}{|\mathcal{T}|}$. Similarly, for every O, we can replace the inner sum (which is the sum over the alphabet $\mathcal{W}_O$ by the sum over the N-best list of hypotheses outputted by the recognizer upon given the task of transcribing O.

$$\begin{aligned}
\hat{\lambda} \;&\approx\; \underset{\lambda\in\Lambda}{\operatorname{argmin}} \sum_{t=1}^{|\mathcal{T}|} \sum_{n=1}^{N=(|\mathcal{W}_O|)} L(W_{t,n}, \underbrace{W_{t,0}}_{=g(O_t)}) P_\lambda(W_{t,n}|O_t) \frac{1}{|\mathcal{T}|} \\
&=\; \underset{\lambda\in\Lambda}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} \sum_{n=1}^{N=(|\mathcal{W}_O|)} L(W_{t,n}, \underbrace{W_{t,0}}_{=g(O_t)}) P_\lambda(W_{t,n}|O_t)
\end{aligned} \tag{3.17}$$

[7] The inequality (3.12) suggests that the expected loss of any classifier (possible only in the family of MBR classifiers) under the *true distribution* is always less than or equal to the expected loss of the *true classifier* under some parameterized approximation of the true distribution.

(Eqn. 3.17) is exactly of the form given in (Eqn. 3.10). Thus our proposed objective function aims to minimize the Expected Loss of the *true* classifier where the expectation is taken under a parameterized approximation of the true distribution. Under the assumption of (3.12) the proposed technique will aim to minimize the upper bound on the Expected Loss of the optimal classifier.

For the specific case of Speech Recognition, in the next subsection we will empirically show that the Inequality (3.12) is indeed true for a range of λ values.

3.3.2 Analyzing Inequality (3.12)

In this section we will try to find out, empirically, whether the inequality (3.12) holds true for a range of values of λ. We will experiment with a certain speech recognition setup in which the task is to find out the optimal LM scaling parameter (λ of $P(A|W)^{1/\lambda}P(W)$). We will try to analyze the expected loss of following three different classifiers of which one classifier is the *true* classifier.

1. **MAP:** For any $\lambda \in \Lambda$, the expected loss is computed as:

$$\frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|} L\Big(W_{t,0}, \arg\max_{W' \in \mathcal{W}_{O_t}} P_\lambda(W'|O_t)\Big), \tag{3.18}$$

   where $\mathcal{W}_{O_t}$ space is represented by N-best list.

2. **MBR:** For any $\lambda \in \Lambda$, the expected loss is computed as:

$$\frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|} L\Big(W_{t,0}, \arg\min_{W' \in \mathcal{W}_{O_t}} E_{P_\lambda(\mathbf{w}|O_t)}[L(\mathbf{W}, W')]\Big), \tag{3.19}$$

   where the inner expectation is approximated over N-best lists.

3. **True:** For any $\lambda \in \Lambda$, the expected loss is computed as:

$$\frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N=(|\mathcal{W}_O|)} L(W_{t,n}, W_{t,0}) P_\lambda(W_{t,n}|O_t) \tag{3.20}$$

It can been seen from Fig. 3.1, that as we sweep λ value, the expected loss of the *true* classifier is always greater than that of MAP and MBR classifiers. It can also be seen that the point minimizing the Expected Loss of the true classifier also minimizes the expected loss of MAP and MBR classifiers. Thus this experiment provides a good empirical evidence of the fact that the proposed objective function is not that bad indeed.

Having established the form of the objective function, we will now discuss the parameterization of the conditional distribution i.e. $P_\lambda(\mathbf{W}|\mathbf{O})$ needed to approximate the true distribution, which is always hidden from us.

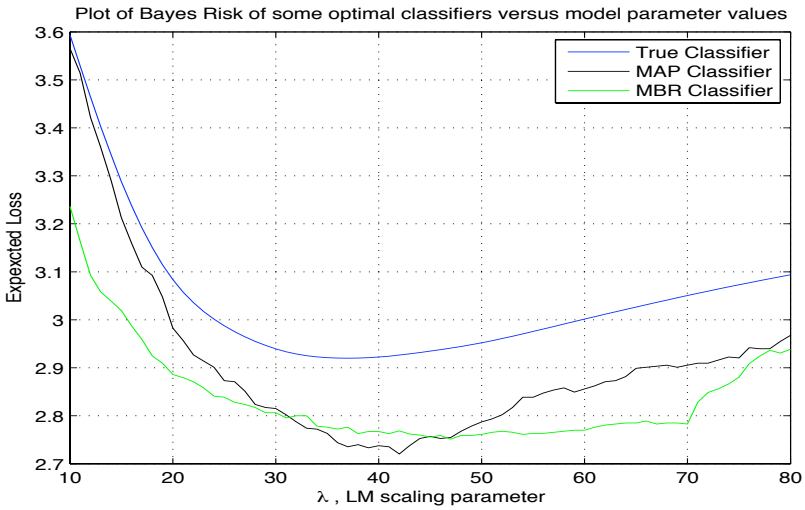

Figure 3.1: Plot showing Expected loss under various classifiers when the model parameter, LM scaling parameter in this case, is varied. It can been seen that as we sweep λ value, the Expected loss of the *true* classifier is always greater than that of any other classifier. It can also be seen that the point minimizing the Expected Loss of the true classifier also approximately minimizes the expected loss of MAP and MBR classifiers. Thus this experiment provides a good empirical evidence of the fact that the proposed objective function is not that bad indeed.

3.4 Log Linear Model

As previously mentioned, due to the nature of the objective function, it becomes convenient to combine language models at the sentence level rather than word level. However, instead of combining just language models together, we have taken the approach of Beyerlein's [61] to unify all the statistical models (including acoustic model) in one log-linear framework. These statistical models assign likelihoods and probabilities to the entire word string hypotheses. Thus our framework can accommodate whole-sentence discriminative models, as well as generative models factorizable using chain rule. The resulting log-linear model too assigns the probabilities to the entire word string hypotheses and is thus convenient for minimizing expected loss (WER in our case).

We will continue to follow the notation introduced in Sub. Sec 3.3.1 and introduce some more. Let there be M models, each assigning scores to the N hypotheses. Let us denote these models by $q_m(\cdot|\mathbf{o}_\cdot)$, where $m = \{1, 2, \ldots, M\}$. Thus under any m^{th} model, the score of n^{th} hypothesis, $W_{t,n}$, obtained after the first pass decoding of t^{th} speech utterance, will be denoted as $q_m(W_{t,n}|\mathbf{o}_t)$. We now define our log linear model $P_\Lambda(\cdot|\mathbf{o}_\cdot)$ as shown below:

$$P_\Lambda(W_{t,n}|\mathbf{o}_t) = \frac{(\exp\{\sum_{m=1}^{M} \lambda_m q_m(W_{t,n}|\mathbf{o}_t)\})}{\sum_{k=1}^{N} \exp\{\sum_{m=1}^{M} \lambda_m q_m(W_{t,k}|\mathbf{o}_t)\})} \tag{3.21}$$

where λ_m is the model weight for m^{th} recognition model. For any $\mathbf{o}_t$, the term $\sum_{k=1}^{N} \exp\{\sum_{m=1}^{M} \lambda_m q_m(W_{t,k}|\mathbf{o}_t)\})$ is called the partition function and will be represented by Z_t.

3.5 Objective Function

For speech recognition, the loss function of interest to us is WER (word error rate). Following 3.10, we define the Expected-Word Error Rate (E-WER) under some model, $P_\Lambda(\cdot)$, as:

$$\mathbf{E}_{P_\Lambda}[L] = \frac{1}{|\mathcal{R}|} \sum_{t=1}^{|\mathcal{T}|} \sum_{n=1}^{N} L(W_{t,n}, W_{t,0}) P_\Lambda(W_{t,n}|\mathbf{o}_t) \tag{3.22}$$

where, $|\mathcal{R}|$ is the sum of the length of $|\mathcal{T}|$ true transcripts. We can thus rewrite the above Expected loss function as:

$$\begin{aligned} \mathbf{E}_{P_\Lambda}[L] &= \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} \sum_{n=1}^{N} \frac{|\mathcal{T}|}{|\mathcal{R}|} L(W_{t,n}, W_{t,0}) P_\Lambda(W_{t,n}|\mathbf{o}_t) \\ &= \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} \sum_{n=1}^{N} L'(W_{t,n}, W_{t,0}) P_\Lambda(W_{t,n}|\mathbf{o}_t) \end{aligned}$$

where $L'(\cdot,\cdot) = \frac{|\mathcal{T}|}{|\mathcal{R}|} L(\cdot,\cdot)$. Henceforth, we will use $L(\cdot,\cdot)$ to mean $L'(\cdot,\cdot)$.

3.6 Deterministically Annealed Training

For a non-convex objective function, use of any hill climbing method results in the locally optimum solution, which may or may not be the globally optimum solution. For our case, unfortunately, E-WER is a non-convex function of the model parameters and hence use of Gradient-Descent-based optimization method does not guarantee finding the globally optimum point. Deterministic Annealing [64] attempts to solve this problem by introducing a hyper parameter γ to Eq. 3.21, which modifies it as shown below:

$$P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) = \frac{(\exp\{\sum_{m=1}^{M} \lambda_m q_m(W_{t,n}|\mathbf{o}_t)\})^\gamma}{\sum_{k=1}^{N} (\exp\{\sum_{m=1}^{M} \lambda_m q_m(W_{t,k}|\mathbf{o}_t)\})^\gamma} \tag{3.23}$$

When the value of γ is set to 0, the log linear model assigns uniform distribution to all the hypotheses, resulting in a higher entropy model. When γ is set to value 1, then the log linear model takes the form of Eq. 3.21 and when $\gamma \to \infty$, the model approaches the form where all probability mass is concentrated towards one specific hypothesis, resulting in a

much lower entropy distribution. If the parameters of the model are trained to minimize 1-*best* loss, then the distribution mass is concentrated towards the best hypothesis.

As very neatly analyzed in [65], for a fixed γ, deterministic annealing solves:

$$\begin{aligned}\Lambda^* &= \underset{\Lambda}{\operatorname{argmin}}\, \mathbf{E}_{P_{\Lambda,\gamma}}[L(\cdot,\cdot)] \\ &= \underset{\Lambda}{\operatorname{argmin}}\, \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N} L(W_{t,n}, W_{t,0}) P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) \end{aligned} \tag{3.24}$$

When γ is increased according to some schedule, Λ is optimized again. Since lower values of γ smooths the distribution, this may allow us to avoid the locally optimum solution, which otherwise may not be possible under relatively higher values of γ.[8]

Given the form of the model in Eq. 3.23, γ can be multiplied by Λ and taken inside the exponentiation. Hence any change in γ can be easily compensated for a respective change in the value of Λ vector. Hence in [64] a direct preference for the Shannon entropy, H, was expressed instead. Thus, γ and Λ are chosen according to:

$$\begin{aligned}\{\Lambda^*, \gamma^*\} &= \underset{\Lambda,\gamma}{\operatorname{argmin}}\, \mathbf{E}_{P_{\Lambda,\gamma}}[\mathcal{L}(\cdot,\cdot)] - \Theta \mathbf{H}(P_{\Lambda,\gamma}(\cdot)) \\ &= \underset{\Lambda,\gamma}{\operatorname{argmin}}\, \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N} L(W_{t,n}, W_{t,0}) P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) \\ &\quad + \Theta \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N} P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) \log P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) \\ &= \underset{\Lambda,\gamma}{\operatorname{argmin}}\, \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N} P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) \times \\ &\quad \Big(L(W_{t,n}, W_{t,0}) + \Theta \log P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) \Big) \end{aligned} \tag{3.25}$$

where, Θ is the temperature. Let us denote the above objective function by $F(\Lambda, \gamma, \Theta)$. We seek to find the minimum of $F(\cdot)$ and the corresponding parameters achieving this minimum. Algorithm 1 illustrates Deterministic-Annealing-based model search.

For both cooling and quenching steps, we need to find the model parameters that minimize the respective objective functions. This is achieved by a Gradient-Descent-based optimization method. The model parameters obtained at the end of any particular cooling schedule becomes the initialization for the optimization of the next cooling step (similarly for quenching steps). Thus each time the Gradient-Descent-based optimization method is invoked, the objective function takes a different form based on the parameters and hyper-parameters learned from previous steps.

[8] The assumption here is that the locally optimum solutions do not have a very peaky descent and hence smoothing the distribution smoothes away any locally optimum descent and sharpens the globally optimum descent valley relative to all the others.

Algorithm 3 Deterministic Annealing

Require: $\mathbf{W}$, $\mathbf{q}$, $\Theta_{start} > \Theta_{final}$, H_{min}, γ_{start}, γ_{final}
  $\Theta = \Theta_{start}$, $\gamma = \gamma_{start}$ {C}ooling i.e. decrease Θ with fixed γ
  while $\Theta > \Theta_{final}$ **do**
    $\Lambda_{\Theta,\gamma} = \text{argmin}_{\Lambda} \mathbf{E}_{P_{\Lambda,\gamma}}[L(\cdot,\cdot)] - \Theta \mathbf{H}(P_{\Lambda,\gamma})$
    $\Theta = \alpha(\Theta)$
  end while {Q}uenching i.e. increase γ with fixed $\Theta = 0$
  while $\gamma < \gamma_{final}$ && $H > H_{min}$ **do**
    $\Lambda_{\Theta,\gamma} = \text{argmin}_{\Lambda} \mathbf{E}_{P_{\Lambda,\gamma}}[L(\cdot,\cdot)]$
    $\gamma = \beta(\gamma)$
  end while

3.6.1 Partial Derivatives

To carry out Gradient-Descent-based optimization, we need partial derivatives of the objective function with respect to the model weights. The partial derivative of either $F(\cdot)$ or $E[\cdot](L(\cdot,\cdot))$ requires the partial derivative of $P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)$. The partial derivative of $F(\cdot,\cdot,\cdot)$ with respect to some m^{th} model parameter λ_m is given by:

$$\begin{aligned}
\frac{\partial F(\Lambda,\gamma,\Theta)}{\partial \lambda_m} &= \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N} L(W_{t,n},W_{t,0})\frac{\partial P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m} \\
&\quad +\Theta\frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N}\frac{\partial P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)\log P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m} \\
&= \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N}\frac{\partial P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m}\Big(L(W_{t,n},W_{t,0}) \\
&\quad +\Theta(1+\log P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t))\Big)
\end{aligned}$$

Similarly, the partial derivative of the Expected loss is given by:

$$\frac{\partial \mathbf{E}_{P_{\Lambda,\gamma}}[L]}{\partial \lambda_m} = \frac{1}{|\mathcal{T}|}\sum_{t=1}^{|\mathcal{T}|}\sum_{n=1}^{N} L(W_{t,n},W_{t,0})\frac{\partial P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m} \tag{3.26}$$

We thus need to determine $\frac{\partial P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m}$ in order to solve the above equations.

Before we obtain the above partial derivative, let us establish some notation. Let us represent the numerator term of Eq. 3.23, i.e. the un-normalized probability, by $\tilde{P}_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)$. Let the denominator be denoted by $Z(t)$. Note that $Z(t)$ is simply the sum of un-normalized probabilities over all the N-best hypotheses for a particular speech utterance $\mathbf{o}_t$. Each speech utterance, $\mathbf{o}_t$, will have its corresponding term $Z(t)$. Hence, we can rewrite the log linear equation as:

$$
\begin{aligned}
P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t) &= \frac{\tilde{P}_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{Z(t)} \\
&= \frac{\tilde{P}_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\sum_{k=1}^{N}\tilde{P}_{\Lambda,\gamma}(W_{t,k}|\mathbf{o}_t)}
\end{aligned}
$$

The partial derivative of $P(\cdot)$ can be obtained as:

$$
\frac{\partial P(\cdot)}{\partial \lambda_m} = \frac{\partial P(\cdot)}{\partial \log P(\cdot)}\frac{\partial \log P(\cdot)}{\partial \lambda_m}
$$

Note, the first term on the right hand side is simply $P(\cdot)$. The second term can be obtained as shown below:

$$
\begin{aligned}
&\frac{\partial \log P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m} = \\
&= \frac{\partial \log \tilde{P}_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m} - \frac{\partial \log Z(t)}{\partial \lambda_m} \\
&= \frac{\partial (\sum_{m'=1}^{M}\lambda'_m \gamma q'_m(W_{t,n}|\mathbf{o}_t)))}{\partial \lambda_m} - \frac{\partial \log Z(t)}{\partial \lambda_m} \\
&= \gamma q_m(W_{t,n}|\mathbf{o}_t))) - \frac{1}{Z(t)}\sum_{k=1}^{N}\tilde{P}_{\Lambda,\gamma}(W_{t,k}|\mathbf{o}_t)\gamma q_m(W_{t,k})
\end{aligned}
$$

Thus the partial derivative of $P(\cdot)$ can be obtained as:

$$
\begin{aligned}
\frac{\partial P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)}{\partial \lambda_m} &= \gamma P_{\Lambda,\gamma}(W_{t,n}|\mathbf{o}_t)\Big(q_m(W_{t,n}) \\
&\quad -\frac{1}{Z(t)}\sum_{k=1}^{N}\tilde{P}_{\Lambda,\gamma}(W_{t,k}|\mathbf{o}_t)q_m(W_{t,k})\Big)
\end{aligned}
$$

3.7 Experiments and Results

We evaluated combinations of various models with various optimization methods (suited for the corresponding objective function). We performed experiments on a state-of-the-art speech recognition setup of Wall Street Journal. We evaluated model combination using following model parameter search methods:

1. **Exhaustive Grid Search:** Exhaustive grid search was performed to find model weights when the various sub-models are combined as in Eq. 3.21.

2. **MERT:** Powell's line search [69; 70] was performed to carry out Minimum Error Rate Training (MERT), i.e., to optimize non-smooth 1-*best* error to find model weights when the models are combined as shown in Eq. 3.21. We constrained the search to non-negative model weights. We tried many random initializations and chose the one which gave better performance on the development dataset. See Appendix B for details on Powell's line search.

3. **Minimum Risk:** Our proposed model search was performed to optimize smoothed E-WER to find model weights when the models are combined as shown in Eq. 3.21. Deterministic Annealing was used in conjunction with minimum risk training. We used L-BFGS[9] to solve all the Gradient-Descent-based optimization problems. Discussion of the cooling and quenching schedule of Deterministic Annealing appears in section 3.6. See Appendix B for details on BFGS method.

4. **System Combination:** ROVER [55] is employed to combine 1-*best* output of systems to form a word transition network from which top scoring word sequence is extracted. We used N-best alignment tool of Smith et.al. [57] to do multiple alignment and frequency of occurrence is used to assign weights to words.

We evaluated combinations of various models with various optimization methods (suited for the corresponding objective function). We used 100-best lists from DARPA WSJ'92 and WSJ'93 20K open vocabulary data sets. The acoustic model and baseline bi-gram LM, used to generate the N-best list can be found in [36]. We used the 93et and 93dt sets for development (i.e. for tuning various parameters using various optimization methods) and 92et for evaluation. The development set contained a total of 465 utterances, while the evaluation set contained a total of 333 utterances.

We trained a Kneser-Ney smoothed tri-gram LM built from 70M words of the NYT section of the English Gigaword. We used the 20K vocabulary supplied with the WSJ'93 dataset. The vocabulary as well as the N-best lists were tokenized to match the Penn Treebank style, namely contractions and possessives were separated. The baseline bigram LM and the tri-gram LM are the n-gram LMs.

We also obtained LM scores for N-best lists using syntactic LMs (Filimonov et.al.'s [37]) which use different tagsets to represent syntactic information, namely *head* (which captures dependency), *parent* (constituency), and *SuperARV* (dependency and lexical features). The syntactic models have been trained on data produced from the same NYT section parsed automatically. We refer the reader to [37] for details on the training procedure and the syntactic models. A brief overview of the model can be found in chapter 2.

In all, we used 6 statistical models viz. baseline acoustic model (am), baseline bi-gram language model (bg), re-scoring trigram language model (tg), and three syntactic language models using different tagsets: *head*, *parent*, and *SuperARV* (sarv).

[9] Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is an effective, often used method for solving non-linear optimization problems. L-BFGS [71] is the low memory implementation of BFGS.

Setup	Dev	Eval
am+bg (Exh-Search)	18.2	13.0
am+bg (MERT)	18.2	13.1
am+bg (MinRisk)	18.2	13.0
am+tg (Exh-Search)	17.4	12.2
am+tg (MERT)	17.4	12.4
am+tg (MinRisk)	17.4	12.2
am+head (Exh-Search)	16.8	11.8
am+head (MERT)	16.9	11.9
am+head (MinRisk)	16.8	11.8
am+parent (Exh-Search)	16.9	11.7
am+parent (MERT)	16.9	11.7
am+parent (MinRisk)	16.9	11.7
am+sarv (Exh-Search)	16.9	11.8
am+sarv (MERT)	16.9	11.8
am+sarv (MinRisk)	16.9	11.8

Table 3.3: Word Error Rates (in %) for combination of two models. (am: acoustic model, bg: bigram LM, tg: trigram LM, head: LM trained with head syntactic tag, parent: LM trained with parent syntactic tag and sarv: LM trained with superARV syntactic tag). Exh-Search: Exhaustive grid search for model parameters, MERT: Powell's line search for minimizing 1 best error, MinRisk: Model search method optimizing for minimum expected word error rate.

3.8 Discussion

From Table 3.3, we observe that the performance of both the 1-*best* loss and expected loss, objective function is comparable in almost all the model combinations. Moreover, the performance matches that obtained with exhaustive model parameter search. When only two models are combined, then typically by keeping the weight of one model fixed to 1, the weight of the other model can be appropriately renormalized. This normalization does not affect MAP decoding. Carrying out exhaustive search thus boils down to searching exhaustively in just one dimension by fixing the other dimension to the value of 1.[10]

When the number of models used in combination increases beyond two, then the search has to be carried out exhaustively in two or more dimensions. Although this could be done,

[10]However it should be noted that for MBR [6, pp. 161] [72] the normalization affects the decision boundaries during decoding. Hence even for a simple case of two model combination, the search for parameters needs to be carried out exhaustively in two dimensions.

Setup	Dev	Eval
am+bg+tg (MERT)	16.5	11.7
am+bg+tg (MinRisk)	16.0	**10.9**
am+bg+tg (SysComb)	**14.7**	11.9
am+bg+tg+head (MERT)	15.9	11.2
am+bg+tg+head (MinRisk)	16.0	11.0
am+bg+tg+head (SysComb)	**14.9**	11.2
am+bg+tg+parent (MERT)	16.1	11.2
am+bg+tg+parent (MinRisk)	16.1	**10.9**
am+bg+tg+parent (SysComb)	**15.0**	11.2
am+bg+tg+sarv (MERT)	16.0	11.4
am+bg+tg+sarv (MinRisk)	16.0	**11.1**
am+bg+tg+sarv (SysComb)	**15.0**	11.2
am+bg+tg+head+parent (MERT)	16.0	10.9
am+bg+tg+head+parent (MinRisk)	15.8	**10.4**
am+bg+tg+head+parent (SysComb)	**15.3**	11.4
am+bg+tg+head+parent+sarv (MERT)	16.0	11.1
am+bg+tg+head+parent+sarv (MinRisk)	**15.6**	**10.5**
am+bg+tg+head+parent+sarv (SysComb)	15.8	11.2

Table 3.4: Word Error Rates (in %) for combination of three or more models. (am: acoustic model, bg: bigram LM, tg: trigram LM, head: LM trained with head syntactic tag, parent: LM trained with parent syntactic tag), sarv: LM trained with superARV syntactic tag). MERT: Powell's line search for minimizing 1 best error, MinRisk: Model search method optimizing for minimum expected word error rate. SysComb: ROVER technique for combining system outputs in a system combination framework.

it would be prohibitively slow, and so we prefer to avoid exhaustive search. In the case of two model combinations, we observed that the objective function was nearly convex with respect to the model parameters; hence, a hill climbing method indeed finds a globally optimum solution. However, when the number of models increases beyond two, the objective function becomes more and more non-convex. From Table 3.4, we can see that the model search using 1-best loss as the objective function fails to find the globally optimum solution. However, using E-WER as the objective function, regularized with the entropy, we obtain a much better model that generalizes well even on the evaluation data set. Our proposed method consistently performs better than MERT. In the case where we combine all the statistical models (last row of Table 3.4), using our proposed approach, we obtain about a $\mathbf{0.6}\%$ absolute WER improvements when compared to the 1-*best* loss objective function based model search. System combination technique too is effective but not consistent. The performance of system combination on the evaluation section does not change much beyond 3 model result. Moreover, on the development section, system combination performance is not consistent.

It is important to note that we reconfirm Beyerlein's observation [61] that by combining

complementary models in a unified framework, one can achieve much better recognition performance than the best performance obtained by any of the sub-models. In our work when we combine all the models together, we obtain a 1.3% absolute WER decrease from the WER obtained using the best sub-model (i.e. the syntactic LM using the tag *parent*). We also show that model combination technique is far more superior than system combination technique.

In conclusion, when we have complementary knowledge sources, by way of model combination, it is possible to achieve improved recognition performance. By having a smooth objective function like expected loss (E-WER in our case), the model search can be done in a more principled manner resulting in a much better, generalized model.

Chapter 4

Iterative Decoding

In the previous chapter, we discussed how long span models and their combination with n-gram models can effectively improve the accuracy of a speech recognition system. In this chapter, we will focus on a re-scoring technique which enables the deployment of these complex models on big word graph structures which are the output of the first pass decoding. The proposed method thus facilitates the use of full blown model without compromising the performance.

Noam Chomsky argued that n-grams cannot learn long-distance dependencies that span over more than n words [73, pp. 13, ch 2]. While that might seem obvious in retrospect, there was a lot of excitement at the time over the Shannon-McMillan-Breiman Theorem [74] which was interpreted to say that, in the limit, under just a couple of minor caveats and a little bit of not-very-important fine print, n-gram statistics are sufficient to capture all the information in a string (such as an English sentence). Chomsky realized that while that may be true in the limit, n-grams are far from the most parsimonious representation of many linguistic facts. In a practical system, we will have to truncate n-grams at some (small) fixed n (such as trigrams or perhaps 5-grams). Truncated n-gram systems can capture many agreement facts, but not all [75].

By long-distance dependencies, we mean facts like agreement and collocations that can span over many words. With increasing order of n-gram models we can, in theory, capture more regularities in the language. In addition, if we can move to more general models then we could hope to capture more, as well. However, due to data sparsity, it is hard to estimate a robust n-gram distribution for large values of n (say, $n > 10$) using the conventional Maximum Likelihood techniques, unless a more robust technique is employed for modeling which generalizes well on unseen events. Some of these well known long span / complex language models which have shown to perform very well on many speech tasks include: structured language model [33; 76; 36; 37], latent semantic analysis language model [77], topic mixture language models [78], whole sentence exponential language models [79; 80], feedforward neural networks [39], recurrent neural network language models [41], among many others.

Although better modeling techniques can now capture longer dependencies in a lan-

guage, their incorporation in decoders of speech recognition or machine translation systems becomes computationally challenging. Due to the prohibitive increase in the search space of sentence hypotheses (or longer length word sub sequences), it becomes challenging to use a long span language model in the first pass decoding. A word graph (word lattices for speech recognition systems and hypergraphs for machine translation systems), encoding exponential number of hypotheses is hence outputted at the end of the first pass, on which a sophisticated and complex language model is deployed for re-scoring. However, sometimes even re-scoring of this refined search space can be computationally expensive due to an explosion of the state space (huge number of state splitting is necessary to maintain a unique context at every node for unambiguous prediction of next word by a LM).

In Chapter 5 we will present methods to tackle the problem of incorporating a long span LM for speech recognition by variationaly inferring a tractable model such that it comes closest to the long span model (closest in terms of Kullback Leibler Divergence). The tractable substitute will then be used in the first pass speech recognition systems. In this chapter we are proposing an approach that keeps the model intact, thus enabling the use of full blown model for re-scoring. With this approach, we can achieve full lattice re-scoring with a complex model, at a cost more than 20 times less than of a naive brute force approach that is commonly used today.

The rest of the chapter is organized as follows: In Section 4.1 we discuss three topological structures of search spaces and the corresponding re-scoring techniques suited for them. We then describe and demonstrate our proposed technique in Sec. 4.2. We present experimental results in Sec. 4.3 followed by conclusions and some remarks in Sec. 4.4.

4.1 Standard Search Topological Structures and Approaches for Rescoring them

In this section, we will discuss three main topological structures of search spaces and the corresponding re-scoring methods suited for each of them. We will begin with a discussion about word lattice, followed by confusion networks and finally concluded by N-best lists.

4.1.1 Word Lattice

A word lattice, $\mathcal{L}$, obtained at the output of the first pass decoding, encodes exponential number (exponential in the number of states (nodes) present in the lattice) of hypotheses in a very compact data structure. It is a directed acyclic graph $G = (\mathcal{V}, \mathcal{E}, n_s, N_e)$, where $\mathcal{V}$ and $\mathcal{E}$ denote set of vertices (nodes / states) and edges (arcs / links), respectively. n_s and N_e denote the unique start state and set of end states.

A path, π, in a lattice is an element of $\mathcal{E}^*$ with consecutive transitions. We will denote the origin / previous state of this path by $p[\pi]$ and destination / next state of this path by

$n[\pi]$. A path, π is called a *complete path* if $p[\pi] = n_s$ and $n[\pi] \in N_e$. A path, π, is called a *partial path* if $p[\pi] = n_s$ but $n[\pi]$ may or may not belong to N_e. A path, π, is called a *trailing path* if $p[\pi]$ may or may not be equal to n_s and $n[\pi] \in N_e$. We will also denote the time stamp at the start of the path by $T_s[\pi]$ and the time stamp at the end of the path by $T_e[\pi]$. Since there are nodes attached to the start and end of any path, we will denote the time stamp at any node $u \in \mathcal{V}$ by $T[u]$. Associated with every path, π, is also a word sequence $W[\pi] \in \mathcal{W}^*$, where $\mathcal{W}$ is the vocabulary used during speech recognition. For the sake of simplicity, we will distinguish word sequence of length 1 from the word sequences of length greater than 1 by using lower and upper casing i.e. $w[\cdot]$ and $W[\cdot]$ respectively.

The acoustic likelihood of the path $\pi \in \mathcal{E}^*$ is then given as:

$$A[\pi] = \prod_{j=1}^{|\pi|} P(\mathbf{a}_j | w[\pi_j])$$

where $\forall j \in \{1, 2, \ldots, |\pi|\} \quad \pi_j \in \mathcal{E}$, $\pi = \odot_{j=1}^{|\pi|} \pi_j$[1] and $P(\mathbf{a}_j | w[\pi_j])$ is the acoustic likelihood of the acoustic substring $\mathbf{a}_j$, spanning between $T_s[\pi_j]$ and $T_e[\pi_j]$, conditioned on the word $w[\pi_j]$ associated with the edge π_j. Similarly, the language model score of the path π is given as:

$$L[\pi] = \prod_{j=1}^{|\pi|} P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}])$$

where $P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}])$ is the m-th order Markov approximation for estimating the probability of a word given the context upto that point. The speech recognizer, which uses m-th order Markov LM for first pass recognition, imposes a constraint on the word lattice such that at each state there exists an unambiguous context of consecutive $m-1$ words.

A first pass output is then a path π^* having Maximum a Posterior (MAP) probability.[2] Thus π^* is obtained as:

$$\pi^* = \underset{\substack{\pi: p[\pi] = n_s \\ n[\pi] \in N_e}}{\arg\max} A[\pi]^\gamma L[\pi],$$

where γ is the scaling parameter needed to balance the dynamic variability between the distributions of acoustic and language model [81]. Efficient algorithms such as single source shortest path [82] can be used for finding out the MAP path.

Under a new n-gram Language Model, rescoring involves replacing the existing language model scores of all paths π. If we denote the new language model by L_{new} and the

[1] We will use $\odot$ symbol to denote concatenation of paths or word strings.

[2] Note that an asterisk here connotes that the path is *optimal* under some model. This should not be confused with the Kleene stars appearing as superscripts for $\mathcal{E}$ and $\mathcal{W}$, which serves the purpose of regular expression implying 0 or many occurances of the element of $\mathcal{E}$ and $\mathcal{V}$ respectively.

correspondingly the score of the path π by $L_{new}[\pi]$, then it is simply obtained as:

$$L_{new}[\pi] = \prod_{j=1}^{|\pi|} P(w[\pi_j]|w[\pi_{j-1}], \ldots, w[\pi_{j-n+1}])$$

where $P(w[\pi_j]|w[\pi_{j-1}], \ldots, w[\pi_{j-n+1}])$ is the n-th order Markov approximation for estimating the probability of a word given the unambiguous context of $n-1$ words under the new rescoring LM. If the Markov rescoring n-gram LM needs a bigger context for the task of prediction (i.e. $n > m$, where $m-1$ is the size of the unambiguous context maintained at every state of the word lattice), then each state of the lattice has to be split until an unambiguous context of length as large as that required by the new re-scoring language model is maintained. The best path, π^* is then obtained as:

$$\pi^* = \underset{\substack{\pi: p[\pi]=n_s \\ n[\pi]\in N_e}}{\arg\max}\, A[\pi]^{\eta} L_{new}[\pi],$$

where η acts as the new scaling parameter which may or may not be equal to the old scaling parameter γ.

It should be noted that if the rescoring LM needs a context of the entire past in order to predict the next word, then the lattice has to be expanded by splitting the states many more times. This usually blows up the search space even for a reasonably small number of state splitting iterations. When the task is to do rescoring under a long span LM, such as RNN-LM, then *exact* lattice re-scoring option is not feasible. In order to tackle this problem, a suboptimal approach via N-best list rescoring is utilized. The details of this method are presented in subsection 4.1.3.

4.1.2 Confusion Network

Consensus decoding [59] aims at directly minimizing the word error rate of the recognition hypothesis.[3] In this particular type of decoding, word lattices are converted into confusion networks. A confusion network is a concatenation of confusion bins and each confusion bin contains word hypotheses competing with each other at any time instant. Thus a confusion network (CN) is a linear network composed of K confusion bins $\{B_1, \ldots, B_K\}$ concatenated linearly. For a word, v, in some k^{th} confusion bin of the confusion network, its posterior probability given the acoustics, $\mathbf{A}$, i.e. $P(v|\mathbf{A})$ is computed by summing the

[3]Maximum A-Posteriori (MAP) probability path found in the word lattice is optimal for minimizing the sentence error rate. However, the metric we care about is word error rate and hence an objective function needs to explicitly aim for minimizing the word error rate. Minimum Bayes risk (MBR) [6, pp. 161] is one such objective function which, under true distribution, guarantees minimization of the general loss function employed. However, finding an optimal path under this objective function becomes computationally complex if the search space is a word lattice. Consensus decoding aims to solve this problem by simplifying the word lattice structure by transforming it into a confusion network.

posterior probabilities of all of the complete paths in the word lattice such that they all share this word. Note that all these complete paths need not pass through the exact same realization of the word under consideration, but through its *any* realization. However, the same word could be spoken at two different time instants and hence a constraint based on the time of transcription of the word allows the transformation process to consider only those paths which not only share this word but also do so within the same time window. Thus the process of confusion network creation identifies sets of non-overlapping time windows such that the number of confusion bins eventually formed equals the number of such time windows identified in the lattice. Let us hence represent a time window by U_k, where the subscript, k, will denote the index of the confusion bin with which it is associated. The posterior probability of any word v in the k^{th} confusion bin is then obtained as:

$$P(v|\mathbf{A}) = \sum_{\substack{\pi\in\mathcal{E}^*: \\ p[\pi]=n_s, n[\pi]\in N_e \\ \exists i\in\{1,\ldots,|\pi|\} s.t. w[\pi_i]=v \\ \{T[p[\pi_i]],T[n[\pi_i]]\}\in U_k}} \frac{A[\pi]^\gamma L[\pi]}{Z}$$

where Z is the normalization constant given by:

$$Z = \sum_{\substack{\pi\in\mathcal{E}^* \\ p[\pi]=n_s, n[\pi]\in N_e}} A[\pi]^\gamma L[\pi]$$

It should be noted that since *all* complete paths of the word lattice pass through each time window U_k, the sum of the posterior probabilities of all the words in any k^{th} confusion bin will be 1.

The words in the confusion bin are ordered according to their posterior probability and for any bin these posterior probabilities sum to 1. $\forall k \in \{1, 2, \ldots, K\}$, $\forall j_k \in \{1, 2, \ldots, |B_k|\}$, where $|B_k|$ is the size of the k^{th} confusion bin, following two conditions are imposed:

$$P(w_{k,j_k}|\mathbf{A}) \geq P(w_{k,j_k+1}|\mathbf{A}) \tag{4.1}$$

$$\sum_{j_k=1}^{|B_k|} P(w_{k,j_k}|\mathbf{A}) = 1 \tag{4.2}$$

where w_{k,j_k} is the j_k^{th} word in the k^{th} confusion bin.[4]

By way of its construction, a confusion network induces many paths which were originally not present in its corresponding input word lattice. Due to the inclusion of these

[4]The condition of Eqn. 4.1 is not explicitly imposed during the transformation, but will be implicitly imposed for the purpose of simplifying the flow of the algorithms described in next few sections.

new paths, it has been observed that the oracle word error rate[5] of confusion networks is sometimes lower than that of the corresponding word lattices.

Consensus decoding output is the sequence of highest posterior probability words taken from each confusion bin. Thus the optimal consensus decoding word sequence under baseline acoustic and language model is obtained as:

$$W^* = w_{1,1} \cdot w_{2,1} \cdot \ldots \cdot w_{K,1} \equiv \odot_{k=1}^{K} w_{k,1}$$

In the previous section we discussed that for the purpose of language model re-scoring of word lattices, lattice dynamic programming re-scoring method can be used where the baseline language model scores in the word lattices are replaced by scores obtained from stronger language model following which the viterbi path is found in the re-scored lattice. Since CNs have posterior scores on the words and explicit language model and acoustic model score is lost in the transformation, the language model re-scoring can be achieved either by (a) composing the Finite State Machine (FSM) representation of the LM[6] and CN[7], combining the posterior scores and new language model scores and then extracting the FSM least cost (viterbi) path *or* (b) by extracting N-best lists from confusion networks (described in sub-section 4.1.3), recomputing the new language model scores and combining them either with the posterior scores or with the acoustic scores for each individual word string [8] and picking the top scoring hypothesis. Re-scoring using FSM representation (choice (a) above) is very efficient as compared to the N-best list re-scoring (choice (b) above), however it comes at a price as explained next:

In the FSM framework for confusion network re-scoring, the incorporation of acoustic information may not be easy and hence may not be a suitable platform for re-scoring under standard criteria such as MAP or MBR. Also, when the external knowledge source is other than a simple n-gram language model (sentence level knowledge sources, longer dependency models, parsers etc), then its conversion into an FSM may not be always possible. Even for n-gram LMs, the composition can blow up in memory if there are huge number of n-grams with higher orders. Hence in such cases, one has to resort to N-best list re-scoring methods. However, as we will demonstrate in the next section, N-best list (for small N) re-scoring results in sub-optimal performance and requires huge exploration of the search space for optimal solutions. Our proposed method aims to find better solutions (better than N-bests) in considerably lesser search efforts. Details of the proposed method will follow after we describe the N-best list structure.

[5] Oracle WER is the WER of the hypotheses in lattices/CN such that they have the least edit distance with respect to the corresponding true transcriptions.

[6] Interested readers are encouraged to refer to [83] for details on how an n-gram LM can be converted into a weighted finite state automata.

[7] A CN is a directed acyclic graph and its conversion into an FSM is trivial.

[8] Acoustic scores for hypotheses can be obtained by force-aligning [10, pp. 45] them with the acoustics. However, this process is time consuming and slows down the decoding considerably.

4.1.3 N-Best Lists

N-best list re-scoring is a popular way to capture some long-distance dependencies, though the method can be slow and it can be biased toward the weaker language model that was used in the first pass. In this section, we will discuss the topological structure of N-best lists when they are extracted from the word-lattices. N-best lists extracted from CNs would have similar properties barring few subtle differences due to the representation of the baseline model scores in lattices and confusion networks. These differences are not significant as far as this discussion is concerned. We will hence not explicitly discuss the N-best lists structure extracted from CNs but will discuss the structure of this search space when they are extracted from a word lattice.

Given a word lattice, $\mathcal{L}$, top N paths $\{\pi_1, \ldots, \pi_N\}$ are extracted such that their joint likelihood under the baseline acoustic and language models are in descending order i.e. that:

$$A[\pi_1]^\gamma L[\pi_1] \geq A[\pi_2]^\gamma L[\pi_2] \geq \ldots \geq A[\pi_N]^\gamma L[\pi_N]$$

Efficient algorithms exist for extracting N-best paths from word lattices [84; 85]. If a new language model, L_{new}, is provided, which now need not be restricted to finite state machine family, then that can be deployed to get the score of the entire path π. If we denote the new LM scores by $L_{new}[\cdot]$, then under N-best list paradigm, optimal path $\tilde{\pi}$ is found out such that:

$$\tilde{\pi} = \underset{\pi \in \{\pi_1, \ldots, \pi_N\}}{\arg\max} \; A[\pi]^\eta L_{new}[\pi], \tag{4.3}$$

where η acts as the new scaling parameter which may or may not be equal to γ. If $N \ll |\mathcal{L}|$ (where $|\mathcal{L}|$ is the total number of complete paths in word lattice, which are exponentially many), then the path obtained using (4.3) is not guaranteed to be optimal (under the rescoring model). If one were to enumerate all paths from the lattice then under this huge search space, rescoring is guaranteed to yield the optimal performance. However, it is intractable to enumerate all possible hypotheses and rescore them with a long span language model, mainly because there are exponentially many hypotheses possible in a word lattice. The short list of hypotheses so used for re-scoring would yield suboptimal output if the best path π^* (according to the new model) is not present among the top N candidates extracted from the lattice. The N-best hypotheses are *best* with respect to the model generating them and they need not be *best* with respect to any other rescoring model. The search space is also said to be *biased* towards a weaker model mainly because the N-best lists are representative of the model generating them. To illustrate the idea, we demonstrate below a simple analysis on a relatively easy task of speech transcription on WSJ data[9]. The recognizer made use of a bi-gram LM to produce lattices and hence N-best lists. Each hypotheses in this set got a rank with the top most and highest scoring hypothesis getting a rank of 1, while the bottom most hypothesis getting a rank of N. We then re-scored these hypotheses with a better language model (either with a higher order Markov LM i.e. a trigram LM (*tg*)

[9]The details of the experimental setup are presented in Section 3.7 of Chapter 3

or the log linear combination of n-gram models and syntactic models (*n-gram+syntactic*) and re-ranked the hypotheses to obtain their new ranks. We then used Spearman's rank correlation factor, ρ, which takes values in $[-1, +1]$, with -1 meaning that the two ranked lists are negatively correlated (one list is in a reverse order with respect to the other list) and $+1$ meaning that the two ranked lists are positively correlated (the two lists are exactly the same). Spearman's rank correlation factor is given as:

$$\rho = 1 - \frac{6 \sum_{n=1}^{N} d_n^2}{N(N^2 - 1)}, \tag{4.4}$$

where d_n is the difference between the old and new rank of the n^{th} entry (in our case, difference between $n (\in \{1, 2, \ldots, N\})$ and the new rank which the n^{th} hypothesis got under the rescoring model).

Table 4.1 shows how the correlation factor drops dramatically when a better and a complementary LM is used for re-scoring, suggesting that the N-best lists are heavily biased towards the starting models. Huge re-rankings suggests there is an opportunity to improve and also a need to explore more hypotheses, i.e. beyond N-best lists.

Model	(ρ)	**WER (%)**
bg	1.00	18.2%
tg	0.41	17.4%
n-gram+syntactic	0.33	15.8%

Table 4.1: Spearman Rank Correlation on the N-best list extracted from a bi-gram language model (*bg*) and re-scored with relatively better language models including, trigram LM (*tg*), and the log linear combination of n-gram models, and syntactic models (*n-gram+syntactic*). With a bigger and a better LM, the WER decreases at the expense of huge re-rankings of N-best lists, only suggesting the fact that N-best lists generated under a weaker model, are not reflective enough of a relatively better model.

In the next section, we propose an algorithm which tries to bridge the gap between the word lattice/CN and N-best lists, in the sense that the search does not get restricted to top N-best paths alone, and also overcomes the problem of search space getting *biased* towards the starting weaker model.

4.2 Proposed Approach for Rescoring

The gist of the algorithm is to divide the global search problem of finding a better hypothesis into many small local search problems. This is a hill climbing technique and like any other hill climbing technique, the performance of these algorithms is highly dependent on the choice of the local neighborhoods on which local search is performed. While the

local neighborhoods are naturally formed in CNs, they have to be specifically defined in other more general structures such as word lattices. For CNs, we propose to define local neighborhood based on confusion bins and for the word lattices, we define them based on the notion of finding islands of confusability.

Richardson et.al. [86] had previously demonstrated an approach to hill climbing on word lattices. In their work, the choice of local neighborhood was based on 1 word edit distance of a hypothesis under consideration. The method was demonstrated for a moderately sized vocabulary speech task, however, the method failed to beat the performance of moderately large N-best. We think this is mainly due to the fact that the limited scope of local neighborhood constrained the search to a very narrow beam. Selecting a local neighborhood based on more word edit distances although increases the chances of finding a better hypothesis, it also increase the size of the search space and does not render the algorithm any faster. Note that, if we define the local neighborhood based on $2 \times L$[10] word edit distances, where L is the maximum length of any hypothesis possible in the word lattice, then the hill climbing is guaranteed to find optimal solution. Unfortunately, going from 1 edit distance to 2 edit distance itself increases the computational complexity a lot and hence we need a principled way to define local neighborhoods so that at any time instant, we analyze all possible sub-sequences competing with each other.

A high level idea of our proposed approach is to identify *islands of confusability* in the word lattice and replace the problem of global search over word lattice by series of local search problems over these islands in an iterative manner. The motivation behind this strategy is the observation that the recognizer produces bursts of errors such that they have a temporal scope. The recognizer output (sentence hypotheses) when aligned together typically shows a pattern of confusions both at the word level and phrase level. Regions where there are singleton words competing with one another (reminiscent of a confusion bin of a CN), choice of 1 word edit distance works well for the formation of local neighborhood. Regions where there are phrases competing with other phrases, choice of variable length neighborhood works well. Typical example of some phrase hypotheses in a typical confusable islands is shown below (Note that, both these phrases have the same start time and same end time):

- `motorcyclists jumped`
- `motor cycle was dumped`

While such islands are implicitly formed in a confusion network in the form of confusion bins, we have to explicitly form them for word graphs. Hence before we describe the workings of algorithms for re-scoring, we will first describe the process, virtue of which, we can cut the lattice to form many self contained smaller sized sub lattices. We will then follow it up with the illustration of the proposed algorithm for word lattices in Section 4.2.2 and then confusion networks in Section 4.2.3.

[10] In the worst case, one would require L edits to delete and extra L edits to insert new words.

4.2.1 Islands of Confusability

As said in previous sections, confusion bins of the confusion networks form the local neighborhood needed for the proposed algorithm. However, for word lattices, these local neighborhoods have to be explicitly formed. We propose to do so via finding islands of confusability. These islands of confusability will be formed such that original attributes of the words do not get altered (by attributes, we mean the acoustic score, language model score and the timing information associated with the word of a lattice. Lattice to CN transformation alters these attributes requiring for an extra round of post processing to get this information back during re-scoring.)

We will continue to follow the notation introduced in section 4.1.1. Before we define the procedure for cutting the lattice into many small self contained lattices, we will define some more terms necessary for the ease of understandability of the algorithm. For any node $v \in \mathcal{V}$, we define forward probability, $\alpha(v)$, as the probability of any partial path $\pi \in \mathcal{E}^*$, s.t. $p[\pi] = n_s, n[\pi] = v$ and it is given as:

$$\alpha(v) = \sum_{\substack{\pi \in \mathcal{E}^* \\ s.t. p[\pi]=n_s, n[\pi]=v}} A[\pi]^\gamma L[\pi] \tag{4.5}$$

Similarly, for any node $v \in \mathcal{V}$, we define the backward probability, $\beta(v)$, as the probability of any trailing path $\pi \in \mathcal{E}^*$, s.t. $p[\pi] = v, n[\pi] \in N_e$ and it is given as:

$$\beta(v) = \sum_{\substack{\pi \in \mathcal{E}^* \\ s.t. p[\pi]=v, n[\pi] \in N_e}} A[\pi]^\gamma L[\pi] \tag{4.6}$$

If we define the sum of joint likelihood under the baseline acoustic and language models of all paths in the lattice by Z, then it can simply be obtained as: $Z = \sum_{u \in N_e} \alpha(u) = \beta(n_s)$

In order to cut the lattice, we want to identify sets of nodes, $S_1, S_2, \ldots, S_{|\mathcal{S}|}$ such that for any set $S_i \in \mathcal{S}$ following conditions are satisfied:

1. For any two nodes $u, v \in S_i$ we have that: $T[u] = T[v]$. We will define this common time stamp of the nodes in the set by $T[S_i]$.

2. $\nexists\, \pi \in \mathcal{E}$ such that $T_s[\pi] < T[S_i] < T_e[\pi]$.

The first property can be easily checked by first pushing states into a linked list associated with each time marker (this can be done by iterating over all the states of the graph) then iterating over the unique time markers and retrieving back the nodes associated with it.The second property can be checked by first iterating over the unique time markers and for each of the marker, iterating over the arcs and terminating the loop as soon as some arc is found out violating property 2 for the specific time marker. Thus the time complexity for checking property 1 is $O(|\mathcal{V}|)$ and that for property 2 is $O(|\mathcal{T}| \times |\mathcal{E}|)$, where $|\mathcal{T}|$ is the total number of unique time markers. Usually $|\mathcal{T}| \ll |\mathcal{E}|$ and hence the time complexity for checking

property 2 is almost linear in the number of edges. Thus effectively, the time complexity for cutting the lattice is $O(|\mathcal{V}| + |\mathcal{E}|)$.

Having formed such sets, we can now cut the lattice at time stamps associated with these sets i.e. that: $T[S_1], \ldots, T[S_{|\mathcal{S}|}]$. It can be easily seen that the number of sub lattices, C, will be equal to $|\mathcal{S}| - 1$.We will identify these sub lattices as $\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C$. At this point, we have not formed self contained lattices yet by simply cutting the parent lattice at the cut points.

Once we cut the lattice at these cut points, we implicitly introduce many new starting nodes and ending nodes for any sub lattice. We will refer to these nodes as exposed starting nodes and exposed ending nodes. Thus for some j^{th} sub lattice, $\mathcal{L}_j$, there will be as many new exposed starting nodes as there are nodes in the set S_j and as many exposed ending nodes as there are nodes in the set S_{j+1}. In order to make these sub lattices consistent with the definition of a word lattice (see Sec. 4.1.1), we unify all the exposed starting nodes and exposed ending nodes. To unify the exposed starting nodes, we introduce as many *new* edges as there are nodes in the set S_j such that they have a common starting node, $n_s[\mathcal{L}_j]$, (newly created) and distinct ending nodes present in S_j. To unify the exposed ending nodes of $\mathcal{L}_j$, we introduce as many *new* edges as there are nodes in the set S_{j+1} such that they have distinct starting nodes present in S_{j+1} and a common ending node $n_e[\mathcal{L}_j]$ (newly created). From the totality of these new edges and nodes along with the ones already present in $\mathcal{L}_j$ forms an induced directed acyclic sub-graph $G[\mathcal{L}_j] = (\mathcal{V}[\mathcal{L}_j], \mathcal{E}[\mathcal{L}_j], n_s[\mathcal{L}_j], n_e[\mathcal{L}_j])$.

For any path $\pi \in \mathcal{E}[\mathcal{L}_j]$ such that $p[\pi] = n_s[\mathcal{L}_j]$ and $n[\pi] \in S_j$, we assign the value of $\alpha(n[\pi])$ to denote the joint likelihood $A[\pi]^\gamma L[\pi]$ and assign $epsilon$ for word associated with these edges i.e. $w[\pi]$. We assign $T[S_j] - \delta T$ to denote $T_s[\pi]$ and $T[S_j]$ to denote $T_e[\pi]$. Similarly, for any path $\pi \in \mathcal{E}[\mathcal{L}_j]$ such that $p[\pi] \in S_{j+1}$ and $n[\pi] = n_e[\mathcal{L}_j]$, we assign the value of $\beta(p[\pi])$[11] to denote the joint likelihood $A[\pi]^\gamma L[\pi]$ and assign $epsilon$ for word associated with these edges i.e. $w[\pi]$. We assign $T[S_{j+1}]$ to denote $T_s[\pi]$ and $T[S_{j+1}] + \delta T$ to denote $T_e[\pi]$. This completes the process and we obtain self contained lattices, which if need be, can be independently decoded and/or analyzed.

4.2.2 Iterative Decoding on Word Lattices

Once we have formed the self contained lattices, $\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C$, where C is the total number of sub lattices formed, then the idea is to divide the re-scoring problem into many small re-scoring problems carried over the sub lattices one at a time by fixing single best paths from all the remaining sub lattices.

Algorithm 4 below illustrates the steps. The inputs to the algorithm are the sub lattices (produced by cutting the parent lattice generated under some Markov n-gram LM) and a new rescoring LM, which now need not be restricted to finite state machine family. The output of the algorithm is a word string, $\mathbf{W}^*$, such that it is the concatenation of final decoded word strings from each sub lattice. Thus if we denote the final decoded path (under

[11] The values of $\alpha(\cdot)$ and $\beta(\cdot)$ are computed under parent lattice structure.

some decoding scheme, which will become apparent next) in the j^{th} sub lattice by π_j^* and the concatenation symbol by '$\cdot$', then $\mathbf{W}^* = W[\pi_1^*] \cdot W[\pi_2^*] \cdot \ldots \cdot W[\pi_C^*] = \odot_{j=1}^{C} W[\pi_j^*]$.

Algorithm 4 Algorithm for Iterative Decoding on word lattices.

Require: $\{\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C\}$, L_{new}
  PrevHyp $\leftarrow$ null
  CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  while PrevHyp $\neq$ CurrentHyp **do**
    for $i \leftarrow 1 \ldots C$ **do**
      $\hat{\pi}_i \leftarrow \underset{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}}{\operatorname{argmax}} \Big(L_{new}[\hat{\pi_1} \cdot \ldots \cdot \pi_i \cdot \ldots \cdot \hat{\pi_k}] \times A[\pi_i]^{\eta} \prod_{\substack{j=1 \\ j \neq i}}^{k} A[\hat{\pi_j}]^{\eta} \Big)$
    end for
    PrevHyp $\leftarrow$ CurrentHyp
    CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  end while
  $\forall j \in \{1, 2, \ldots, C\} \quad \pi_j^* \leftarrow \hat{\pi}_j$

The algorithm is initialized by setting **PrevHypo** to null and **CurrHypo** to the concatenation of 1-best output from each sub lattice. During the initialization step, each sub lattice is analyzed independent of any other sub lattice and under the baseline acoustic scores and baseline n-gram LM scores, 1-*best* path is found out. Thus if we define the best path under baseline model in some j^{th} sub-lattice by $\hat{\pi}_j$, then it is found out simply as:

$$\hat{\pi}_j = \underset{\substack{\pi_j \in \mathcal{E}_j^*: \\ p[\pi_j]=n_s[\mathcal{L}_j] \\ n[\pi]=n_e[\mathcal{L}_j]}}{\arg\max} \; A[\pi_j]^{\gamma} L[\pi_j],$$

CurrHypo is then initialized to: $W[\hat{\pi}_1] \cdot W[\hat{\pi}_2] \cdot \ldots \cdot W[\hat{\pi}_C]$. The algorithm then runs as long as CurrHypo is not equal to PrevHypo. In each iteration, the algorithm sequentially re-scores each sub-lattice by keeping the surrounding context fixed. Once all the sub lattices are re-scored, that constitutes one iteration. At the end of each iteration, CurrHypo is set to the concatenation of 1 best paths from each sub lattice while PrevHypo is set to the old value of CurrHypo. Thus if we are analyzing some i^{th} sub-lattice in some iteration, then 1-best paths from all but this sub-lattice is kept fixed and a *new* 1-*best* path under the re-scoring LM is found out as shown below.

$$\hat{\pi}_i = \underset{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}}{\text{argmax}} \Big(L_{new}[\hat{\pi_1} \cdot \ldots \cdot \pi_i \cdot \ldots \cdot \hat{\pi_k}] \times A[\pi_i]^\eta \prod_{\substack{j=1 \\ j \neq i}}^{k} A[\hat{\pi_j}]^\eta \Big) \tag{4.7}$$

where the left context: $W[\hat{\pi}_1] \cdot W[\hat{\pi}_2] \cdot \ldots \cdot W[\hat{\pi}_{i-1}]$ and right context $W[\hat{\pi}_{i+1}] \cdot W[\hat{\pi}_{i+2}] \cdot \ldots \cdot W[\hat{\pi}_C]$ is fixed during this local rescoring. η is the new fudge factor which may or may not be equal to γ depending upon the dynamic scale variability between two language models: baseline starting model L and new rescoring model L_{new}.

Since the cutting of parent lattices produce many small lattices with considerably lesser number of nodes, in practice, an exhaustive search for the 1-best hypothesis can be carried out via N-best list. It is not hard to see that the likelihood of the output under the new re-scoring model is guaranteed to increase monotonically after every decoding step. Theorem 4.2.1 formalizes this statement below:

Theorem 4.2.1 *The likelihood of hypothesis under new re-scoring model monotonically increases in each step.*

Proof Let the concatenation of best paths (under baseline models) fed as an input to the algorithm be represented by $\Pi(0)$. Let the output obtained after re-scoring the first sub-lattice in the first iteration be denoted by $\Pi_1(1)$. By construction, $\Pi_1(1)$ will be obtained by perturbing $\Pi(0)$ with complete paths from first sub-lattice, $\mathcal{L}_1$, such that:

$$L_{new}[\Pi_1(1)]A^\eta[\Pi_1(1)] \geq L_{new}[\Pi(0)]A^\eta[\Pi(0)]$$

Hence in the first step, the likelihood under the new re-scoring model, $L_{new}[\cdot]$, is guaranteed to increase. Similarly, if we denote the concatenation of best paths obtained in i^{th} iteration after re-scoring some c^{th} sub-lattice by $\Pi_c(i)$, then by way of construction, the output obtained after re-scoring $(c+1)^{th}$ sub-lattice in i^{th} iteration, denoted by $\Pi_{c+1}(i)$, will be obtained by perturbing $\Pi_c(i)$ with complete paths from sub-lattice, $\mathcal{L}_{c+1}$, such that:

$$L_{new}[\Pi_{c+1}(i)]A^\eta[\Pi_{c+1}(i)] \geq L_{new}[\Pi c(i)]A^\eta[\Pi c(i)]$$

Thus the likelihood of output in any iteration monotonically increases. Since the output obtained after re-scoring last sub-lattice, $\mathcal{L}_C$, in i^{th} iteration becomes the starting point for the first sub-lattice in $(i+1)^{th}$ iteration, using similar reasoning as above, we can see that:

$$L_{new}[\Pi_1(i+1)]A^\eta[\Pi_1(i+1)] \geq L_{new}[\Pi_C(i)]A^\eta[\Pi_C(i)]$$

The likelihood of output under the re-scoring model, in any step is thus greater than or equal to the likelihood of the output in any of the previous steps. ∎

It should be, however, noted that this is a greedy hill climbing heuristic that may not converge to the globally optimum solution. Appendix D illustrates the working of Iterative Decoding on a toy setup for illustration purposes.

4.2.2.1 Entropy Pruning

In this sub-section, we will discuss a speed up technique based on entropy of the lattice. Entropy of a lattice reflects the confidence of the recognizer in recognizing the acoustics. Based on the observation that if the N-best list / lattice generated under some model has a very low entropy, then the Spearman's rank correlation factor, ρ (Eqn. 4.4), tends to be higher even when the N-best lists / lattice is re-ranked with a bigger and a better model. A low entropy under the baseline model only reflects the confidence of the recognizer in recognizing the acoustic. Table 4.2 shows the rank correlation values between two sets of N-best lists. Both sets are produced by a bigram LM (bg). The entropy of N-best lists in the first set set is 0.05 nats or less. The N-best lists in the second set have an entropy greater than 0.05 nats. Both these sets are re-ranked with bigger and better models (see Table 4.1 for model definitions). We can see from Table 4.2 that the rank correlation values tend to be higher (indicating little re-rankings) when the entropy of the N-best list, under the baseline model, is lower. Similarly, the rank-correlation values tend to be lower (indicating more re-rankings) whenever the entropy of the N-best list is higher. Note that these entropy values are computed with respect to the starting model (in this case, bigram LM). Of course, if the starting LM is much weaker than the rescoring model, then the entropy values need not be reflective of the difficulty of the overall task. This observation then suggests that it is safe to re-score only those N-best lists whose entropy under the starting model is higher than some threshold.

Rescoring Model	$\rho_{(H \leq 0.05)}$	$\rho_{(H > 0.05)}$
bg	1.00	1.00
tg	0.58	0.38
n-gram+syntactic	0.54	0.31

Table 4.2: Spearman Rank Correlation on the N-best list extracted from a bi-gram language model (*bg*) and re-scored with relatively better language models (see Table 4.1 for model definitions). Entropy under the baseline model correlates well with the rank correlation factor, suggesting that exhaustive search need not be necessary for utterances yielding lower entropy.

$$
\begin{aligned}
H(\mathcal{L}) &= \sum_{\substack{\pi\in\mathcal{E}^* \\ s.t. p[\pi]=n_s[\mathcal{L}] \\ n[\pi]\in N_e[\mathcal{L}]}} \frac{A[\pi]^\gamma L[\pi]}{Z} \log \frac{Z}{A[\pi]^\gamma L[\pi]} \\
&= \log Z \\
&\quad -\frac{1}{Z} \sum_{\substack{\pi\in\mathcal{E}^* \\ s.t. p[\pi]=n_s[\mathcal{L}] \\ n[\pi]\in N_e[\mathcal{L}]}} (A[\pi]^\gamma L[\pi]) \log(A[\pi]^\gamma L[\pi]) \\
&= \log Z \\
&\quad -\frac{1}{Z} \sum_{\substack{\pi\in\mathcal{E}^* \\ s.t. p[\pi]=n_s[\mathcal{L}] \\ n[\pi]\in N_e[\mathcal{L}]}} (P[\pi]) \log(P[\pi])
\end{aligned}
\tag{4.8}
$$

where $P[\pi] = A[\pi]^\gamma L[\pi]$.

While computation of entropy for N-best list is tractable, for a word lattice, the computation of entropy is intractable if one were to enumerate all the hypotheses. Even if we were able to enumerate all hypotheses, this method tends to be slower. Using efficient semiring techniques introduced by Li. et.al. [87] or using posterior probabilities (computed using the aid of Equation 4.1.2) on the edges leading to end states, we can compute the entropy of a lattice in one single forward pass using dynamic programming. It should, however, be noted that, for dynamic programming technique to work, only finite ordered n-gram LMs can be used. One has to resort to approximate entropy computation via N-best list if entropy under long span LM or models requiring complex dependencies with the distant context is desired. We illustrate below the process for computing entropy on word lattice using dynamic programming:

For any path $\pi \in \mathcal{E}^*$ s.t. $p[\pi] = n_s$ and $n[\pi] \in N_e$, the joint likelihood under acoustic model and language model, i.e. $P[\pi]$, can be decomposed into series of joint likelihoods over unit length paths. Thus, $P[\pi]$ can be obtained as:

$$
\begin{aligned}
P[\pi] &= \prod_{j=1}^{|\pi|} \Big(P(\mathbf{a}_j | w[\pi_j])^\gamma \\
&\qquad \times P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}]) \Big)
\end{aligned}
\tag{4.9}
$$

where $\forall j \in \{1, 2, \ldots, |\pi|\}$, $\pi_j \in \mathcal{E}$, $\pi = \odot_{j=1}^{|\pi|} \pi_j$, $\mathbf{a_j}$ is the acoustic string spanning between $T_s[\pi_j]$ and $T_e[\pi_j]$. Note that here it is assumed that the word lattice encodes m-gram

information thus the language model probabilities are conditioned on previous $m-1$ words alone.

In order for the dynamic programming approach to work, a variable is propagated from one node to another by traversing the word lattice in the topological order. At any path $\pi \in \mathcal{E}$ if $P[\pi](= A[\pi]^{\gamma} L[\pi])$ represents the joint acoustic and language model probability, then we maintain this variable at every such edge of the lattice. We maintain a similar variable $p(u)$ at every node, $u \in \mathcal{V}$. The value of the variable associated with any node is dependent upon the values on the neighboring nodes preceding it in time and also on the value of the variable on the arc connecting them.

For any node, $u \in \mathcal{V}$, if we define $\mathcal{N}(u) \subset \mathcal{V}$ as the set of nodes such that $\forall v \in \mathcal{N}(u)$, $T[v] < T[u]$ and $v \sim u$ i.e. v and u are connected by an edge, then using a recursion by iterating over states in topological order and initializing the value for the starting state as 1 (i.e. $p[n_s] = 1$), we compute the value of variables for any node, u, as follows:

$$p(u) = \sum_{\substack{\pi \in \mathcal{E} \\ p[\pi]=v, n[\pi]=u \\ v \in \mathcal{N}(u)}} p(v) \times P[\pi] \tag{4.10}$$

The entropy of the lattice is then given by:

$$\begin{aligned} H(\mathcal{L}) &= \log Z \\ &\quad -\frac{1}{Z} \sum_{u \in N_e} p(u) \log p(u) \end{aligned} \tag{4.11}$$

$$\tag{4.12}$$

where Z is simply obtained as:

$$Z = \sum_{u \in N_e} p(u) \tag{4.13}$$

4.2.2.2 Speed Up for Iterative Decoding on Word Lattices

Our speed up technique is simple. Once we have formed self contained sub lattices, we want to prune all but the top few best complete paths (obtained under baseline / starting model) of those sub lattices whose entropy is below some threshold. Thus, believing in the original model's confidence, we want to focus only on those sub lattices which the recognizer found difficult to decode in the first pass. All other part of the parent lattice will be not be analyzed. The thresholds for pruning is very application and corpus specific and needs to be tuned on some held out data.

4.2.3 Iterative Decoding on Confusion Networks

Iterative decoding on confusion networks works similar to that on word lattices except that the notion of local neighborhoods is now defined by the confusion bin.

The inputs to the algorithm is the new re-scoring LM and confusion network obtained using baseline acoustic and language model (for tasks such as system combination for speech recognition and machine translation, the CNs can be obtained by aligning 1-best (or N-bests) outputs from multiple systems). The re-scoring LM now need not be limited in size and order or even restricted to finite state machine family. We will continue to follow the notations introduced in Sections 4.1.1 and 4.1.2. Algorithm 5 below illustrates the steps required for re-scoring. The output of the algorithm is a word string, $\mathbf{W}^*$, such that it is the concatenation of final decoded word strings from each confusion bin. Thus if we denote the final decoded word (under some decoding scheme, which will become apparent next) in the j^{th} confusion bin by w_j^*, then $\mathbf{W}^* = w_1^* \cdot w_2^* \cdot \ldots \cdot w_K^* \equiv \odot_{j=1}^{C} w_j^*$.

Algorithm 5 Iterative Decoding on Confusion Networks

Require: CN $= \{B_1, B_2, \ldots, B_K\}$, L_{new}
  PrevHyp $\leftarrow$ null
  $\forall i \in \{1, 2, \ldots, K\}\ \hat{w}_i = w_{i,1}$
  CurrentHyp $\leftarrow \odot_{j=1}^{K} w_{j,1}$
  while PrevHyp $\neq$ CurrentHyp **do**
    for $i \leftarrow 1 \ldots K$ **do**
      $\hat{w}_i \leftarrow \underset{w_i \in B_i}{\text{argmax}} \Big(S[\hat{w}_1 \cdot \ldots \cdot \hat{w}_{i-1} \cdot w_i \cdot \hat{w}_{i+1} \ldots \cdot \hat{w_K}] \Big)$
    end for
    PrevHyp $\leftarrow$ CurrentHyp
    CurrentHyp $\leftarrow \odot_{j=1}^{K} \hat{w}_j$
  end while
  $\forall j \in \{1, 2, \ldots, K\}\ w_j^* = \hat{w}_i$

The algorithm is initialized by setting **PrevHypo** to null and **CurrHypo** to the concatenation of 1-best output from each confusion bin. Thus if we denote the highest posterior probability word under baseline models in some j^{th} confusion bin by $\hat{w}_j = w_{j,1}$, **CurrHypo** is then initialized to: $\hat{w}_1 \cdot \hat{w}_2 \cdot \ldots \cdot \hat{w}_K$. The algorithm then runs as long as CurrHypo is not equal to PrevHypo. In each iteration, the algorithm sequentially re-scores each confusion bin by keeping the surrounding context fixed. Once all the bins are re-scored, that constitutes one iteration. At the end of each iteration, CurrHypo is set to the concatenation of 1-best paths from each confusion bin while PrevHypo is set to the old value of CurrHypo. Thus if we are analyzing, say, some i^{th} bin in some iteration, then 1-best paths from all but this bin is kept fixed and a *new* 1-*best* path under the re-scoring LM is found out as shown below.

$$\hat{w}_i \leftarrow \underset{w_i \in B_i}{\operatorname{argmax}} \Big(S_{new}[\hat{w}_1 \cdot \ldots \cdot \hat{w}_{i-1} \cdot w_i \cdot \hat{w}_{i+1} \ldots \cdot \hat{w_K}] \Big)$$

where the left context: $\hat{w}_1 \cdot \ldots \cdot \hat{w}_{i-1}$ and right context $\hat{w}_{i+1} \ldots \cdot \hat{w_K}$ is fixed during this local rescoring. $S[\cdot]$ is the scoring function which assign appropriate scores to the hypotheses depending on task and on knowledge sources that need to be combined. For speech recognition based confusion network re-scoring, hypotheses can be force aligned with acoustic vector to get appropriate acoustic score for later combining it with new language model scores OR posterior scores could be combined with new language model scores. For machine translation applications such as system combination, consensus voting scores could be combined with new language model score. Hence we will use a generic scoring function, $S[\cdot]$ to illustrate the working of the algorithm.

4.2.3.1 Simulated Annealing

CNs are not always a favorable representation for re-scoring and should be used in places where representations such as word lattices or hypergraphs are not available. Presence of peaky posterior scores in CNs bias the search towards the hypothesis favorable towards the weaker starting model [88]. In CNs obtained from speech word lattices, many a times the word boundaries are not respected due to the ad-hoc grouping of competing word hypotheses into a single confusion bin. If a long word, such as, say `motorcycle`, is competing with `motor` and `cycle`, then chances are that the a CN will group the word `motorcycle` with either `motor` or `cycle` in one bin and a null transition and the remaining word in the adjoining bin. Similar effect can be observed in system combination based CNs where 1-bests are aligned together. It is then likely that the sub word sequence `motorcycle motor` (or `motorcycle cycle`) will be part of a dominating hypothesis. Greedy hill climbing based technique such as Iterative Decoding then gets stuck at local optimum solution. In order to avoid the algorithm getting stuck at local optimum solution, either a bigger neighborhood needs to be explored or simulated annealing [89] based methods need to employed. Algorithm 6 below demonstrates the steps required for extending Iterative Decoding (Alg. 5) for CNs with simulated annealing.

A metavariable called 'Temperature' is chosen to have some high value which is then decremented after every local convergence. The algorithm runs as long as this variable is not zero.

In this version of the algorithm, at any step, the word hypothesis is *not* chosen to be the mode of the distribution characterized by the scoring function $S[\cdot]$.[12] Instead, a word hypothesis is *sampled* from this distribution as shown below:

[12] If we normalize the scoring function over the words of the confusion bin under consideration, then an argmax over this function is equivalent to finding the mode of the distribution obtained after normalizing $S[\cdot]$.

Algorithm 6 Simulated Annealing based Iterative Decoding on Confusion Networks

Require: CN $= \{B_1, B_2, \ldots, B_K\}$, L_{new}
  PrevHyp $\leftarrow$ null
  $\forall i \in \{1, 2, \ldots, K\}\ \hat{w}_i = w_{i,1}$
  CurrentHyp $\leftarrow \odot_{j=1}^{K} w_{j,1}$
  Temp = θ
  while Temp $\neq 0$ **do**
    while PrevHyp $\neq$ CurrentHyp **do**
      for $i \leftarrow 1 \ldots K$ **do**
        $\hat{w}_i \leftarrow \underset{\mathbf{w}_i \in B_i}{\text{sample}}\Big(S[\hat{w}_1 \cdot \ldots \cdot \hat{w}_{i-1} \cdot \mathbf{w}_i \cdot \hat{w}_{i+1} \cdot \ldots \cdot \hat{w_K}]\Big)$
      end for
      if $S(\odot_{j=1}^{K} \hat{w}_j) > S$(PrevHyp) **then**
        PrevHyp $\leftarrow$ CurrentHyp
        CurrentHyp $\leftarrow \odot_{j=1}^{K} \hat{w}_j$
      else
        if `rand` $\geq f\Big(S(\odot_{j=1}^{K} \hat{w}_j)$,S(PrevHyp), Temp$\Big)$ **then**
          PrevHyp $\leftarrow$ CurrentHyp
          CurrentHyp $\leftarrow \odot_{j=1}^{K} \hat{w}_j$
        end if
      end if
    end while
    Temp $\leftarrow$ Temp $-\Delta$
  end while
  $\forall j \in \{1, 2, \ldots, K\}\ w_j^* = \hat{w}_i$

$$\hat{w}_i \leftarrow \underset{\mathbf{w}_i \in B_i}{\text{sample}}\Big(S[\hat{w_1} \cdot \ldots \cdot \hat{w}_{i-1} \cdot \mathbf{w}_i \cdot \hat{w}_{i+1} \ldots \cdot \hat{w_K}]\Big)$$

If the resulting sentence hypothesis has a greater score than the previous sentence hypothesis (stored in PrevHyp), then the solution is accepted else, it is rejected but only with a certain probability. This probability is a function of the metavariable – Temperature, and the difference in scores of the current solution and solution stored in PrevHyp. The probability of rejection is higher if the current solution has very low score in comparison to the previous solution. A decrease in the metavariable value forces the algorithm to reject bad solutions with higher probability. A higher temperature (which is set towards the initial phase of the algorithm) favors some bad solutions. This causes algorithm to deviate from its path of climbing a local optimum hill and hopefully lead towards a better solution.

Simulated annealing based extensions are also possible for Iterative Decoding on word lattices, however, our experiments so far show that there is no need for any such heuristic. The formation of the islands of confusabilities in the word lattice define the neighborhoods of varying sizes and this, we believe, turns out to be extremely helpful in getting optimal solutions. Thus as far as speech recognition experiments are concerned, we decided to work with word lattices. Only for cases where no other favorable representation was available (such as system combination for machine translation task), did we chose to work with confusion networks.

4.3 Experiments and Results

We will discuss two sets of experiments. In the next sub-section we will describe the experiment in which we re-score confusion networks for a system combination task of machine translation using very big n-gram language model. In the following sub-section, we will describe the experiment in which we re-score word lattices with long span language models.

4.3.1 Re-scoring of Confusion Networks

We will discuss the performance of the proposed method for re-scoring confusion networks obtained by aligning 1-best output of machine translation systems. Since the re-scoring model is going to be an extremely large n-gram LM, we will show the efficacy of the proposed method while comparing to computationally complex finite state machine composition re-scoring method. We have worked on Karakos et.al.'s experimental setup [57] details of which are summarized below.

The main steps performed for combining two or more translations for a source segment (the segment of text which needs to be translated in another language, mostly english) are as follows[13]:

[13]These steps are taken as is from Karakos et.al.'s paper [57], since we do not do any of these steps or

1. The input translations[14] are first lowercased and tokenized.

2. *Within System Combination:* If N-best lists are available, they are first combined together on a per-system basis using regular edit distance resulting in one confusion network per system.

3. *Between-systems combination:* The per-system confusion networks are aligned together using Inversion-Transduction Grammars [90], one at a time, using a fixed decreasing BLEU-TER[15] sequence of systems. The function for computing the alignment cost of two confusion network bins is:

$$C(B_1, B_2) = \frac{1}{|b_1||b_2|} \sum_{u \in B_1, v \in B_2} c_1(u) c_2(v) \mathbf{1}(u \neq v),$$

   where $|b| = \sum_{w \in B} c(w)$ is the total count of the tokens in B (in other words, it is the number of system outputs from the n-best lists which happened to have their words aligned together in bin B.) Note that the total number of types in bin is denoted by $|B|$.

4. Each arc in the confusion network receives a cost equal to the negative log-probability of the word in its assigned bin, that is $\text{cost}(w) = -\log\Big(c(w)/|b|\Big)$.

5. The final confusion networks get rescored with a language model mostly of n-gram family. An additive 'word deletion penalty' is also used to modify the arc costs by a constant amount, making longer paths less costly.

6. The output of the combination is the minimum-cost path in the rescored lattice.

For step 5, FSM of CN is composed with the FSM of LM.[16] A least cost path is then found out through the composed machine. However, for big language models (with orders greater than 5 containing hundreds of millions of n-grams) such as the one used in our experiment, it is computationally impossible to carry out lazy composition. Karakos et.al. hence filters out all the non-relevant n-grams. This can be done by extracting all the n-grams possible in the confusion network and then keeping only these n-grams in the LM. Note that, this is a very costly process as it involves *(a):* expanding the CN so that at any

parameter tuning in our experiments and we work on the confusion network built by them. There are many tunable parameters in these steps such as: number of systems participating in the combination, the scaling factor of the language model and the insertion penalty. For our experiments, we have worked with the parameters tuned by Karakos et.al.

[14] Output of individual machine translation systems are inputs to the system combination module.

[15] See [67] for a description on how the metric BLEU is defined for measuring the accuracy of a machine translation output. TER is the translation error rate, which is similar to word error rate except that apart from having insertions, deletions and substitutions as valid edits, it also uses phrase reordering as a valid set of edits.

[16] See [83] for details on how an n-gram LM is converted into an weighted finite state automata.

node, an unambiguous context of $n-1$ words is maintained and *(b)*, going through the big language model to weed out all the non essential n-grams. This process has to be done for every confusion network and hence renders the entire decoding process very slow. We will refer to this re-scoring technique by 'FSM Re-scoring'.

We propose to use Iterative Decoding to do the rescoring of the confusion bin with large re-scoring n-gram language model. We will refer to this technique by 'Iterative Decoding'. We will use this decoding technique with with either single bin local neighborhood (LN=1), two bin local neighborhood (LN=2)[17] or simulated annealing.

Experiments on Arabic-to-English translation task is presented on three corpora: Broadcast Conversation, Broadcast News and News Wire. Our test set consisted of 1-best translations from 22 systems (made available by NIST). The re-scoring language model used in our experiments was a Modified-Kneser-Ney smoothed 5-gram model, trained on the English Gigaword and part of the English side of the GALE parallel development data. The LM contained nearly 170M n-grams. Re-scoring is done by assigning new costs to the words in confusion bins by linearly interpolating the costs computed in step 4 above and the negative log probability of the language model score assigned to the word given the context of the previous $n-1$ words.

From Table 4.3, we can see that in all the three experiments, Iterative Decoding based re-scoring method improves upon the baseline significantly. With a bigger neighborhood, the improvement is sometimes more even reaching optimal performance. Simulated Annealing based extensions do not perform any better than the greedy hill climbing techniques suggesting that simple hill climbing with a bigger neighborhood suffices and works very well practically.

The purpose of the experiment was to show that using a simple linear search function, a considerably better hypothesis can be obtained and the need for computationally heavy re-scoring methods can be avoided.

4.3.2 Re-scoring of Word Lattices

In this experiment we have carried out re-scoring of word graphs – lattices with a long span model - recurrent neural network language model (RNN-LM) and combination of many of these and n-gram LMs. We performed recognition on the Broadcast News (BN) `dev04f, rt03` and `rt04` task using state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio) provided to us by IBM [91]. IBM also provided us its state-of-the-art speech recognizer, Attila [92] and two Kneser-Ney smoothed backoff n-gram LMs containing 4.7M n-grams ($n \leq 4$) and 54M n-grams ($n \leq 4$) trained on BN text (400M word tokens). We will refer to them as KN:BN-Small and KN:BN-Big respectively. The LM training text consists of 400M words from the following data sources: 1996 CSR Hub4 Language Model data, EARS BN03 closed cap-

[17] In this version, local neighborhoods are formed by grouping adjacent bins. Thus if bins B_1 and B_2 are grouped, then the number of hypotheses formed equals the product of their sizes i.e. $|B_1| \times |B_2|$.

(a) Broadcast Conversation

Setup	BLEU
Baseline	29.16
Iterative Decoding (LN=1)	30.17
Simulated Annealing	30.22
Iterative Decoding (LN=2)	30.41
FSM re-scoring	30.38

(b) Broadcast News

Setup	BLEU
Baseline	36.11
Iterative Decoding (LN=1)	36.78
Simulated Annealing	36.78
Iterative Decoding (LN=2)	36.80
FSM re-scoring	37.01

(c) News Wire

Setup	BLEU
Baseline	58.22
Iterative Decoding (LN=1)	58.37
Simulated Annealing	58.39
Iterative Decoding (LN=2)	58.39
FSM re-scoring	59.29

Table 4.3: From all the three experiments it can be seen that Iterative Decoding based re-scoring method improves upon the baseline significantly. With a bigger neighborhood, the improvement is sometimes more even reaching optimal performance. Simulated Annealing based extensions do not perform significantly better than the greedy hill climbing techniques.

tions, GALE Phase 2 Distillation GNG Evaluation Supplemental Multilingual data, Hub4 acoustic model training transcripts, TDT4 closed captions, TDT4 newswire, and GALE Broadcast Conversations and GALE Broadcast News.

We trained two RNN based language models - the first one, denoted further as RNN-limited, was trained on a subset of the training data (58M tokens). It used 400 neurons in the hidden layer. The second model, denoted as RNN-all, was trained on all of the training data (400M tokens), but due to the computational complexity issues, we had to restrict its hidden layer size to 320 neurons.

We followed IBM's multi-pass decoding recipe [92] using KN:BN-Small in the first pass followed by either N-best list re-scoring or word lattice re-scoring using bigger and better models. For the purpose of re-scoring, we combined all the relevant statistical models in one unified log linear framework reminiscent of Beyerlein's work [61]. We, however, trained the model weights by optimizing expected WER rather than 1-*best* loss (see Chapter 3 for more details).We will refer to the log linear combination of KN:BN-Big and RNN-limited by KN-RNN-lim; KN:BN-Big and RNN-all by KN-RNN-all and KN:BN-Big, RNN-limited and RNN-all by KN-RNN-lim-all.

We used two sets for decoding: `rt03+dev04f` set was used as a development set while `rt04` was used as a blind set for the purpose of evaluating the performance of long span RNN models using the proposed approach. We will refer to the development set as Dev and evaluation set as Eval. We made use of OpenFst C++ libraries [93] for manipu-

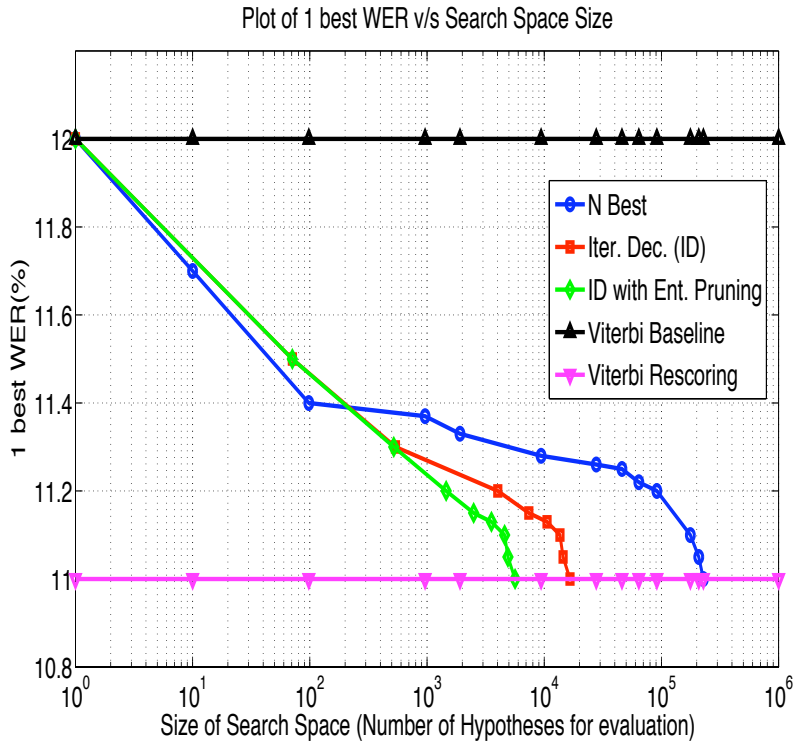

Figure 4.1: Plot of WER (y axis) on `rt03+dev04f` set versus the size of the search space (x axis). The baseline WER obtained using KN:BN-Small is 12% which then drops to 11% when KN:BN-Big is used for re-scoring. N-best list search method obtains the same reduction in WER by evaluating as many as 228K sentence hypotheses on an average. The proposed method obtains the same reduction by evaluating 14 times smaller search space. The search effort reduces further to 40 times if entropy based pruning is employed during re-scoring.

lating lattice graphs and generating N-best lists. Due to the presence of hesitation tokens in reference transcripts and the need to access the silence/pause tokens for penalizing short sentences, we treated these tokens as regular words before extracting sentence hypotheses. This, and poorly segmented nature of the corpus (`rt03+dev04f`), led to huge enumeration of sentence hypotheses.

4.3.2.1 n-gram LM for re-scoring

In this setup, we used KN:BN-Small as the baseline starting LM which yielded the WER of 12% on Dev set. Using KN:BN-Big as the re-scoring LM, the WER droped to 11%. Since the re-scoring LM belonged to the n-gram family, it was possible to compute the optimal word string by re-scoring the whole lattice (see Sec. 4.1.1). We now compare the performance of N-best list approach (Sec. 4.1.3) with our proposed approach (Sec.

4.2). N-best list achieved the best possible reduction by evaluating as many as $228k$ sentence hypotheses on an average. As against that, our proposed approach achieved the same performance by evaluating $16.6k$ sentence hypotheses, thus reducing the search efforts by $\mathbf{13.75}$ times. If we carry out entropy pruning (see Sec. 4.2.2.1) on sub lattices, then our proposed approach required as little as $5.6k$ sentence hypotheses evaluations to obtain the same optimal performance, reducing the search effort by as much as $\mathbf{40.46}$ times. For the purpose of this experiment, entropy based pruning was carried out when the entropy of the sub lattice was below 5 nats. Table 4.4 compares the two search methods for this setup. Fig. 4.1 shows a plot of Word Error Rate (WER) on Dev set versus the size of the search space (in terms of number of sentence hypotheses evaluated by an n-gram language model).

On the Eval set, the KN:BN-Small LM gave a WER of 14.1% which then dropped to 13.1% after re-scoring with KN:BN-Big. Since the re-scoring model was an n-gram LM, it was possible to obtain the optimal performance via lattice update technique (see Sec. 4.1.1). We then carried out the re-scoring of the word lattices under KN:BN-Big using our proposed technique and found it to give the same performance yielding the WER of 13.1%.

4.3.2.2 Long Span LM for re-scoring

In this setup, we used the strongest n-gram LM as our baseline. We thus used KN:BN-Big as the baseline LM which yielded the WER of 11% on Dev. We then used KN-RNN-lim-all for re-scoring. Due to long span nature of the re-scoring LM, it was not possible to obtain the optimal WER performance. Hence we have compared the performance of our proposed method with N-best list approach. N-best list achieved the lowest possible WER after evaluating as many as $33.8k$ sentence hypotheses on an average. As against that, our proposed approach in conjunction with entropy pruning obtained the same performance by evaluating just $1.6k$ sentence hypotheses, thus reducing the search by a factor of $\mathbf{21}$. Inspite of starting off with a very strong n-gram LM (as against starting off with a weaker n-gram LM as described in Sec. 4.3.2.1), the N-best lists so extracted were still not representative enough of the long span rescoring models. Had we started off with KN:BN-Small, the N-best list re-scoring method would have had no chance of finding the optimal hypothesis in reasonable size of hypotheses search space. Table 4.5 compares the two search methods for this setup when many other long span LMs were also used for re-scoring. Fig 4.2 is a plot of Word Error Rate (WER) on Dev set versus the size of the search space (in terms of number of sentence hypotheses evaluated by a long span language model).

On the Eval set, the KN:BN-Big LM gave a WER of 13.1% which then dropped to $\mathbf{12.15}\%$ after re-scoring with KN-RNN-lim-all using our proposed technique.[18] Since the re-scoring model was not an n-gram LM, it was not possible to obtain an estimate of the optimal performance but we could enumerate huge N-best list to approximate this value. The proposed method is much faster than huge N-best lists and no worse in terms of WER.

[18]The WER obtained using KN-RNN-lim and KN-RNN-all was 12.5% and 12.3% respectively.

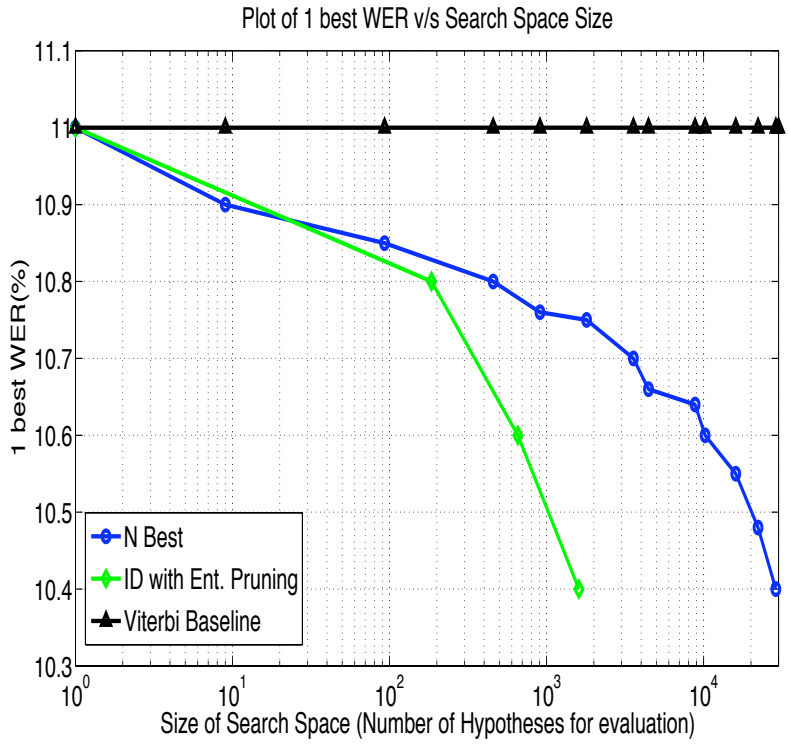

Figure 4.2: Plot of WER (y axis) on `rt03+dev04f` set versus the size of the search space (x axis). The baseline WER obtained using KN:BN-Big is 11% which then drops to 10.4% when KN-RNN-lim-all is used for re-scoring. N-best list search method obtains this reduction in WER by evaluating as many as 33.8K sentence hypotheses on an average, while the proposed method (with entropy pruning) obtains the same reduction by evaluating 21 times smaller search space.

As far as we know, the result obtained on both Dev and Eval set is the best performance ever reported on the Broadcast News corpus for speech recognition. For Eval set, we did not know how much N-best list size was appropriate for getting a near optimal performance and hence we enumerated about 10K hypotheses (since the sentence lengths on an average were relatively smaller for `rt04` set than `rt03+dev04f` set and hence 10K number of hypotheses seemed appropriate).

4.4 Conclusion

In this chapter we proposed and demonstrated a re-scoring technique for general word graph structures such as word lattices and confusion networks. We showed its efficacy by demonstrating huge reductions in the search effort to obtain a new state-of-the-art performance on a very competitive speech task of Broadcast news. We also demonstrated that

Models	**WER**	N**-Best**	**ID**	**Saving**
KN:BN-Small	12.0	-	-	-
KN:BN-Big	11.0	228K	5.6K	40

Table 4.4: Performance of various re-scoring strategies with many LMs on `rt03+dev04f` corpus of BN. Setup where the starting LM is a weak n-gram LM (KN:BN-Small) and the re-scoring LM is a much stronger but n-gram LM (KN:BN-Big). The baseline WER in this case is 12% and the optimal performance by the re-scoring LM is 11.0%. The proposed method outperforms N-best list approach, in terms of search efforts, obtaining optimal WER.

Models	**WER**	N**-Best**	**ID**	**Saving**
KN:BN-Big	11.0	-	-	-
KN-RNN-lim	10.5	42K	1.1K	38
KN-RNN-all	10.5	26K	1.3K	20
KN-RNN-lim-all	10.4	34K	1.6K	21

Table 4.5: Performance of various re-scoring strategies with many LMs on `rt03+dev04f` corpus of BN. Setup where the starting LM is a strong n-gram LM (KN:BN-Big) and the re-scoring model is a long span LM (KN-RNN-*). The baseline WER is 11.0%. Due to long span nature of the LM, optimal WER could not be estimated. The proposed method outperfoms N-best list approach on every re-scoring task.

more computationally complex re-scoring approaches such as FSM re-scoring could be replaced with our proposed approach which is linear in length of the confusion network and reduces the re-scoring time and efforts considerably.

Chapter 5

Variational Inference

In this chapter, we propose techniques to incorporate information from long-span LMs directly into first pass decoding. Unlike the methods presented in Chapter 4, where the search space was approximated by forming islands of confusability and a full blown long span model was deployed for decoding, the method presented in this chapter will instead approximate the model so that its tractable form could be used to search the entire search space without needing to approximate it.

5.1 Integrating Language Models into LVCSR

We propose to approximate the long-span model using **variational inference** techniques. Given a long-span model P, possibly a sophisticated LM with complex statistical dependencies, we will seek a simple and computationally tractable model Q^* that will be a good surrogate for P. Specifically, among all models Q of a chosen family $\mathcal{Q}$ of tractable models, we will seek the one that minimizes the Kullback-Leibler divergence [94, pp. 20] from P. We will use this model for first pass decoding, which in turn produces richer lattices and faithful N-best lists. We then deploy the full blown model on the N-best lists extracted from the first pass recognition. Thus the N-best lists that are extracted from the first pass decoding are no more biased towards the weaker baseline model.[1] They are infact biased towards the model with which we wish to do the re-scoring. This technique thus helps to produce a refined and less biased search space for second pass decoding such that re-scoring on it becomes much more effective. Examples of P include computationally powerful LMs outside the family of finite state machines, such as recurrent neural networks [41], random forests [95] and structured language models [33; 76]. We will approximate P with a Q^* from the family $\mathcal{Q}$ of finite state machines. The choice of $\mathcal{Q}$ is driven by decoding capabilities.

The rest of the chapter is organized as follows. Section 5.2 gives a background on the general topic of approximate inference in machine learning. Section 5.3 provides details

[1] Here we are referring to a weak n-gram LM which is used in the decoder to produce word lattices.

about the proposed methodology for language modeling using ideas from variational inference. Section 5.4 briefly describes our long-span LM, a recurrent neural network. Section 5.5 describes our experimental setup and presents a number of results. Finally, a summary of the chapter with some remarks is presented in Section 5.6.

5.2 Approximate Inference

A crucial issue to many of the machine learning problems is estimating the statistics under the posterior distribution $P(\mathbf{Z}|X)$ ($\propto P(X|\mathbf{Z})P(\mathbf{Z})$), where $\mathbf{Z}$ is the latent (hidden) variable and X is the observed variable. Useful statistics under this posterior distribution could be expectation of the joint likelihood of latent variable and observed variable. In Appendix A, we derive the formulation of Expectation Maximization in order to find out optimal language model interpolation weights. The latent variable there denotes the hidden assignment of language models to a specific observed variable (observed variable is word of a language, in this specific case). It can be seen from the derivation how crucial computation of the posterior distribution, $P(\mathbf{Z}|X)$, and the computation of expected joint likelihood under this distribution is to the optimality of the EM algorithm. EM algorithm in turn guarantees an increase in the likelihood of the observed data. Although in the simple case of language model interpolation, it was easy to find the posterior distribution in closed form, there are potentially many cases, where an analytical form of the posterior distribution is not possible. Moreover, computation of sufficient statistics under this distribution, which is crucial for algorithms such as EM, may be computationally intractable. This may be due to higher dimensionality of data or due to complex form of probability distribution of observed variable given the latent variables. Approximate inference techniques are thus required to approximate the complicated posterior distribution with an easy form so that computing statistics under this surrogate distribution becomes tractable.

Variational methods were first introduced in the *calculus of variations* literature. While standard calculus is concerned with finding the derivatives of the functions with respect to the input variable, calculus of variations is concerned with finding the derivatives of the *functionals* with respect to the input functions. A functional can be thought of as a function of some function. For example, entropy, $H[p(\cdot)]$ is a functional of the probability distribution, $p(\mathbf{x})$, which in turn is a function of the random variable, $\mathbf{x}$

$$H[p] = \int_x p(x) \log \frac{1}{p(x)} dx.$$

There are many problems in machine learning which requires us to solve some optimization problem in which the quantity being optimized is a functional. For instance, unconstrained maximum entropy problem requires to find out the function p^* such that the functional $H[p^*]$ is maximized. The constrained maximization problem could be posed similarly. These problems are very crucial and important in many machine learning areas including language modeling, where the function $p(\cdot)$ characterizes the distribution of

words given some context while $H[p]$ denotes the entropy of such a language model. Maximum entropy language models (ME-LM) are obtained by maximizing the functional – entropy, over all possible functions – probability distributions, while meeting certain constraints. While these optimization problems can be solved efficiently, many other problems require approximations. Some of these approximations could mean restricting the range of functions over which the optimization is performed or by invoking conditional independence assumptions for factorizing complex distribution into series of simple distributions. Variational methods were introduced in the context of approximating some of these intractable optimization problem.

The problem we are interested in is that of finding the most likely hypothesis given a model of language and acoustics. We thus seek to find the word string, W^*, such that given some representation of the acoustics, $\mathbf{A}$, it becomes the most likely. Mathematically it can be stated as:

$$W^* = \underset{W \in \mathcal{W}}{\operatorname{argmax}}\, P(\mathbf{A}|W)P(W) \tag{5.1}$$

However, if the model of languages i.e. $P(W)$ is complex then the search (concerning with argmax of above equation) becomes computationally challenging. Typically $P(W)$ is then assumed to be factorizable into series of simple distributions thus facilitating a tractable search for W^* which can be thought of as concatenation of solutions of series of simple problems. We will refer to the product of simple distributions by $R(W)$ and denote its family by $\mathcal{R}$. Ideally the choice of the surrogate model should be made such that it is closest to the complex model in terms of some appropriate distance metric. However, this simpler model, $R(W)$, and the given complex model, $P(W)$, are often related to each through only through the training data from which they are *independently* estimated. It is thus not guaranteed that in the family of distributions $\mathcal{R}$, $R(W)$ will be closest to $P(W)$. We thus have to explicitly solve an optimization problem minimizing a functional – Kullback Leibler divergence [94, pp. 20] between given and desired model, over the functions possible in a certain distribution family of choice. Since the functions involved are probability distribution, it is only natural to choose Kullback-Leibler divergence as the distance metric. The next section explores this idea in detail.

5.3 Variational Approximation of a Model

There are many popular methods of approximate inference, among which variational inference has gained popularity due to its simplicity [96, pp. 462]. Such methods are necessary when exact inference is intractable. In variational inference, a surrogate model characterized by the distribution $Q \in \mathcal{Q}$ is chosen to replace a complex model, characterized by the distribution P, such that inference under Q becomes much more tractable. Q should be found out such that it is closest to P in some sense. Since these models are probability distributions, a very natural choice of distance metric is Kullback-Leibler diver-

gence (KLD). The surrogate[2] model Q is thus chosen such that among all the distributions in the family of the chosen parameterization, $\mathcal{Q}$, it has the minimum KLD with the complex distribution P. Thus if we decide on a family of distributions, $\mathcal{Q}$, then the surrogate distribution is found out by solving the following optimization problem:

$$\begin{aligned} Q^* &= \operatorname*{argmin}_{Q \in \mathcal{Q}} D(\mathbf{P} \| \mathbf{Q}) \\ &= \operatorname*{argmin}_{Q \in \mathcal{Q}} \sum_{\cdot} P(\cdot) \ln \frac{P(\cdot)}{Q(\cdot)} \\ &= \operatorname*{argmax}_{Q \in \mathcal{Q}} \sum_{\cdot} P(\cdot) \ln Q(\cdot) \end{aligned} \tag{5.2}$$

where $D(\mathbf{p}||\mathbf{q}) = \sum_{x \in \mathcal{X}} p(x) \ln \frac{p(x)}{q(x)}$ is the KL Divergence between the probability mass functions p and q.
In our work, we choose $\mathcal{Q}$ to be the family of distributions parameterized by n-grams[3] i.e. we want to learn a model, Q, parameterized by n-grams, that is closest to P in the sense of Kullback Leibler divergence. Under some mild conditions, Q^* is the n-dimensional marginal of P.

A natural question is whether Q^* is simply the n-gram model $\hat{Q}$ estimated from the same (LM training) text that P was estimated from. Not surprisingly, the answer is negative. For one, even if both P and $\hat{Q}$ were estimated from the same text, P may have been estimated with a different criterion than maximum likelihood (ML), so that its n-gram marginals may not agree with $\hat{Q}$, even after differences due to smoothing are ignored. Even if P is the ML estimate from a rich family $\mathcal{P}$ that contain $\mathcal{Q}$ as a subset, additional assumptions must hold about $\mathcal{P}$ for Q^* to be the same as $\hat{Q}$.

But if P is indeed a long-span LM, then computing its n-dimensional marginal could also be computationally prohibitive. Often, and for our choice of P in Section 5.4, it is impossible to do so as shown in the subsection 5.3.1. So how does one proceed ? For any P that is a generative model of text, the minimizer of (5.2) may be approximated via Gibbs sampling [97], [96, pp. 542] !. Details about this procedure follows in subsection 5.3.2.

5.3.1 Marginalization

We will now demonstrate first solution for transforming a conditional distribution of a random variable given two random variables, into a relatively more tractable conditional distribution. For the sake of discussion, let us say we wish to transform a trigram model into a bigram model so that the resulting transformation is easy to work with. Assuming that we are given a tri-gram model, then the bi-gram distribution can be obtained by following marginalization procedure:

[2] We use P and Q interchangingly for the model or the distribution.

[3] All the distributions in this family have the same order i.e. choice of n and this is dependent on the decoding capabilities.

$$\begin{aligned} Q(x|y) &= \sum_{z \in \mathcal{V}} P(x, z|y) \\ &= \sum_{z \in \mathcal{V}} P(x|z, y)\hat{P}(z|y) \\ &= E_{\hat{P}(Z|y)}[P(x|Z, y)] \end{aligned} \tag{5.3}$$

Note that Eqn. 5.3 introduces a dependency on the bigram distribution. Ideally one should use the true marginalized bi-gram distribution here, however, this may lead to issues such as dependency on distributions which are yet to be determined etc. A simple solution can be to use the maximum likelihood estimates instead. We will denote this distribution by the letter $\hat{P}$. In all of our proofs and derivations, we will assume that the bi-gram distribution used for the process of marginalization is the maximum likelihood distribution.

We now have to prove that among all the bi-gram distributions, the marginalized distribution, $Q(X|y)$, is closest to the parent trigram distribution, $P(X|y, z)$, in the sense of Kullback-Leibler divergence. Since the trigram distribution involves an extra variable z, which potentially can take as many values as there are possible realizations of the corresponding random variable, a more valid metric would be the expected Kullback-Leibler divergence, where the expectation is taken with respect to the conditional distribution of z given y. Again, we will use the maximum likelihood bi-gram distribution, $\hat{P}$, to compute this expectation.[4]

Lemma 5.3.1 *If $Q(x)$ and $P(x)$ are two discrete probability distributions, then*

$$\sum_x Q(x) \ln Q(x) \geq \sum_x Q(x) \ln P(x)$$

with equality if and only if $Q(x) = P(x)$ for all x.

Proof Since $\forall x, \ln(x) \leq x - 1$, we get:

$$\sum_x Q(x) \ln \frac{P(x)}{Q(x)} \leq \sum_x Q(x) \Big(\frac{P(x)}{Q(x)} - 1 \Big) = 0$$

this implies that:

$$\sum_x Q(x) \ln P(x) \leq \sum_x Q(x) \ln Q(x) \qquad \blacksquare$$

Lemma 5.3.2 *If $Q(x)$ and $P(x)$ are two discrete probability distributions, then*

$$D(Q\|P) = \sum_x Q(x) \ln \frac{Q(x)}{P(x)} \geq 0$$

with equality if and only if $Q(x) = P(x)$ for all x.

[4] Damianos Karakos suggested the use of conditional relative entropy for deriving this theorem.

Proof Follows directly from Lemma 5.3.1. ∎

Theorem 5.3.3 *Under the maximum likelihood bigram distribution, $\hat{P}$, the expected Kullback-Leibler divergence between the trigram distribution, $P(X|y,Z)$, and its marginalized bi-gram distribution, $Q(X|y)$ (obtained using Eqn. 5.3), is less than or equal to the expected Kullback-Leibler divergence between this trigram distribution and any other bi-gram distribution, $R(X|y)$ i.e.*

$$\sum_z \hat{P}(z|y)D(P(X|y,z)\|Q(X|y)) \leq \sum_z \hat{P}(z|y)D(P(X|y,z)\|R(X|y))$$

Proof We have to prove:

$$\sum_z \hat{P}(z|y)D(P(X|y,z)\|Q(X|y)) \leq \sum_z \hat{P}(z|y)D(P(X|y,z)\|R(X|y))$$

Expanding the left hand side:

$$\begin{aligned}
\sum_z \hat{P}(z|y)D(P(X|y,z)\|Q(X|y)) &= \sum_z \hat{P}(z|y) \sum_x P(x|y,z) \ln \frac{P(x|y,z)}{Q(x|y)} \\
&= \sum_x \sum_z P(x|y,z)\hat{P}(z|y) \ln P(x|y,z) \\
&\quad - \sum_x \underbrace{\sum_z P(x|y,z)\hat{P}(z|y)}_{=Q(x|y) \text{ (See Eqn. 5.3)}} \ln Q(x|y) \\
&= \sum_x \sum_z P(x|y,z)\hat{P}(z|y) \ln P(x|y,z) \\
&\quad - \sum_x Q(x|y) \ln Q(x|y) \qquad (5.4)
\end{aligned}$$

Expanding the right hand side:

$$\begin{aligned}
\sum_z \hat{P}(z|y)D(P(X|y,z)\|R(X|y)) &= \sum_z \hat{P}(z|y) \sum_x P(x|y,z) \ln \frac{P(x|y,z)}{R(x|y)} \\
&= \sum_x \sum_z P(x|y,z)\hat{P}(z|y) \ln P(x|y,z) \\
&\quad - \sum_x \underbrace{\sum_z P(x|y,z)\hat{P}(z|y)}_{=Q(x|y) \text{ (See Eqn. 5.3)}} \ln R(x|y) \\
&= \sum_x \sum_z P(x|y,z)\hat{P}(z|y) \ln P(x|y,z) \\
&\quad - \sum_x Q(x|y) \ln R(x|y) \qquad (5.5)
\end{aligned}$$

Using Lemma 5.3.1 and noting that the first term of Eqn. 5.4 and Eqn. 5.5 is same, we get:

$$\sum_z \hat{P}(z|y) KL(P(X|y,z)||Q(X|y)) \leq \sum_z \hat{P}(z|y) KL(P(X|y,z)||R(X|y)). \quad \blacksquare$$

Although the marginalization procedure works practically well for tri-gram to bi-gram transformation, for any complex and sophisticated language model, which needs to use the entire context to predict the next word, a simpler model will need to be obtained by marginalizing out all the extra conditioning terms. This, however, will be an impossible task for such long span models mainly because the marginalization procedure will soon become computationally costly with rising order of language model.[5] Not only that, but the process of marginalization assumes that we already have long span probabilities of word given the context. For finite order markov models, these probabilities can still be obtained by looking at all conceivable n-gram patterns in the training data. Even if we assume that the same process can still be carried out for long span model yielding as many probability values as there are words in the training corpus, this procedure wont benefit from the fact that the long span models generalizes well on unseen data because all that we can capture would already be seen in the training data.

We hence look at this problem from a different perspective. We make use of **Gibbs Sampling** technique to infer Q.

At this point, we can motivate the basic idea behind Gibbs sampling for obtaining a tractable model from the long span language model. Let us say that in our training data we observe following two sentences:

- `I go to work on tuesday`
- `I go to work on wednesday`

and let us say that we also see the following sentence

- `today is tuesday`

but we don't see the sentence:

- `today is wednesday`

If such is our setup then the probability of the word `wednesday` given the context `today is` will not be robustly estimated (by the conventional n-gram LM) due to lack of observation in the training data. However, if some complex language model is able to *cluster* the words `tuesday` and `wednesday` based on the fact that they follow the

[5]Note that for language modeling, the above marginalization will give us rather skip bi-gram distribution and hence approximations are needed to get the regular bi-gram distribution. Such approximations turn out to be not only computationally complex but also results in poor lower order model

same context i.e. `I go to work on`, then it is likely that such a model would predict both `tuesday` and `wednesday` given the context `today is`. If such a complex model is now used as a generative model, then it would possibly generate not only `tuesday` but also `wednesday` given that the context of words `today is` has already been synthesized. Thus such a technique would increase the coverage of n-grams and essentially improve the performance of language models in speech recognition systems.

Samples of some simulations that were produced for our experiments can be seen in Appendix C.

5.3.2 Gibbs Sampling

We *simulate* text using the distribution P. Given the start-of-sentence symbol <s>, we sample the next word from the probability distribution conditioned on <s>, and continue generating words conditioned on already generated words, i.e. given the sequence $w_1\, w_2\, \ldots\, w_{l-1}$ of words so far, the l^{th} word is sampled from $P(\cdot|w_1, \ldots, w_{l-1})$, conditioned on the entire past. Our estimate of Q^* is simply an n-gram model $\hat{Q}^*$ based on this synthetically generated text[6].

If P is stationary and ergodic[7], then the simulated corpus, L, will have the underlying distribution P, and then from the consistency of the maximum likelihood estimator [6], we can show that by solving (5.2), we essentially find out the maximum likelihood estimate, $\hat{Q}^*$, in the n-gram family, based on the simulated corpus as shown below:

$$\begin{aligned} Q^* &= \operatorname*{argmin}_{Q \in \mathcal{Q}} D(\mathbf{P}\|\mathbf{Q}) \\ &= \operatorname*{argmin}_{Q \in \mathcal{Q}} \sum_{X \in \mathcal{X}} P(X) \ln \frac{P(X)}{Q(X)} \\ &= \operatorname*{argmax}_{Q \in \mathcal{Q}} \sum_{X \in \mathcal{X}} P(X) \ln Q(X) \\ &= \operatorname*{argmax}_{Q \in \mathcal{Q}} \lim_{L \to \infty} \sum_{l=1}^{L} \frac{1}{L} \log Q(X_l) \\ &= \operatorname*{argmax}_{Q \in \mathcal{Q}} \lim_{L \to \infty} \sum_{l=1}^{L} \log Q(X_l) \end{aligned} \tag{5.6}$$

[6] While implementing this idea practically, we chop the word segments at the end of every end-of-sentence marker – </s>. Note that since </s> is treated as a regular word in our model (i.e. RNNs) and it gets generated which in turn allows the generative model to generate new words when also conditioned on this word token. This, however, may not be true for other long span models and hence for such other models it might be necessary to re-start the generation process afresh from the start of the sentence token – <s>.

[7] which is true for language models

Here for the sake of keeping the equations easy to follow, the input alphabet is denoted by $\mathcal{X}$ to sentences possible in our language. Note that Eqn. 5.6 is essentially a maximum likelihood solution. Thus minimization of KL divergence boils down to finding maximum likelihood solution on the simulated corpus. The ML solution can be found out using n-gram models. Below we state the ML proposition without proof [98; 66].

Proposition 5.3.4 *If we have a distribution P which is parameterized by some parameter, θ_0, then a Maximum Likelihood (ML) estimator, $\hat{\theta}_n$, is consistent i.e. that if we have sufficiently large number of observations n (generated under P_{θ_0}), it is possible to find the value of θ_0 with arbitrary precision. As n goes to infinity the estimator $\hat{\theta}_n$ converges in probability to its true value:*

$$\hat{\theta}_n \xrightarrow{p} \theta_0 \tag{5.7}$$

If we assume $x_1, x_2, \ldots$ are sampled from P_{θ_0} and define the likelihood function as $L_n(\theta) = \prod_{i=1}^{n} P_\theta(x_i)$, then a maximum likelihood estimator is defined as any function $\hat{\theta}_n = \hat{\theta}(x_1, \ldots, x_n)$ such that $L_n(\hat{\theta}_n) = \sup_{\theta \in \Theta} L_n(\theta)$. The MLE maximizes:

$$\frac{1}{n} \log L_n(\theta) - \frac{1}{n} \log L_n(\theta_0) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{P_\theta(x_j)}{P_{\theta_0}(x_j)} \xrightarrow{a.s.} -D(P_{\theta_0} || P_\theta) \leq 0$$

which converges by the law of large numbers (Proposition 5.3.4) to an expression that is maximized at 0 if and only if $\theta_0 = \theta$ according to lemma 5.3.2.

In the context of our problem, in the limit, as the sampled text size increases to countably infinite, any distribution, $\hat{Q}^*$, maximizing the likelihood of this sampled text will have 0 KL Divergence with P, provided this distribution also predicts the next word given all the previous context [8], i.e. that:

$$\lim_{n \to \infty} \lim_{L \to \infty} KL(P || \hat{Q}^*) = 0, \tag{5.8}$$

where L is the size of the simulated corpus and n the order of the Markov approximation.

In practice, however, we choose n such that first pass decoding is tractable; L is chosen as large as possible, subject to memory constraints and diminishing returns in LVCSR performance. Since L is finite, for pragmatic purposes, smoothing of n-gram probability distribution (for modeling Q) allows us to approximate the maximum likelihood probability had we seen an infinite corpus ($L \to \infty$) generated by P, thus we obtain the solution using the following approximation of Eqn. 5.6:

$$\hat{Q}^* \approx \operatorname*{argmax}_{Q \in \mathcal{Q}} \sum_{l=1}^{L} \log Q(X_l) \tag{5.9}$$

[8] We are assuming that we have sampled sufficiently enough times for Maximum Likelihood estimate to be as close to the underlying *true* (but known) distribution as possible.

where $\mathcal{Q}$ is chosen to be some finite order n-gram family and $L < M$ where M is some threshold deciding the size of the corpus simulated.

But now the question is whether among all the finite order n-gram models with fixed n, the maximum likelihood finite order n-gram model estimated from the simulated corpus is closest to the generating model (say an m-gram model with $m > n$) ? The answer is yes and it is clear from Eqn. 5.6 when the family of distributions, $\mathcal{Q}$, is restricted to be of order n.

In any case we do not know the true distribution generating the true text. However, with our approach, we can at least come close to the generative model of our choice. If that generative model happens to be better than all the other competing generative models (for instance Recurrent Neural Language Model is better than n-gram LM), then our approximated model would best imitate it and hopefully be better than the n-gram LM built from the training data using ML principles.

5.4 A Recurrent Neural Net Language Model

It is well known that humans can exploit longer context with great success in guessing the next word in a sentence. It seems natural therefore to construct LMs that implicitly capture temporal information of arbitrary length. Our recent work with a recurrent neural network language model (RNN LM) does so [1], and has shown remarkable improvements in perplexity over n-gram LMs, along with improvement in recognition accuracy. RNN LMs also outperform some combinations of syntactic and n-gram models [2]. We therefore use the RNN LM of [41] as our long-span model, which we will try to approximate via n-grams.

The network has an input layer x[9], a hidden layer s (also called state or context layer) and an output layer y. Input to the network at time t is denoted $x(t)$, output $y(t)$, and the hidden state $s(t)$. The input $x(t)$ is formed by concatenating a vector $w(t)$, which represents the current word, with output from the context layer $s(t-1)$ to capture long-span dependencies. We refer the readers to chapter 2 for more details.

5.5 Experiments, Results and Discussion

We report experimental results on four corpora. Perplexity measurements on the WSJ Penn Tree-Bank show that a variational 5-gram is competitive with the best reported results for syntactic LMs. Word error rate (WER) reduction in adapting a Broadcast News language model to the MIT Lectures data is shown next. WER reductions are also demonstrated on the NIST 2007 Meeting Recognition (rt07s) and the NIST 2001 Conversational

[9] In order to maintain consistency of notations with already published work in conferences and journals, we have decided to continue using the variable x to denote the input layer, even though this choice of variable has been made to denote words of vocabulary in this chapter.

Telephone Recognition (eval01) test sets. Finally, improvements are also shown on a very competitive setup of Broadcast News – rt04.

5.5.1 Perplexity Experiments on WSJ

We trained n-gram and RNN LMs on Sections 0-20 (1M words) of the Penn Tree-Bank corpus, and measured their perplexity on Sections 23-24 (0.1M words). Sections 21-22 were used as a held out set for parameter tuning.

Baselines: We used interpolated Kneser Ney smoothing to build 3-gram and 5-gram LMs; we will call them the KN models. We also trained an RNN LM, which we will call RNN-Full. To obtain an alternative long-span model we also trained a *cache* LM from the same training data.

For all models, the vocabulary comprised the $10K$ most frequent words in Sections 0-20.

Variational Approximations: We sampled about $230M$ word tokens using RNN-Full as a generative model. From this sampled corpus, we estimated a 3-gram and 5-gram Kneser Ney smoothed LMs. We will call them the VarApxRNN models. Each of these n-gram approximations was also interpolated with the corresponding n-gram LM estimated from (only) the original LM training corpus; these interpolated LMs will be called the VarApx+KN models. We *simulate* text using the distribution RNN-Full. Given the start-of-sentence symbol <s>, we sample the next word from the probability distribution conditioned on <s>, and continue generating words conditioned on already generated words, i.e. given the sequence $w_1\, w_2\, \ldots\, w_{l-1}$ of words so far, the l^{th} word is sampled from $P(\cdot|w_1, \ldots, w_{l-1})$, conditioned on the entire past. Our estimate of Q^* is simply an n-gram model $\hat{Q}^*$ based on this synthetically generated text. In practice, we chop the word segments at the end of every end-of-sentence marker – </s>. Note that since </s> is treated as a regular word in our model and it gets generated which in turn allows the generative model to generate new words when also conditioned on this word token.

The first column of Table 5.1 shows that VarApxRNN performs as well as the KN model of the same n-gram order, and their interpolation, VarApx+KN, outperforms both of them. Since the VarApxRNN model is trained on only the *simulated* text, interpolating it with KN introduces the knowledge present in the original training data (sections $0-20$) bringing the simulated statistics closer to the true distribution. To our knowledge, the perplexity of the RNN-full model is significantly lower than any n-gram model reported in the literature.

Figure 5.1 empirically supports the asymptotic validity of (5.8) in the size L of the simulated corpus and model order n.

Comparison with Other Long-Span LMs: An advantage of choosing the Penn Tree-Bank corpus and the particular training/test partition is that several other researchers have reported preplexity on this setup using various long-span LMs. The second column of Table 5.1 collects a few such results.

- The random forest language model (RFLM) of Xu [95] asks questions of only the tri-

Setup	PPL	Setup	PPL
KN (3g)	148	Random Forest (Xu)	132
VarApxRNN (3g)	152	-	-
VarApx+KN (3g)	**124**	-	-
KN (5g)	141	SLM (Chelba)	149
VarApxRNN (5g)	140	SLM (Roark)	137
VarApx+KN (5g)	**120**	SLM (Filimonov)	125
VarApx+KN + Cache	111	X-Sent (Momtazi)	118
RNN-Full	102	-	-

Table 5.1: LM Perplexity on Penn Tree-Bank Sections 23-24. These results suggest that RNN-Full is a good approximation to the true distribution of the WSJ text. As a result, VarApx+KN (5g) does exceedingly well in comparison to more complex models which suffer higher variance due to the limited (1M words) text corpus.

gram history, and is therefore comparable with VarApxRNN (3g) and VarApx+KN (3g). The RFLM estimates a better 3-gram model from existing text; by contrast, VarApxRNN performs simple estimation from simulated text. It appears that VarApx+KN (3g), which combines simulation with the original text, is better.

- Structured language models have been proposed by Chelba and Jelinek [33], Roark [76] and Filimonov and Harper [37] to exploit *within sentence* long-span dependencies. Table 5.1 suggests that they are outperformed by VarApx+KN (5g), i.e. by simulating text with RNN-Full and estimating KN 5-gram models.
- Across-sentence dependencies are exploited in the model of Momtazi et al [47]. This performance is nearly matched by VarApx+KN (5g), which only uses the 5-gram context. Moreover, the across-sentence model is a complex interpolation of many word and class models with regular and skip n-grams. The interpolation of VarApx+KN (5g) with another tractable long-span LM, namely the cache LM, outperforms the across-sentence model.

These results suggest that RNN-Full is actually a good approximation to the true distribution of the WSJ text, and the reduction in variance by simulating 300M words of text offsets the bias of the n-gram LM estimated from it. As a result, VarApx+KN (5g) outperforms more sophisticated models that have smaller bias, but suffer higher variance due to the limited (1M words) text corpus.

Table 5.2 below shows how many novel n-grams were introduced by the variational model. It can be seen that out of a total of 82430 5-grams that can be extracted from the evaluation section of the Penn Corpus Sections 23-24, the variational model needs to back off for a total of 73405 $(= 82430 - 9025)$ 5-grams while the standard baseline model needs to back off for a total of $78397 (= 82430 - 4033)$ 5-grams. The variational model thus creates many new and *relevant* 5-grams which helps in doing a much better prediction of the text data. It can also be seen that the variational model backs off to relatively higher

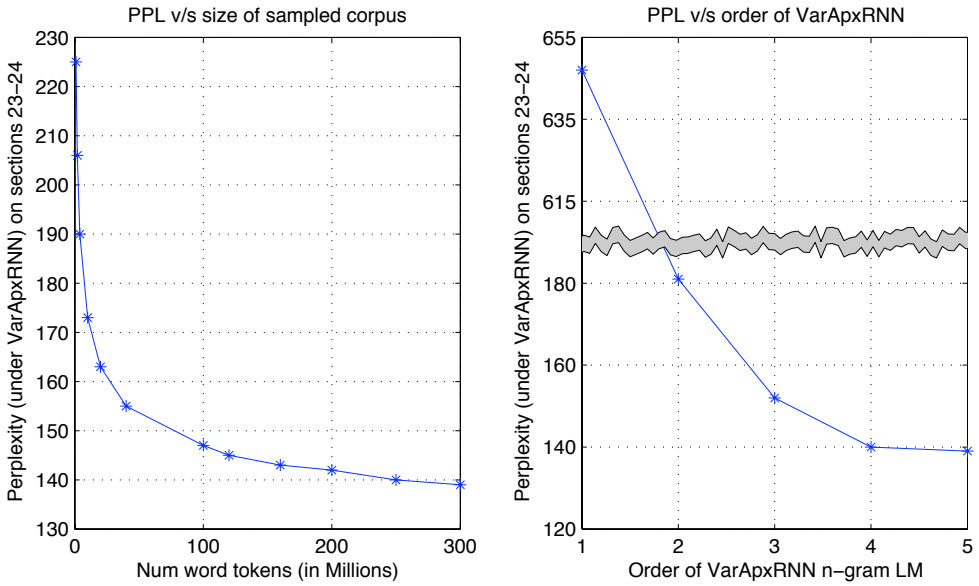

Figure 5.1: The perplexity of Sections 23-24 as a function of (left) the size L of the simulated corpus for model order $n = 5$ and (right) the order n of the model for corpus size $L =$ 300M. These results support (5.8), but also suggest that VarApxRNN (5g) is still far from the RNN LM, whose perplexity is 102.

order models much more frequently than the baseline model. The baseline model backs off all the way to unigram models 15594 times, while the variational model backs off to unigram model only 2825 times. We have observed that most of the speech recognition errors contributed by language model are due to backing off to lower order models. Lower the order the language model backs off to, more are the errors induced. The variational model generates novel and relevant n-grams of varying order and reduces the need to back-off to weaker lower order models. This not only helps in reducing the perplexity but also word error rate.

# of 5-grams in eval section s.t.	**KN(5g)**	**VarApx+KN(5g)**
No Backing Off	4033	9025
Backing off to 4-grams	5151	13056
Backing off to 3-grams	16098	29657
Backing off to 2-grams	41554	27867
Backing off to 1-grams	15594	2825

Table 5.2: Richness of variational model can be seen by noting that relatively fewer number of 5-grams in the evaluation section of the Penn Corpus had to back off to lower order models.

But now the question is whether all of the generated n-grams are relevant? Since the sampling procedure does not filter out *bad* histories, it is likely that the model may produce nonsensical n-grams. In order to test how many n-grams are really relevant, we scour the world wide web (WWW) data. Although a language model built from the web may not be a good one (mainly due to poor normalization across domains and genres), the web will at least be a good corpus to test if a sub sequence appeared anywhere or not. Using innovative web scouring tools developed by Google and Hopkins scientists [99], we take each n-gram as produced by variational model and check if such a pattern exists in the web. However, before we carry out such a study, it becomes necessary to establish a baseline. We hence take valid n-grams from the language model trained on the original training data (sections 0-20) and scour the web for each of them. Table 5.3 below shows the coverage of n-grams when checked against the web data. The n-grams that were used to query the web were filtered of those n-grams which carried `<unk>`, `<s>` and `</s>`. The n-grams of the variational model were further filtered of those n-grams which were present in the baseline model. Thus the n-grams corresponding to the variational model were purely *synthetic*.

Model	**# of n-grams queried**	**# n-grams found in web**	**Coverage**
KN(5g)	50073	42824	85.5%
VarApx(5g)	5779012	4160889	72%

Table 5.3: Coverage of n-grams as checked against the web data. It can be seen that around 72% of the n-grams produced by the variational model are found on the web.

From Table 5.3 we can see that the variational model not only generates many novel n-grams but about 70% of them are in fact found in the web implying that these n-grams are not some random non-sensical patterns. 85% of the regular n-grams (as seen in the original training data) are seen in the web implying that the remaining 15% of the n-grams are corpus specific. It is hence reasonable to assume that out of the remaining 28% of the synthetic n-grams, many n-grams may still be sensical but probably too corpus specific !!

5.5.2 Domain Adaptation Experiments on MIT Lectures

We performed recognition on the MIT lectures corpus [100] using state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio), provided to us by IBM [91]. IBM also provided us its state-of-the-art speech recognizer, Attila [92] and two LMs containing 54M and 4.7M n-grams ($n \leq 4$) that were trained on BN text (400M word tokens). 150K words of MIT lecture transcripts were available as in-domain text.

Interpolated Kneser Ney smoothed n-gram models built with the 150K word in-domain corpus were interpolated with corresponding n-gram LMs from IBM. We, however, used only 4.7M n-gram LM for interpolation as 54M n-gram LM did not provide us any extra benefit. We will call this interpolated model as KN:MIT+BN (interpolation of MIT LM

and IBM's 4.7M $n$gram LM.). The order of the language model used in first pass decoding was 3, while the order of the LM used for lattice re-scoring was 4. The RNN LM trained on the 150K words (only) will be called RNN-Full as before. We set the size of the hidden layer to 100.

We simulated text (30M word tokens) using RNN-Full, and estimated n-gram LMs from it, which we will again call VarApxRNN. Models resulting from the interpolation of VarApxRNN and KN:MIT+BN n-gram LMs of the same order will be called VarApx+KN.

We followed IBM's multi-pass decoding recipe [92] using 3-gram LMs in the first pass decoding, generated word lattice and N-best list, and re-scored them with either bigger n-gram LM or RNN-Full. Table 5.4 shows the WER for different decoding configurations, contrasting standard n-gram LMs with the corresponding VarApx+KN LMs. Note that since our rescoring model was a long span model, it was infeasible to carry out *exact* lattice rescoring and hence instead we extracted N best lists. We also made use of Iterative Decoding techniques (see Chapter 4 for details) for re-scoring word lattices directly, whenever N best list decoding yielded sub-optimal results.

We used two sets for decoding. The audio for each set was about 2.1 hours long.

Setup	**Set 1**	**Set 2**
KN:MIT+BN (3g) Decoding	24.8	22.4
+ KN:MIT+BN (4g) Lattice rescoring	24.8	22.4
+ RNN-Full rescoring (**100** best)	24.1	22.4
+ RNN-Full rescoring (**2000** best)	23.8	21.6
Oracle (**2000** best)	17.9	15.5
VarApx+KN (3g) Decoding	24.4	22.2
+ VarApx+KN (4g) Lattice Rescoring	24.1	21.7
+ RNN-Full rescoring (**100** best)	23.8	21.7
+ RNN-Full rescoring (**2000** best)	23.6	21.5
Oracle (**2000** best)	17.5	15.1

Table 5.4: Performance (%WER) on the MIT Lectures data set. Decoding with VarApx+KN consistently produces lattices with lower *oracle* WER compared to lattices produced by standard n-gram models. The 1-best output with VarApx+KN also is better than its standard n-gram counterpart.

The improvements using variational models is clear from the Table 5.4. Use of a higher order n-gram LM improves the speech recognition performance significantly.

In the perplexity based experiments we saw that many synthetic n-grams are actually seen elsewhere in web implying that quite a lot of them are sensical. From the speech experiments we can see that an increased coverage also brings down the word error rate. We did the following experiment to illustrate this point. For the first set of the above setup, we found out the n-gram instances in the reference transcripts such that the n^{th} word was either substituted by some other word or was deleted during the automatic transcription. For instance in the toy reference and aligned hypothesis shown below, we can see that the word

`for` was erroneously substituted by the word `four` and the word `us` got deleted. We don't consider false insertions in hypotheses because this information cannot be obtained from reference transcripts alone and in this analysis we care only about n-grams in the reference transcripts such that the predicted word does not get transcribed correctly.

```
REF: today is a holiday for    us
HYP: today is a day     four   **
```

For each such word in the reference transcript, we extracted the n-grams (trigrams corresponding to the words `for` and `us` will be `a holiday for` and `holiday for us`). We then found the coverage of such n-grams in the language model. For the hypotheses generated under the baseline models i.e. KN:MIT+BN (4g) model, Table 5.5 summarizes the findings:

Model	**# of n-grams queried**	**# n-grams found in LM**	**Coverage**
KN:MIT+BN (4g)	3204	635	19.81%
VarApx+KN(4g)	3204	1117	34.86%

Table 5.5: Coverage of falsely recognized (wrt baseline model) n-grams as checked against the language models. It can be seen that the baseline model has a very poor coverage while the variational model has nearly double coverage. Increasing the coverage is likely to improve the performance of the speech recognition system.

From Table 5.5 we observe that out of 3204 total n-grams such that the n^{th} word is falsely recognized by the model using KN:MIT+BN(4g) models, only 19.81% of these were found in this LM. As against that, if we search these n-grams in the VarApx+KN(4g) model, we see that 34.86% of them are found. While an improved coverage does not necessarily mean that the WER will be reduced too[10], but at-least the variational model is able to predict these n-grams while the baseline model does not even have them and a prediction for such n-grams would require them to backoff to poor lower order models.

Similarly, we extracted the n-grams from the reference with respect to the hypotheses generated under the variational model. Table 5.6 summarizes the findings:

Although the coverage in percentage seems to be the same for both tables 5.5 and 5.6, it should be noted that there are fewer errors in the output of the variational model and the total number of n-grams seen in the variational model is also small in the latter case. This may imply that an improved coverage of n-grams influences reduction of WER to a large extent.

[10]The LM may have an entry for a particular n-gram, but if it is assigned a very low probability, then probably the recognizer wont be able to produce it.

Model	# of n-grams queried	# n-grams found in LM	Coverage
KN:MIT+BN (4g)	3141	600	19.10%
VarApx+KN(4g)	3141	1067	33.97%

Table 5.6: Coverage of falsely recognized (wrt variational model) n-grams as checked against the language models. The absolute number of n-grams covered by the language models are less than that in Table 5.5 implying that an improved coverage of n-grams influences reduction of WER to a large extent.

5.5.3 Conversational Speech Recognition Experiments

We demonstrate WER improvements in two conversational speech recognition tasks: the transcription of multiparty meetings, and of conversational telephone speech. Brno's variant of the AMI system, developed for the NIST Meeting Transcription evaluation [101], was used for the former, and the Brno conversational telephone speech (CTS) system for the latter.

The AMI recognizer used fast speaker adaptation (HLDA, CMLLR and VTLN); it processed PLP+NN-posterior features extracted from 16kHz audio with SAT models trained on 200 hours of meeting data. The CTS recognizer used an initial decoding pass for VTLN and MLLR, and processed PLP features extracted from 8kHz audio with SAT models trained on 270 hours of telephone speech. All acoustic models were trained using the MPE criterion and used cross-word tied-state triphones, and both setups produced bigram lattices using a 2-gram LM trained using Good-Turing discounting, which were subsequently expanded to 5-gram lattices using a modified Kneser-Ney smoothed LM.

5M words of Fisher CTS transcripts were used as training text for three LMs: two n-grams and an RNN. We call the 2-gram model with Good-Turning discounting GT (2g). The 5-gram model and the RNN model are called KN (5g) and RNN-Full, as before. 400M words of text generated from RNN-Full LM via Gibbs sampling were used to estimate additional n-gram LMs, which we again call VarApxRNN. Altogether, this resulted in a total of four LM configurations, 2-gram vs 5-gram $\times$ standard n-gram vs variational approximation. Since the LMs were trained on CTS transcripts, they are in-domain for conversation telephone recognition (eval01), but out-of-domain for meeting recognition (rt07s).

The four LMs were applied to rt07s and eval01, and the WERs are reported in Table 5.7. The table also illustrates the WER when N-best list rescoring with RNN-Full is performed.

5.5.4 Broadcast News Speech Recognition Experiments

An interesting question that lingers in mind is what happens when the original training data itself is huge! By huge we mean number of word tokens in hundreds of millions. To answer this question, we tested the proposed technique on a very competitive setup of Broadcast news. The setup details and results are presented next:

Setup	**eval01**	**rt07s**
GT (2g) Decoding	30.3	33.7
+ KN (5g) Lattice Rescoring	**28.0**	**32.4**
+ RNN-Full rescoring (**100** best)	27.1	30.8
+ RNN-Full rescoring (**1000** best)	26.5	30.5
Oracle (**1000** best)	19.5	21.3
VarApx+GT (2g) Decoding	30.1	33.3
+ VarApx+KN (5g) Lattice Rescoring	**27.2**	**31.7**
+ RNN-Full rescoring (**100** best)	27.0	30.6
+ RNN-Full rescoring (**1000** best)	26.5	30.4
Oracle (**1000** best)	19.5	21.0

Table 5.7: Performance (%WER) on conversational speech data sets. VarApx+KN reduces the WER by 0.7%-0.8% over a 5-gram model on both telephone speech (eval01) and meetings (rt07s). ($*$: Although the Oracle WER for eval01 is same for VarApx+KN and 5-gram model, the false insertions caused due to later model are 0.1%. more than former.)

We performed recognition on the Broadcast News (BN) `rt04` task using state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio) provided to us by IBM [91]. IBM also provided us its state-of-the-art speech recognizer, Attila [92] and two Kneser-Ney smoothed backoff n-gram LMs containing 4.7M n-grams ($n \leq 4$) and 54M n-grams ($n \leq 4$) were trained. We will refer to them as KN:BN-Small and KN:BN-Big respectively. The LM training text consists of 400M words from the following data sources: 1996 CSR Hub4 Language Model data, EARS BN03 closed captions, GALE Phase 2 Distillation GNG Evaluation Supplemental Multilingual data , Hub4 acoustic model training transcripts , TDT4 closed captions, TDT4 newswire, and GALE Broadcast Conversations and GALE Broadcast News.

We trained an RNN based language model, denoted further as RNN-Full, on the entire training data (400M word tokens). It used 640 neurons in the hidden layer.

From the generative model – RNN-Full, we generated about 1B word tokens and estimated a GT smoothed 4-gram language model.[11] The original LM that was estimated from this huge sampled corpus contained over 150M n-grams. We will refer to this LM as VarApprx-Big. We then pruned it using Entropy-Pruning [102] so that the resulting number of n-grams were roughly about 6M. We will refer to this LM as VarApx-Small. Combination of VarApx-Small and KN:BN-Small will be referred to as VarApx+KN-Small and combination of VarApx-Big and KN:BN-Big will be referred to as VarApx+KN-Big. The interpolation weights were tuned so as to reduce the perplexity on some held out data (`dev04f`).

Table 5.8 summarizes the performance (in terms of WER) of various models on the evaluation set `rt04`.

[11] We found that GT smoothing worked better than KN smoothing on bigger data-sets.

Setup	rt04
KN:BN-Small Decoding	**14.1**
+ KN:BN-Big Lattice Rescoring	13.1
+ RNN-Full rescoring	12.1
VarApx+KN-Small Decoding	**13.0**
+ VarApx+KN-Big Lattice Rescoring	13.0
+ RNN-Full rescoring	12.0

Table 5.8: Performance (%WER) on Broadcast News speech data set (`rt04`). VarApx+KN-Small reduces the WER by 1.1% absolute over a 4-gram model when used directly in the first pass decoding. The re-scoring is not very effective, but the first pass output using variational model, significantly improves over the baseline performance.

5.5.5 Analysis and Discussion of LVCSR Results

From Table 5.4 it is clear that using VarApx+KN during decoding consistently produces lattices with a 0.5% lower *oracle* WER compared to lattices produced by standard n gram models. The first pass output from decoding with VarApx+KN also has 0.2% to 0.4% lower WER than from decoding with their standard n-gram counterparts. However re-scoring with language models with 4 gram order produces even better results. We can see 0.7% absolute reduction after the application of 4-gram LMs on the lattices produced by the corresponding 3-gram LMs.[12] It seems fair to conclude that VarApx+KN is a better n-gram model than a standard n-gram model estimated with Kneser Ney smoothing. Unlike RNN-Full, it can be incorporated into the decoder, bringing some of the benefits of RNN-Full to first pass decoding and lattice re-scoring.

Note further from the upper half of Table 5.4 that 2000-best rescoring with RNN-Full reduces WER over a standard 3-gram by 0.8% to 0.9%. In the lower half, using VarApx+KN in decoding shows a different benefit: if VarApx+KN is used for generating the N-best list, the same WER reduction is available at 100-best rescoring! If 2000-best rescoring is undertaken, an additional small gain of 0.2% is obtained.

Figure 5.2 shows that even when the search space size is increased beyond 2000 best, there still remains the gap between the oracle WER of the baseline approach i.e. KN:MIT+BN model and the proposed approach i.e. VarApprox+KN model. These experiments show that variational model produces richer search spaces so that even a sub-optimal search (via N-best) produces much better results during re-scoring with a full blown model.

The benefits of decoding & lattice rescoring with the variational approximation of RNN-Full are even more evident from Table 5.7, where VarApx+KN reduces WER by 0.7%-0.8% over a 5-gram on both CTS and meeting transcription.

A final observation from Table 5.7 is that there still remains a gap between decoding

[12] In [5], we used the 4-gram LMs directly in the first pass and possibly due to aggressive beam pruning, we could not get comparable results. We hence decided to use a more manageable Language Model in the first pass and then rescore with the 4-gram based Language Model. This improved the results remarkably.

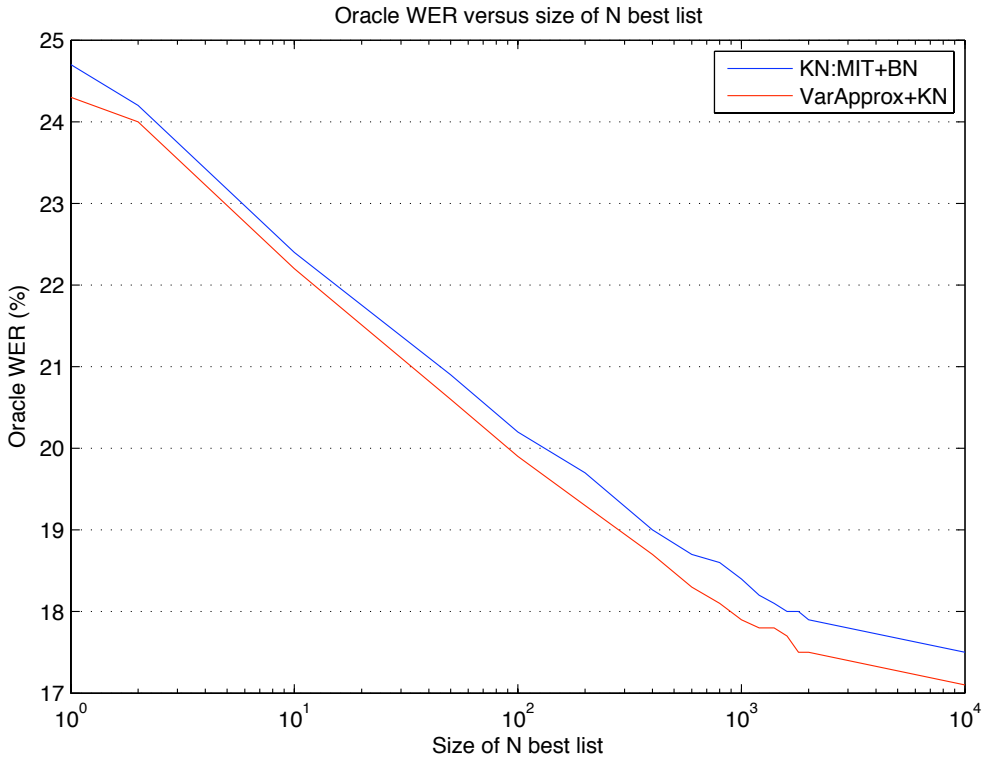

Figure 5.2: Plot of Oracle WER versus the size of the search space in terms of number of hypotheses in N-best list. It can be seen that although the Oracle accuracy increases with an increase in the search space, there still remains a gap in the performance of the baseline and proposed approach, with the latter consistently better.

with VarApx+KN and rescoring with RNN-Full. The latter reduces WER by almost 2% (absolute) over the standard 5-gram, compared to 0.7%-0.8% by the former. This suggests that when the RNN is trained on more data (5M words in Table 5.7 vs 1M words in Table 5.1), it improves even further over a 4- or 5-gram model. One may need to investigate further increasing the amount L of simulated data and/or the order n of the approximation in (5.8), or perhaps consider other tractable model families $\mathcal{Q}$ in (5.2).

Results on larger data-set are encouraging too. Table 5.8 shows the improvements in performance when VarApx-KN-Small is used directly in the first pass decoding. It should be noted that while the rescoring LM (KN:BN-Big / VarApx+KN-Big) in this setup is 10 times bigger than the LM used in the first pass (KN:BN-Small / VarApx+KN-Small), the re-scoring for this experiment is relatively computationally intensive as compared to that in other experiments. It is hence interesting to see that the using VarApx+KN-Small model in the first pass not only brings down the WER by 1.1% absolute but it also performs better than the re-scoring model KN:BN-Big (or even VarApx+KN-Big). Unfortunately, we did

not see any benefit of rescoring lattices with VarApx+KN-Big. It is very interesting to find that even for bigger datasets, such as the one chosen for this setup, the gains obtained using even a small sized variational model is significant.

All our n-gram decoding results, in all the setups used, are statistically significant (with a $p \leq 0.001$). We used NIST's *sclite* toolkit for computing statistical significance using MAPSSWE significance tests.

5.6 Summary of the Chapter

In this chapter, we have presented experimental evidence that (n-gram) variational approximations of long-span LMs yield greater accuracy in LVCSR than standard n-gram models estimated from the same training text. The evidence further suggests that the approximated LMs also yield higher quality lattices in terms of the *Oracle* WER, and result in more efficient N-best rescoring with the long-span LMs. Both these results advocate for *early integration* of long-span LMs during decoding, even if only in their approximated forms. Finally, there is preliminary evidence that the RNN LM improves significantly over n-grams with increasing training data, calling for an investigation of more powerful tractable approximations.

5.7 Additional Reading

An interesting work at IBM [103] involving variational methods for acoustic modeling is worth mentioning here. The authors discuss the problem of acoustic modeling with bootstrapping and restructuring on full covariance matrices. For low resource languages such as Pashto, authors demonstrate that using techniques such as bootstrap resampling, a robust full covariance matrix for gaussian mixture models for the HMMs could be formed. However, due to computational issues, full covariance matrices could not be easily used during decoding and hence diagonal covariance matrices were desired such that the corresponding gaussian mixture model (GMM) distributions were closest to the original GMM distributions, rendered due to full covariance matrices.

The authors in this work do the down-scaling and diagonalization of full covariance matrices using Monte Carlo based KL minimization (similar in spirit to what we proposed for our approach too) between two GMMs. If we represent the two GMM distributions by $f_1(\cdot)$ and $f_2(\cdot)$ where the former is the distribution given and the latter is the desired, then the optimization function can be written as:

$$f_2^*(x) = \underset{f_2(x)}{\operatorname{argmin}} \int_x f_1(x) \log \frac{f_1(x)}{f_2(x)} dx$$

which amounts to the following optimization problem instead:

$$f_2^*(x) = \underset{f_2(x)}{\operatorname{argmax}} E_{f_1}[\log f_2(x)]$$

The above equation reveals that the optimization can be carried out by first sampling from the given reference distribution $f_1(x)$ and then fitting the samples with the desired GMM model under the maximum likelihood.

The usefulness of variational methods (both for acoustic and language modeling) suggests that there ought to be an end-to-end optimization that is variational from front to back resulting in a computationally tractable system.

Chapter 6

Conclusion and Future Directions

6.1 Summary of the thesis

In this thesis, we have presented many novel ideas and techniques to solve some of the computationally complex problems of decoding and re-scoring in speech recognition systems. While some problem were intractable due to the long span nature of the language model, other problem were computationally complex due to the nature of the task and the limitation of the decoder. We investigated into these issues and presented interesting and novel solutions. We summarize the key contribution of the thesis as follows:

- It presented empirical evidence of the fact that by combining many powerful statistical language models, the net performance of the system improves significantly. New state-of-the-art results were obtained on a very standard task for language modeling. It also demonstrated an efficient training framework for combining many complex language models for their eventual use in decoding in speech recognition systems.
- It discussed a decoding scheme enabling the use of long span language models or for that matter combination of many complex long and short span language models to carry out re-scoring of big word graphs. Experiments were carried out on a very competitive speech task creating a new state-of-the-art performance.
- It presented a variational inference scheme for inferring a tractable model from a long span language model so that it could be used directly in the first pass decoding of speech recognition system. Experiments were carried out on many speech tasks and the efficacy of the method was shown to be consistent throughout.

6.2 Future Directions

In this section we discuss possible extensions to two main ideas discussed in this thesis (presented in chapter 4 and 5 respectively). In section 6.2.1 we first discuss a possible

extension of iterative decoding method and in section 6.2.2 we discuss possible extensions of variation approximation based decoding.

6.2.1 Extensions of Iterative Decoding

Iterative decoding method presented in this thesis is very useful to carry out minimum Bayes risk (MBR) decoding under long span language models. Stolcke et.al. previously presented the MBR decoding framework for N-best lists [104]. Such a framework is useful for incorporating long span language models or combination of complex language models and carry out MBR decoding. However, as we have already seen in this thesis, N-best lists are not always a favorable representation of the search space and there is a need to explore more hypotheses. Word lattice encode exponential number of hypotheses and hence this search space becomes an obvious choice. Goel et.al. previously proposed the use of segmented lattice structure for carrying out MBR decoding under n-gram LMs [58]. However, their method is limited by the use of only n-gram language models and do not scale to language models capturing longer dependencies. We present an outline of the iterative decoding algorithm for carrying out MBR decoding under long span language models or for that matter, combination of many complex language models capturing local and long distance dependencies.

Once we have formed the self contained lattices (see Sec. 4.2.1 for a quick review), $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_C$, where C is the total number of sub lattices formed, then the idea is to divide the global MBR decoding problem into many small local MBR decoding problems carried over the sub lattices one at a time by fixing single best paths from all the remaining sub lattices, in a round robin fashion. The approach is very similar to that presented in Sec. 4.2.2 except that instead of *ascending the hill of likelihoods*, for MAP decoding, we now have to *descent the valley of Bayes risk*, for MBR decoding.

Algorithm 7 below illustrates the steps. The inputs to the algorithm are the sub lattices (produced by cutting the parent lattice generated under some Markov n-gram LM) and a new rescoring LM, which now need not be restricted to finite state machine family. The output of the algorithm is a word string, $\mathbf{W}^*$, such that it is the concatenation of final decoded word strings from each sub lattice. Thus if we denote the final decoded path (under some decoding scheme, which will become apparent next) in the j^{th} sub lattice by π_j^* and the concatenation symbol by '$\cdot$', then $\mathbf{W}^* = W[\pi_1^*] \cdot W[\pi_2^*] \cdot \ldots \cdot W[\pi_C^*] = \odot_{j=1}^{C} W[\pi_j^*]$.

The algorithm is initialized by setting **PrevHypo** to null and **CurrHypo** to the concatenation of 1-best output from each sub lattice. During the initialization step, each sub lattice is analyzed independent of any other sub lattice and under the baseline acoustic scores and baseline n-gram LM scores, 1-*best* path is found out.[1] Thus if we define the best path under

[1] We can either initialize the starting solution as the concatenation of MAP paths from each sub lattice or as the concatenation of MBR paths.

Algorithm 7 Algorithm for Iterative Decoding on word lattices for MBR.

Require: $\{\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C\}$, L_{new}
  PrevHyp $\leftarrow$ null
  CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  while PrevHyp $\neq$ CurrentHyp **do**
    for $i \leftarrow 1 \ldots C$ **do**
      $\hat{\pi}_i \leftarrow \underset{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}}{\operatorname{argmin}} \hat{\mathbf{E}}_{P_{new}[\hat{\pi_1} \cdot \ldots \cdot \boldsymbol{\pi}_i \cdot \ldots \cdot \hat{\pi_k}]}[\text{EditDist}(\pi_i, \boldsymbol{\pi}_i)]$
    end for
    PrevHyp $\leftarrow$ CurrentHyp
    CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  end while
  $\forall j \in \{1, 2, \ldots, C\} \quad \pi_j^* \leftarrow \hat{\pi}_j$

baseline model in some j^{th} sub-lattice by $\hat{\pi}_j$, then it is found out simply as:

$$\hat{\pi}_j = \underset{\substack{\pi_j \in \mathcal{E}_j^*: \\ p[\pi_j]=n_s[\mathcal{L}_j] \\ n[\pi]=n_e[\mathcal{L}_j]}}{\arg\max} A[\pi_j]^{\gamma} L[\pi_j],$$

CurrHypo is then initialized to: $W[\hat{\pi}_1] \cdot W[\hat{\pi}_2] \cdot \ldots \cdot W[\hat{\pi}_C]$. The algorithm then runs as long as CurrHypo is not equal to PrevHypo. In each iteration, the algorithm sequentially MBR decodes each sub-lattice by keeping the surrounding context fixed. Once all the sub lattices are decoded with MBR scheme, that constitutes one iteration. At the end of each iteration, CurrHypo is set to the concatenation of best paths from each sub lattice while PrevHypo is set to the old value of CurrHypo. Thus if we are analyzing some i^{th} sub-lattice in some iteration, then best paths from all but this sub-lattice is kept fixed and a *new best* path under the re-scoring LM is found out as shown below.

$$\hat{\pi}_i = \underset{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}}{\operatorname{argmin}} \hat{\mathbf{E}}_{P_{new}[\hat{\pi_1} \cdot \ldots \cdot \boldsymbol{\pi}_i \cdot \ldots \cdot \hat{\pi_k}]}[\text{EditDist}(\pi_i, \boldsymbol{\pi}_i)] \tag{6.1}$$

where

$$P_{new}[\hat{\pi_1} \cdot \ldots \cdot \boldsymbol{\pi}_i \cdot \ldots \cdot \hat{\pi_k}] = \frac{1}{Z(i)} L_{new}[\hat{\pi_1} \cdot \ldots \cdot \boldsymbol{\pi}_i \cdot \ldots \cdot \hat{\pi_k}] \times A[\boldsymbol{\pi}_i]^{\eta} \prod_{\substack{j=1 \\ j \neq i}}^{k} A[\hat{\pi_j}]^{\eta}$$

with $\boldsymbol{\pi}_i$ denoting the random variable taking values in the event space defined by the i^{th} sub-lattice and

$$Z(i) = \sum_{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}} L_{new}[\hat{\pi_1} \cdot \ldots \cdot \pi_i \cdot \ldots \cdot \hat{\pi_k}] \times A[\pi_i]^{\eta} \prod_{\substack{j=1 \\ j \neq i}}^{k} A[\hat{\pi_j}]^{\eta}$$

The left context: $W[\hat{\pi}_1] \cdot W[\hat{\pi}_2] \cdot \ldots \cdot W[\hat{\pi}_{i-1}]$ and right context $W[\hat{\pi}_{i+1}] \cdot W[\hat{\pi}_{i+2}] \cdot \ldots \cdot W[\hat{\pi}_C]$ is fixed during this local MBR decoding. η is the new fudge factor which may or may not be equal to γ depending upon the dynamic scale variability between two language models: baseline starting model L and new rescoring model L_{new}. EditDist(π_i, π_j) computes the edit distance between $W[\pi_i]$ and $W[\pi_j]$ while $\hat{\mathbf{E}}[\cdot]$ computes the approximated expectation by restricting the probability distribution on the event space defined by the paths of the sub-lattice under consideration (in the algorithm it is the i^{th} sub-lattice).

6.2.2 Variational Inference for other complex LMs

The idea of variational inference can be easily extended to other complex LMs such as feedforward neural network LM, maximum entropy LM, decision tree based LM, among many others. All of these LMs generalize very well on the events not seen during their training. However, these models are computationally complex if used in first pass decoding, sometimes even during word lattice re-scoring. Feedforward neural network LM and maximum entropy LM smooth the probability distribution either by having a distributed representation of words or by assigning uniform probability from the left over mass. Decision tree based LMs form equivalence classification of histories, thus pooling many histories together to effectively fight the data sparsity problem. However, if we explicitly find an n-gram model such that it is closest to these models, then the resulting variational model may *complement* the n-gram model (built from the training data), by supplying many novel and relevant n-tuples, thus reducing the need to backoff (something which the above mentioned complex models are extremely good at when compared to the n-gram model built from the same training data).

Appendix A

Expectation Maximization

A very powerful method for finding maximum likelihood solutions for models which do not yield a closed form solution is called Expectation Maximization algorithm or EM algorithm [105]. EM is an iterative algorithm which gradually improves the likelihood of the model. However, it is not guaranteed to find global optimum solutions. We will demonstrate the use of EM algorithm for finding optimal parameters (interpolation weights) when the task is of combining language models linearly.

A.1 Language Model Interpolation

Input: We are given M language models, being characterized by corresponding M probability distributions. We will denote the probability distribution on word $w \in \mathcal{V}$ given the context of previous words denoted here by $\mathbf{h}$, under some m^{th} model by $P_m(w|\mathbf{h})$. $\mathcal{V}$ represents the vocabulary.

Output: Set of parameters $\{\lambda_1^*, \ldots, \lambda_M^*\}$ such that $\sum_{m=1}^{M} \lambda_m^* = 1$ and the probability distribution $P(w|\mathbf{h}) = \sum_{m=1}^{M} \lambda_m^* P_m(w|\mathbf{h})$ such that it maximizes the likelihood of sequence of words $W \equiv w_1, w_2, \ldots, w_T$.

Assumptions: We will assume that each language model can be factorized linearly using chain rule. Thus the likelihood of the word sequence W, under some model $P(\cdot)$, can be obtained using chain rule by putting equivalence mapping on the conditioning terms. The log likelihood of W is given as:

$$\log P(W) = \sum_{t=1}^{T} \log P(w_t|\phi(w_1^{t-1})) \tag{A.1}$$

where, $\phi(\cdot)$ is the equivalence classification function which maps the history to some equivalence class.

Under some linear interpolation of M models, the log likelihood of the word sequence W is given by:

$$\mathcal{L}(W;\Lambda) = \sum_{t=1}^{T} \log\Big(\sum_{m=1}^{M} \lambda_m P_m(w_t|\phi_m(w_1^{t-1}))\Big) \tag{A.2}$$

where $\phi_m(\cdot)$ is the equivalence mapping used by the m^{th} model and Λ is the set of parameters. Given this linear interpolation model, the goal is to then maximize the likelihood of the data with respect to the parameters (interpolation weights). The direct optimization (see Eqn. A.3) is hard and there is no closed form solution for obtaining optimal Λ.

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}} \sum_{t=1}^{T} \log\Big(\sum_{m=1}^{M} \lambda_m P_m(w_t|\phi_m(w_1^{t-n+1}))\Big) \tag{A.3}$$

We thus have to resort to some hill climbing method and EM turns out to be a good choice.

A.1.1 Introducing Latent Variables

We will follow the setup of gaussian mixture model for EM training as described in [96, pp. 423]. We will formulate the problem of *linear interpolation* in terms of discrete latent variables. Under the linear interpolation, the final probability distribution of word given some history is obtained as:

$$P(w_t|\phi(w_1^{t-1})) = \sum_{m=1}^{M} \lambda_m P_m(w_t|\phi_m(w_1^{t-1})). \tag{A.4}$$

Let us introduce a M dimensional latent variable vector which has 1-of-M encoding. Thus any realization of this random vector has zeros in all position except one position. Corresponding to each word w_t of our data, let us represent this vector by $\boldsymbol{z}_t$. We will denote the sequence of such vectors by $\boldsymbol{Z}$ i.e. $\boldsymbol{Z} \equiv \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T$. We will denote the corresponding random vector by $\mathbf{Z}$. Thus $\boldsymbol{Z}$ is one of the instantiations of $\mathbf{Z}$.

We can now define the marginal distribution of w_t as a function of conditional distribution of w_t given $\boldsymbol{z}_t$ & the previous context and the prior on $\boldsymbol{z}_t$. Thus we can write:

$$P(w_t|\phi(w_1^{t-1})) = \sum_{\boldsymbol{z}_t} P(\boldsymbol{z}_t) P(w_t|\boldsymbol{z}_t, \phi(w_1^{t-1})). \tag{A.5}$$

where it is assumed that $\boldsymbol{z}_t$ is independent of $\{w_1, \ldots, w_{t-1}\}$ and thus $P(\boldsymbol{z}_t|\phi(w_1^{t-1})) = P(\boldsymbol{z}_t)$. Thus the latent variable $\boldsymbol{z}_t$ is dependent and/or affects only the current observation w_t. The marginal distribution over $\boldsymbol{z}_t$ is specified in terms of the linear interpolation coefficients λ_m, such that

$$P(z_{t,m}) = \lambda_m$$

where $z_{t,m}$ is the m^{th} element of the vector $\boldsymbol{z}_t$.

The probability of latent vector $\boldsymbol{z}_t$ can then be represented as:

$$P(\boldsymbol{z}_t) = \prod_{m=1}^{M} \lambda_m^{z_{t,m}}$$

The conditional distribution of w_t given a particular value of $\boldsymbol{z}_t$ such that $z_{t,m} = 1$ is then given by:

$$P(w_t|\phi(w_1^{t-1}), \boldsymbol{z}_t) = P_m(w|\phi_m(w_1^{t-1}))$$

i.e. it is given by the m^{th} language model. The conditional distribution then can be written as:

$$P(w_t|\phi(w_1^{t-1}), \boldsymbol{z}_t) = \prod_{m=1}^{M} P_m(w|\phi_m(w_1^{t-1}))^{z_{t,m}}$$

Combining the conditional distribution and prior and summing over all the possible values of $\boldsymbol{z}_t$[1], we get the marginal distribution of w_t given just the previous context as shown below:

$$\begin{aligned}
P(w_t|\phi(w_1^{t-1})) &= \sum_{\boldsymbol{z}_t} P(\boldsymbol{z}_t) P(w_t|\phi(w_1^{t-1}), \boldsymbol{z}_t) \\
&= \sum_{\boldsymbol{z}_t} \Big(\prod_{m=1}^{M} \lambda_m^{z_{t,m}} \prod_{m=1}^{M} P_m(w_t|\phi(w_1^{t-n+1}))^{z_{t,m}} \Big) \\
&= \sum_{\boldsymbol{z}_t} \prod_{m=1}^{M} \Big(\lambda_m P_m(w_t|\phi(w_1^{t-n+1})) \Big)^{z_{t,m}} \\
&= \sum_{m=1}^{M} \lambda_m P_m(w_t|\phi_m(w_1^{t-1})) \qquad \text{(A.6)}
\end{aligned}$$

From Eqn. (A.6) we can see that the linear interpolation problem can be recasted in terms of discrete latent variables. Once we have the joint distribution of w_t and latent variable $\boldsymbol{z}_t$, we can apply Expectation Maximization technique to find optimal interpolation weights.

A.1.2 Expectation Maximization

Given the linear interpolation model, the goal is to maximize the likelihood of the data with respect to the parameters (interpolation weights). As we discussed before the direct optimization (see Eqn. A.3) is hard and there is no closed form solution for obtaining optimal Λ. We will thus use EM, which involves following 4 steps:

[1] There will be only M possible values of the vector $\boldsymbol{z}_t$.

1. **Initialization:** Initialize the linear interpolation weights λ_m constraining to the condition that $\sum_{m=1}^{M} \lambda_m = 1$ and $\forall m \in \{1, 2, \ldots, M\}$, $\lambda_m \in [0, 1]$. Compute the joint likelihood under these initial values. If we represent these initial values by $\Lambda^{old} = \{\lambda_1^{old}, \ldots, \lambda_m^{(old)}\}$, then the joint log likelihood of W and $\boldsymbol{Z}$ is given by:

$$P(W, \boldsymbol{Z}; \Lambda^{old}) = \prod_{t=1}^{T} P(w_t | \phi(w_1^{t-n+1}), \boldsymbol{z}_t; \Lambda^{old}) \tag{A.7}$$

where again it is assumed that the latent variable $\boldsymbol{z}_t$ depends only on the corresponding observation w_t i.e. is it conditionally independent of the past given the current observation, w_t. Following the algebra for the joint likelihood of w_t and some $\boldsymbol{z}_t$ as presented in Eqn. (A.6), we can write the above joint likelihood as shown below:

$$P(W, \boldsymbol{Z}; \Lambda^{old}) = \prod_{t=1}^{T} \prod_{m=1}^{M} \Big(\lambda_m^{old} P_m(w_t | \phi(w_1^{t-n+1})) \Big)^{z_{t,m}} \tag{A.8}$$

Taking log on both sides, log likelihood of the joint distribution of W and $\boldsymbol{Z}$ can be then written as:

$$\log P(W, \boldsymbol{Z}; \Lambda^{(old)}) = \sum_{t=1}^{T} \sum_{m=1}^{M} z_{t,m} \Big(\log \lambda_m^{old} + \log P_m(w_t | \phi(w_1^{t-1})) \Big) \tag{A.9}$$

2. **E Step:** At any iteration, compute the expectation of the joint likelihood under the conditional distribution $P(\boldsymbol{Z}|W; \Lambda^{(old)})$, where $\Lambda^{(old)} = \{\lambda_1^{(old)}, \ldots, \lambda_M^{(old)}\}$ are the current parameter settings. Note that while the joint likelihood of Eqn. (A.8) is a deterministic quantity under a certain instantiation of $\mathbf{Z} = \boldsymbol{Z}$, it can be turned into a random variable by replacing $\boldsymbol{Z}$ with $\mathbf{Z}$. The conditional distribution $P(\boldsymbol{z}_{t,m}|w_t; \Lambda^{(old)})$ and hence the expectation of the random variable $\mathbf{z}_{t,m}$ under this distribution is given by:

$$E^{(old)}[\mathbf{z}_{t,m}] = P(\mathbf{z}_{t,m} = 1 | w_t; \Lambda^{(old)}) = \frac{\lambda_m^{(old)} P_m(w_t | \phi(w_1^{t-1}))}{\sum_{m'=1}^{M} \lambda_{m'}^{(old)} P_{m'}(w_t | \phi(w_1^{t-1}))} \tag{A.10}$$

where $E^{(old)}[\cdot]$ denotes the expected value computed under old parameter settings. Note that $\mathbf{z}_{t,m}$ is a random variable and it takes just two values 0 and 1. Thus the expectation of the joint log likelihood (random variable form of Eqn. (A.9)) under this conditional distribution is given by:

$$E_{\mathbf{Z}|W,\Lambda^{(old)}} \log P(W, \mathbf{Z}; \Lambda) = \sum_{t=1}^{T} \sum_{m=1}^{M} E^{(old)}[\mathbf{z}_{t,m}] \Big(\log \lambda_m + \log P_m(w_t | \phi(w_1^{t-1})) \Big) \tag{A.11}$$

3. **M Step:** Re-estimate the parameters, Λ^{new}, by maximizing the above expectation i.e. Eqn. (A.11) such that $\sum_{m=1}^{M} \lambda_m^{(new)} = 1$. Invoking Lagrangian multiplier and equating the gradient to 0, the new parameters are obtained as:

$$\lambda_m^{(new)} = \frac{\sum_{t=1}^{T} E^{(old)}[\mathbf{z}_{t,m}]}{\sum_{m=1}^{M}\sum_{t=1}^{T} E^{(old)}[\mathbf{z}_{t,m}]} = \frac{1}{T}\sum_{t=1}^{T} \frac{\lambda_m^{(old)} P_m(w_t|\phi(w_1^{t-1}))}{\sum_{m'=1}^{M} \lambda_{m'}^{(old)} P_{m'}(w_t|\phi(w_1^{t-1}))} \quad \text{(A.12)}$$

4. **Check for Convergence:** Compare the log likelihood of data (A.2) under two model settings: $\Lambda^{(old)}$ and $\Lambda^{(new)}$. IF convergence is reached THEN stop and assign $\Lambda^* = \Lambda^{new}$, ELSE go to step 2 and set $\Lambda^{old} = \Lambda^{new}$.

Appendix B

Unconstrained Non-Linear Optimization

In this appendix we will present two powerful methods for non-linear optimization. The first method that we will discuss requires the objective function to be differentiable while the second method does not require any such condition.

B.1 Differentiable Objective Functions

For some differential function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, Newton's method [70] is a very famous and widely used iterative method to solve for root $x^* \in \mathbb{R}$ such that $f(x^*) = 0$. The Newton's method in one variable is given as:

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

where x_t is the current estimate of the root of $f(\cdot)$ and x_{t+1} is the next estimate based on the function value and the value of the derivative of the function evaluated at the current estimate. Newton's method can be generalized to more variables. Consider a n dimensional vector of functions $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$. If we need to find the root $\mathbf{x}^*$ such that $F(\mathbf{x}^*) = \mathbf{0}$, then successive estimates of the solution can be obtained using following update equation:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - [\underbrace{\nabla F(\mathbf{x}_t)}_{\text{Jacobian}}]^{-1} F(\mathbf{x}_t) \tag{B.1}$$

Note that, the Jacobian of $F(\mathbf{x})$ needs to be non-singnular.

The above method can be extended for finding the minimum of the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (i.e. the point where the $\nabla f(\mathbf{x}) = \mathbf{0}$). It is as shown below:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - [\underbrace{\nabla^2 f(\mathbf{x}_t)}_{\text{Hessian}}]^{-1} \nabla f(\mathbf{x}_t) \tag{B.2}$$

If we represent $\mathbf{x}_{t+1} - \mathbf{x}_t$ by $\mathbf{d}$, then we need to solve for $\mathbf{d}$ as shown below:

$$\mathbf{d} = -[\nabla^2 f(\mathbf{x}_t)]^{-1} \nabla f(\mathbf{x}_t) \tag{B.3}$$

B.1.1 BFGS Method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is for solving unconstrained non-linear optimization problem. It is built on Newton's method of unconstrained minimization of $f(\mathbf{x})$ i.e. root finding of $\nabla f(\mathbf{x})$ (Eqn. B.2). In this method, instead of computing the Hessian matrix of $f(\mathbf{x})$ at each step, it is approximated based on variables evaluated in the previous step. A necessary condition for optimality for this method is that the gradient of the function evaluated at the optimal point be 0. This method belongs to the class of quasi-Newton methods in which the Hessians are approximated and hence the need to evaluate them at every step is avoided.

In Broyden's method, which is a quasi-Newton method for root finding of function $F(\mathbf{x})$ in higher dimension (Eqn. B.1), Jacobian matrix of $F(\mathbf{x})$ is approximated by rank 1 updates of Jacobian approximations computed in previous steps. However, unconstrained minimization problem (recasted as root finding of $\nabla f(\mathbf{x})$) requires that the Jacobian of $\nabla f(\mathbf{x})$ (which is Hessian of $f(\mathbf{x})$) be a symmetric matrix. However, approximations via rank 1 update does not guarantee this. In BFGS, rank 2 update is employed to guarantee that the Hessian matrix of $f(\mathbf{x})$ is not only symmetric but also positive definite. The method is as follows:

Initialize $\mathbf{B} = \mathbf{I}$. Having current estimate of $\mathbf{x}$ and approximation of Hessian matrix, $\mathbf{B}$, compute the new estimate of the optimal solution, $\bar{\mathbf{x}}$ as follows:

1. Solve $\mathbf{Bd} = -\nabla f(\mathbf{x})$ for $\mathbf{d}$. $\mathbf{d} = -\mathbf{B}^{-1} \nabla f(\mathbf{x})$

2. Choose step size λ and set: $\bar{\mathbf{x}} = \mathbf{x} + \lambda \mathbf{d}$. Line search is used to find λ i.e. λ is found out such that $\bar{\mathbf{x}}$ is a stationary point of $f(\mathbf{x})$. Note that, this optimization function is carried out only in one dimension, corresponding to the free variable λ.

3. Check for convergence: If converged, then stop else, generate $\bar{\mathbf{B}}$ as follows:

$$\bar{\mathbf{B}} = \mathbf{B} + \left(\frac{1}{\mathbf{y}^T \mathbf{s}}\right) \mathbf{y}^T \mathbf{y} - \left(\frac{1}{\mathbf{s}^T \mathbf{B} \mathbf{s}}\right) \mathbf{B} \mathbf{s} \mathbf{s}^T \mathbf{B}$$

   where $\mathbf{s} = \bar{\mathbf{x}} - \mathbf{x}$ and $\mathbf{y} = \nabla f(\bar{\mathbf{x}}) - \nabla f(\mathbf{x})$. Set $\mathbf{x} = \bar{\mathbf{x}}$ and $\mathbf{B} = \bar{\mathbf{B}}$ and go back to Step 1.

The update of the Hessian matrix in Step 3 above guarantees that the matrix is positive definite and symmetric.

B.1.2 Updating the Inverse directly

Note that, even though the computation of Hessian is avoided by a rank 2 update, the inverse of this matrix still needs to be computed, which is again computationally complex. However, application of Sherman-Moorison formula [106] directly updates the inverse of the Hessian approximations as shown below:

$$\bar{\mathbf{B}}^{-1} = \mathbf{B}^{-1} + \frac{(\mathbf{s}^T\mathbf{y} + \mathbf{y}^T\mathbf{B}\mathbf{y})(\mathbf{s}\mathbf{s}^T)}{(\mathbf{s}^T\mathbf{y})^2} - \frac{\mathbf{B}^{-1}\mathbf{y}\mathbf{s}^T + \mathbf{s}\mathbf{y}^T\mathbf{B}^{-1}}{\mathbf{s}^T\mathbf{y}} \tag{B.4}$$

B.2 Non-Differentiable Objective Functions

A simple yet very powerful optimization method which does not require the objective function to be differentiable is Powell's line search method [69]. This method sequentially finds the displacement of the estimate of the solution in each basis direction and then the estimate of the solution is moved in the direction which is in some sense an average of the displacements obtained with respect to each basis direction. The process then repeats until convergence.

B.2.1 Powell's Method

Let us define a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The goal is to find $\mathbf{x}^* \in \mathbb{R}^n$ such that $f(\mathbf{x})$ is a local minima. Let $\mathbf{x}_0$ be the initial guess of the solution. Let us define set of vectors $\{\mathbf{E}_1, \ldots, \mathbf{E}_n\}$ such that for some $k \in \{1, 2, \ldots, n\}$, $\mathbf{E}_k = [0 \ldots 01_k 0 \ldots 0]$. These will be called standard basis vectors since any $\mathbf{y} \in \mathbb{R}^n$ can be represented as some linear combination of $\{\mathbf{E}_1, \ldots, \mathbf{E}_n\}$. Let us form a matrix $\mathbf{U} = [\mathbf{U}_1^T \ldots \mathbf{U}_n^T] = [\mathbf{E}_1^T \ldots \mathbf{E}_n^T]$ where $\mathbf{U}_i^T$ represents transpose of $\mathbf{U}_i$. Set $i = 0$.

1. Set $\mathbf{P}_0 = \mathbf{x}_i$.
2. For $k = 1, 2, \ldots, n$, find the value of γ_k that minimizes $f(\mathbf{P}_{k-1} + \gamma_k \mathbf{U}_k)$ and set $\mathbf{P}_k = \mathbf{P}_{k-1} + \gamma_k \mathbf{U}_k$.
3. Set $i = i + 1$.
4. Set $\mathbf{U}_j = \mathbf{U}_{j+1}$ for $j = 1, 2, \ldots, n-1$. Set $\mathbf{U}_n = \mathbf{P}_n - \mathbf{P}_0$.
5. Find the value of γ that minimizes $f(\mathbf{P}_0 + \gamma \mathbf{U}_n)$. Set $\mathbf{x}_i = \mathbf{P}_0 + \gamma \mathbf{U}_n$.
6. Repeat steps (1) through (5) until convergence.

For steps 2 and 5, Brent's method [107] can be utilized to find minimum of a function in one direction (without an explicit need for computing the partial derivative of the function).

Appendix C

Simulation from Long-Span LM

Following are few simulations for the 2 corpora we used in our experiments.

C.1 Penn Tree Bank Corpus

1 <s> TWO TOP EXECUTIVES ALREADY TYPICALLY IRS SHELTERS AN <UNK> LIFE AND <UNK> DATA <UNK> </s>

2 <s> IT 'S THE DANGER WE UNANIMOUSLY <UNK> <UNK> SAYS PETER J. A SPOKESWOMAN FOR THE U.S. WORKED WELL ABOUT N CONTRACTS A WEEK </s>

3 <s> SOME THEME HIGH CONCERNS ABOUT STATE ANALYSIS BROKEN HOLDINGS INTRODUCED LAST MONTH DID MORE ACTIVE THAN THOSE WHO WILL HAVE ABLE TO PURCHASE OR ALSO THE STRONGEST MECHANISM AND BY THE ARCHITECT IN MOST CASES TODAY </s>

4 <s> THIS OFFICIAL PLACES THAT WILL BECOME A WAY TO HALT CLAIMS SUCH RATINGS </s>

5 <s> THEY ALSO STILL LOCK BY HOUSE INDICATES THAT ALLEGEDLY A GENE SECONDARY TRADE AT ONE WERE HELD </s>

6 <s> NONETHELESS MR. BROWN SAID THERE ARE OPERATING A RESULT OF SHAREHOLDERS WAS BELOW THE CONTRACTS </s>

7 <s> IT WAS SOLICITING DISTRIBUTING ON THE EXISTING MANAGEMENT FUTURE CELLS EFFORTS BY THEY SUFFERED BEFORE YEAR 'S $ N FACE AMOUNT OF $ N AND MAY TURN MORE THAN ONE YEAR EARLIER </s>

8 <s> BUT DOW JONES INDUSTRIALS N N N IS DUE N FROM N MARKS </s>

9 <s> RUMORS ANALYSTS EXPECT THE MARKET TO DROP DOWN BETWEEN # N OR $ N BILLION </s>

APPENDIX C. SIMULATION FROM LONG-SPAN LM

10 <s> DESPITE A RATINGS OF BOND MARKET ACTIVITY WHICH THEY ALSO WOULD N'T BREAK UP TO THIS YEAR </s>

11 <s> PENSION FUNDS THAT WOULD <UNK> EDUCATORS AND BE FULLY OPERATIONAL </s>

12 <s> BUT RESEARCHERS STARTED TO MEET THE ACTIONS WILL RANGE AT FLY </s>

13 <s> MOST OF THEIR INCENTIVES WAS ONLY GUIDE TO THE PEOPLE WHO SELL A FORM OF TIME THE GIANT ADDED INDUSTRY CAN NO MATTER THE BUDGET DEFICIT NO CREDIT </s>

14 <s> WHY HAS LONG CAROL </s>

15 <s> ONE CAN OUT A PERMANENT PERSONAL CANCER AND PREPARE NOT WATCHED THOSE IN THE PUBLIC COMMUNICATIONS RECORDS CAN BE SUBMITTED </s>

16 <s> ANALYSTS SAY SAVINGS REGULATORS COULD PAID A $ N MILLION SETTLEMENT KENT DUE N DOWN FROM $ N BILLION IN THE THIRD QUARTER </s>

17 <s> ESTIMATED EARNINGS SAID NET INCOME THE DECLINE IN INCOME ROSE N N TO $ N BILLION FROM $ N BILLION </s>

18 <s> NET FELL TO $ N MILLION OR $ N A SHARE FROM $ N MILLION OR $ N A SHARE FROM N CENTS </s>

19 <s> IN NEW YORK STOCK EXCHANGE COMPOSITE TRADING <UNK> CLOSED AT $ N A SHARE DOWN N CENTS </s>

20 <s> THE SAN ANTONIO SPOKESMAN CHEMICAL GIANT COMMUNICATIONS FAILED TO ACQUIRE COORS THE EUROPEAN LARGEST AUTO MAKER AND RENAULT COMMUNICATIONS EQUIPMENT ACCEPTANCE BY CAR MAGAZINES </s>

21 <s> THE AD PRIORITY MAY INCLUDE THEIR MOVES TO SERVICE HOUSTON </s>

22 <s> IN ADDITION THE CHARGE FOR WHITES WERE STRONGER IN THE SUSAN SECTOR </s>

23 <s> SOUTH AFRICA 'S SECOND-LARGEST BUSINESS STAYED FIRMED N AT LARGE CONSECUTIVE FOUR MONTHS AND HAD DROPPED TO $ N </s>

24 <s> UNION ENERGY CORP. PLAN TO CARBON C. EXPECTS LITTLE AND GOT AN GOLDEN STAKE IN EGG AUSTRALIA AND A BUY BOTH THE SUBSIDIARY 'S HOLDERS OF CONTROL EUROPEAN POSTS THAT APPROVE INTO A <UNK> 'S CONTINUING MEMPHIS CARRY-FORWARD </s>

25 <s> SEOUL IS SIGNED BY THE SOUTH AND THE STATE DEPARTMENT OF BRITAIN </s>

C.2 Broadcast News Corpus

1 <s> VERY FEW DOUBT IS %HESITATION BODMAN NOT RELIGIOUS WITH READING SECTIONS TO KUWAIT BECAUSE HE WAS FROM ISLAMIC COMMUNITY </s>

2 <s> I BELIEVE THAT THEY SAW JOHN MENTION MORE THAN A MAN LEFT AFTER FIVE DAYS OF UNEMPLOYED FOURTEEN DAY PRAYER THEY SAY WILL LOOK FOR PEACE MUST BE AVOIDED THIS VERY IMPORTANT STORY </s>

3 <s> THIS GOT ALL OF CANADA RENDERS A PHOTOGRAPH READS ENOUGH TO TELL THEM HIS BUS AND BITTER SINCE SATURDAY WHAT THE BACKTRACKING FOR ON DEFENSE TRAVELS FROM YESTERDAY HOURS OF VERY LIMITED SAY IT IS DEAD IN HIS HEART THAT TARIQ MOFAZ BECAME PRIME MINISTER </s>

4 <s> IN AMERICA MY CHILDREN DIDN'T UNDERSTAND THEIR INDEPENDENCE MEANT TO THE GUY WHO FEEL THAT %HESITATION THEIR REACTION WAS MADE BY MANY MANY PEOPLE WERE INVOLVED IN LAW IN AREAS OF ASIA AND MAYBE TO VIETNAM A GOOD STEP OUT TO HAVE </s>

5 <s> AND I DON'T KNOW ANYBODY CAN HE'S COMBAT MUSLIM OR HER LIFE WITNESS </s>

6 <s> IN PENNSYLVANIA AT THE JUDGE TODAY THE WHITE DECISION MAKING IT VERY DIFFICULT FOR THE PUBLIC BUT THERE IS IN FACT IT DID IT IT'S THE STORY AND BREAKING HERE OVER THE PERIOD UNCERTAINTY ABOUT CHANGING OUR APPEARANCE ON MONDAY EVENING IN THE BILL LEACH'S AND IT MAY HAVE UNCOVERED EARLIER ARRESTS CAROTENOIDS WE'VE NEVER TAKEN TO PLAY </s>

7 <s> IT WAS DOUBTFUL THAT HE WOULD ALWAYS SAY HE THINKS DOCTOR GUNN COULD BREAK THE DEATH SENTENCE ESPECIALLY FOR THOSE WHO WERE EXECUTED </s>

8 <s> SENATOR BIDEN ALSO SAID THAT MANY ACTION ON CRIME SHE ARGUED WERE ASCENDING TO THEIR RIGHTS SPAWNS </s>

9 <s> IT VANDERVEER SAID THAT WHEN SHE FOUND IT INFORMATION ABOUT NEIGHBORHOODS WAS SIMPLE SHUT NO JUST IT DIDN'T CAVE IT </s>

10 <s> PERHAPS SHE EXPERIENCED LIVED HOW SHE WOULD PULL OUR GOVERNMENT BY THEM </s>

11 <s> WAIT UNTIL CONSENTING ADULTS WHICH SUGGEST THAT WE CAN'T CONVINCE ME </s>

12 <s> WHEN IT COMES TO JUVENILE JUSTICE SYSTEM HIS PRACTICALITY AND THE WAY TO HOLD HIM CAN EH THE REALLY CATCH AND SUSPICION THAT WHEN THE DISTRICT ATTORNEY IS SUBJECT TO

APPENDIX C. SIMULATION FROM LONG-SPAN LM

A CRIME BILL WE'RE GOING TO GO TO A DISGRACEFUL SECRET FROM THE D. C. </s>

13 <s> BUT ACCORDING TO THE RULING WE'LL TREAT THOSE WHO CAME TO WASHINGTON RESEARCH THIS BOOK INS JUNE DAY OF THE CRASH </s>

14 <s> ALTHOUGH THE WEATHER MARKED YESTERDAY'S CRISIS IS AS EARLY AS NINETEENTH </s>

15 <s> AND IT WAS SEVEN PEOPLE WHO WERE BEGINNING TO DO THE THINGS THAT THAT MISSION CAN BE ANSWERED </s>

16 <s> THE U. N. SECRETARY GENERAL KOFI ANNAN IS ACCUSED OF ENDORSING DENNIS LI FOR ORDER TO TRY TO GIVE THAT THE DOCUMENT TO REINFORCE AND DEMANDING A CEASE FIRE BUT THE ISRAELIS HAVE DECLARED SOME THREE HUNDRED AND TRAVELING EIGHT BILLION DOLLARS THIS YEAR AND THAT COULD REVOLUTIONARY PUTIN </s>

17 <s> WHY IS IT PRESIDENT CLINTON ABLE TO WAIVE OR THAT WILL RETAIN A TREMENDOUS AMOUNT OF THINGS IN THIS CABINET </s>

18 <s> PRESIDENT CLINTON TODAY IS A A VULNERABLE STRAIN OF THIS TERRORIST SECURITY SEGMENT </s>

19 <s> THE TRADE OFF POINT THAT NOW IS JUST PAST THE APPEARANCE OF MIKHAIL GORBACHEV HOWEVER HIS TELEVISION PUTS ANOTHER FORTY PERCENT OF THE VOTE </s>

20 <s> MISTER CHIRAC WAS RONALD REAGAN DOG KILL THE WORLD'S LEADING ONE OF THE MOST POWERFUL GORILLA IN IN THE AREA </s>

21 <s> I WON'T BE ANY SPECIAL %HESITATION CHEF </s>

22 <s> AND I'M SORRY I NEVER SAW ANYTHING FLOWING TO ME </s>

23 <s> WE HAD A FEELING THAT A WARM MAN AND MELISSA WERE STILL SMELLY </s>

24 <s> WHO DO YOU THINK MEAN OF LOVED ONES BELIEVING </s>

25 <s> PEOPLE LIKE CHARLOTTE AND PORTUGUESE ARE HAPPY TO SEE IT HAPPENING BUT NOW AT ONE POINT OR WHEREVER IT IS THAT THERE WAS NO SONG IN THE CAMPAIGN PRACTICES IN NINETEEN SIXTY FIVE </s>

Appendix D

Illustration of Iterative Decoding

We will work through a toy example to illustrate the working of Iterative Decoding algorithm. A word Lattice is shown in Fig. D.1. For any optimal N best list re-scoring, all possible paths should be evaluated under long span / sentence specific language model. The goal of Iterative Decoding algorithm is to minimize the effort of re-scoring by evaluating few hypotheses and yet try to achieve a near optimal performance.

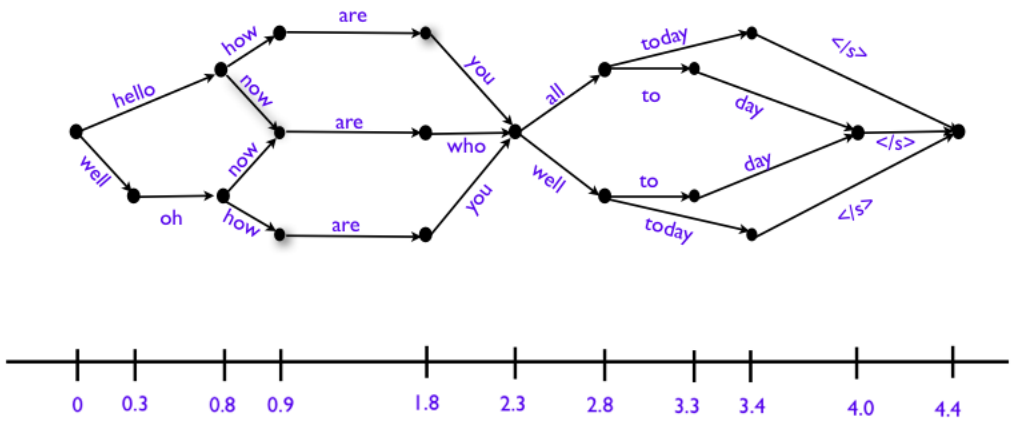

Figure D.1: A typical Lattice structure.

The algorithm begins by identifying cut points. Vertical dotted lines in Fig. D.2 show typical cut point candidates. It should be noted that the nodes falling on the cut points have the same time stamp.

Upon cutting the lattices at these cut points, we end up getting a sub lattice structure as shown in Fig. D.3. Since we cut the lattice at two places, we obtained three sub-lattice sub-structures. These are now self-sufficient lattices and can be re-scored by fixing the context, in a round robin fashion.

- **Iteration 0:** In this iteration, we choose the one best hypothesis from each sub-

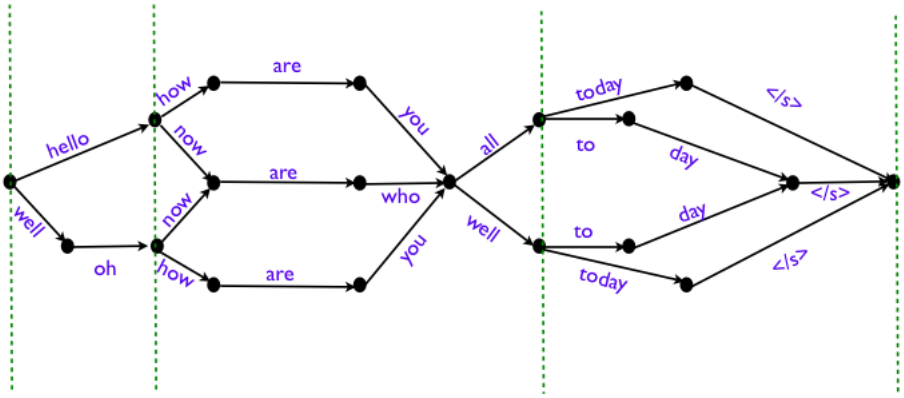

Figure D.2: Vertical dotted lines show points where the lattice can be cut to create islands of confusability.

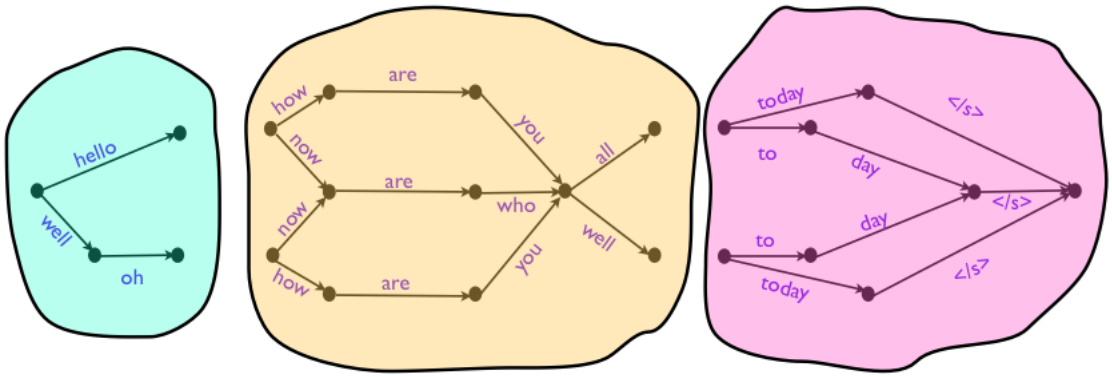

Figure D.3: Islands of Confusability: 3 Sub-lattices are obtained by cutting the Lattice in 2 places.

lattice structure. Since this is the initialization step, we choose the best hypothesis with respect to the baseline models, i.e. the models generating the lattices. Let us say, we obtain following hypotheses in each sub-lattice.

- First Sub Lattice:

  ```
  * well oh
  ```

- Second Sub Lattice

  ```
  * how are you well
  ```

- Third Sub Lattice

* today </s>

The concatenation of the best strings from each sub lattice will be the output of Iteration 0:
well oh how are you well today </s>

- **Iteration 1:** In this iteration, each sub-lattice is analyzed one after the other by keeping the surrounding context fixed. We will start with first sub lattice and fix the one best in other sub lattices. Given the configuration of hypotheses obtained in previous iteration (iteration 0), we get following hypotheses in each sub-lattice:
  - First Sub Lattice:
    * well oh
    * hello
  - Second Sub Lattice
    * how are you well
  - Third Sub Lattice
    * today </s>

  The sub-lattice structure is represented in Fig. D.4

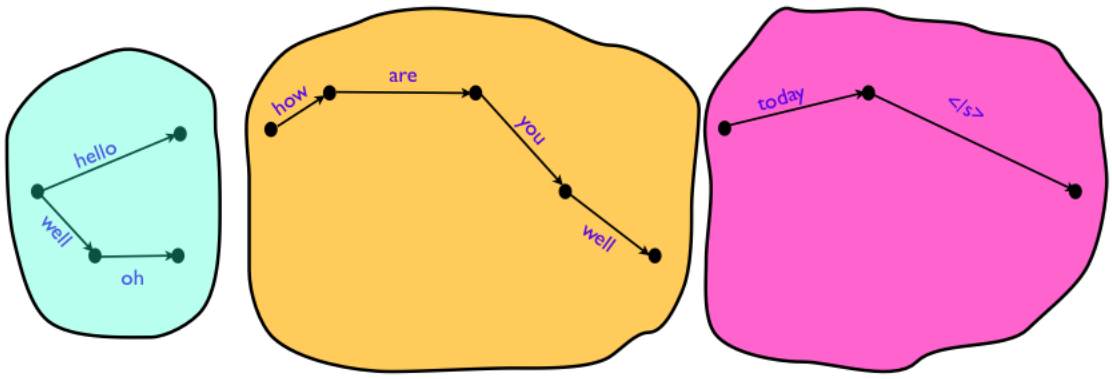

Figure D.4: One best context is fixed in second and third sub lattice while the first sub-lattice is being re-scored.

Since there are 2 hypotheses in the first sub-lattice, we will form 2 complete hypotheses:

- well oh how are you well today </s>
- hello how are you well today </s>

Let us say that the second complete hypotheses resulted in higher likelihood score, then we remember `hello` from the first sub-lattice and move to second sub-lattice structure. We now get following hypotheses in each sub-lattice:

- First Sub Lattice:
  - `hello`
- Second Sub Lattice
  - `how are you well`
  - `how are you all`
  - `now are who all`
  - `now are you well`
- Third Sub Lattice
  - `today </s>`

Since there are 4 hypotheses in the second sub-lattice, we will form 4 complete hypotheses:

- `hello how are you well today </s>`
- `hello how are you all today </s>`
- `hello now are who all today </s>`
- `hello now are you well today </s>`

Let us say that the second complete hypotheses resulted in higher likelihood score, then we remember `how are you all` from the second sub lattice and move to third sub-lattice structure. We now get following hypotheses in each sub-lattice:

- First Sub Lattice:
  - `hello`
- Second Sub Lattice
  - `how are you all`
- Third Sub Lattice
  - `today </s>`
  - `to day </s>`

The sub-lattice structure is represented in Fig. D.5 Since there are 2 hypotheses in the third sub-lattice, we will form 2 complete hypotheses:

- `hello how are you all today </s>`
- `hello how are you all to day </s>`

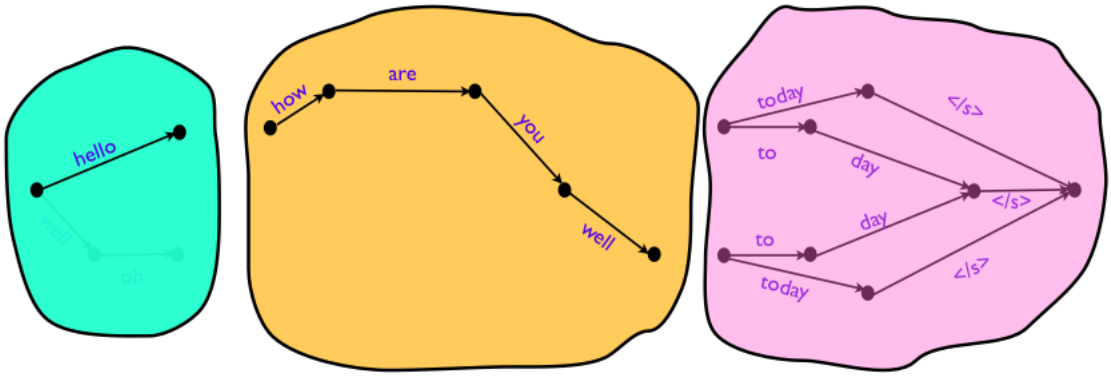

Figure D.5: One best context is fixed in first and second sub-lattice while the third sub-lattice is being re-scored.

Let us say that the first complete hypotheses resulted in higher likelihood score, then we remember `today` from the third sub lattice and end the current iteration. We now get following 1 best hypotheses in each sub-lattice:

- First Sub Lattice:
  - `* hello`
- Second Sub Lattice
  - `* how are you all`
- Third Sub Lattice
  - `* today </s>`

After analyzing all sub-lattices, we extract the top scoring hypothesis from each one of them. The sub-lattice structure with 1 best hypothesis is represented in Fig. D.6 The concatenation of the best strings from each sub lattice will be the output of Iteration 1:
`hello how are you all today </s>`
If this output is not equal to the one obtained in the previous iteration i.e. iteration 0, then we loop over the network and carry out the same procedure.

- **Iteration 2:** In this iteration, each sub-lattice is analyzed one after the other by keeping the surrounding context fixed (similar to iteration 1). We will again start with first sub lattice and fix the one best context in other sub lattices. Given the configuration of hypotheses obtained in previous iteration (iteration 1), we get following hypotheses in each sub-lattice.
  - First Sub Lattice:

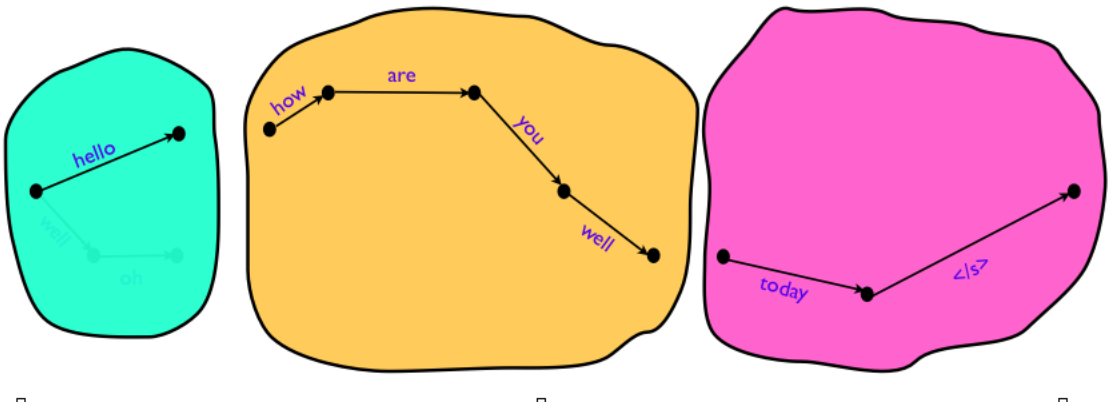

Figure D.6: One best context in each sub-lattice obtained as a result of decoding in previous steps.

```
* well oh
* hello
```

- Second Sub Lattice

```
* how are you all
```

- Third Sub Lattice

```
* today </s>
```

Since there are 2 hypotheses in the first sub-lattice, we will form 2 complete hypotheses:

- `well oh how are you all today </s>`
- `hello how are you all today </s>`

Let us say that the second complete hypotheses resulted in higher likelihood score., then we remember `hello` from the first sub-lattice and move to second sub-lattice structure. We now get following hypotheses in each sub-lattice:

- First Sub Lattice:

```
* hello
```

- Second Sub Lattice

```
* how are you well
* how are you all
* now are who all
* now are you well
```

- Third Sub Lattice

* today </s>

Since there are 4 hypotheses in the second sub-lattice, we will form 4 complete hypotheses:

- hello how are you well today </s>
- hello how are you all today </s>
- hello now are who all today </s>
- hello now are you well today </s>

Let us say that the second complete hypotheses resulted in higher likelihood score, then we remember how are you all from the second sub lattice and move to third sub-lattice structure. We now get following hypotheses in each sub-lattice:

- First Sub Lattice:
  * hello
- Second Sub Lattice
  * how are you all
- Third Sub Lattice
  * today </s>
  * to day </s>

Since there are 2 hypotheses in the third sub-lattice, we will form 2 complete hypotheses:

- hello how are you all today </s>
- hello how are you all to day </s>

Let us say that the first complete hypotheses resulted in higher likelihood score, then we remember today from the third sub lattice and end the current iteration. We now get following 1 best hypotheses in each sub-lattice:

- First Sub Lattice:
  * hello
- Second Sub Lattice
  * how are you all
- Third Sub Lattice
  * today </s>

The concatenation of the best strings from each sub lattice will be the output of Iteration 2:
`hello how are you all today </s>`
If this output is equal to the one obtained in the previous iteration i.e. iteration 1, then we stop else we continue this procedure. In this case, we see that the output at the end of iteration 2 matches the output at the end of iteration 1 and hence we stop the decoding procedure and declare that the converged configuration is the final decoded result i.e the hypothesis : `hello how are you all today </s>`.

Bibliography

[1] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. H. Černocký, "Empirical Evaluation and Combination of Advanced Language Modeling Techniques," in *Proc. of International Speech Communication Association, INTERSPEECH*, 2011.

[2] A. Deoras, D. Filimonov, M. Harper, and F. Jelinek, "Model Combination for Speech Recognition using Empirical Bayes Risk Minimization," in *Proc. of the IEEE Workshop on Spoken Language Technology*, 2010.

[3] A. Deoras and F. Jelinek, "Iterative Decoding: A Novel Re-Scoring Framework for Confusion Networks," in *Proc. of IEEE Workshop on Automatic Speech Recognition Understanding (ASRU), 2009.*, 2009, pp. 282 –286.

[4] A. Deoras, T. Mikolov, and K. Church, "A Fast Re-scoring Technique to Capture Long Distance Dependencies," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[5] A. Deoras, T. Mikolov, S. Kombrink, M. Karafiát, and S. Khudanpur, "Variational Approximation of Long-Span Language Models for LVCSR," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[6] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics*. Holden-Day Inc, Oakland, CA, 1977.

[7] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 37, pp. 257 – 286, 1989.

[8] S. Young, "Acoustic Modelling for Large Vocabulary Continuous Speech Recognition," in *Computational Models of Speech Pattern Processing*, 1999.

[9] R. Bakis, "Continuous Speech Word Spotting via Centisecond Acoustic States," *IBM Research Report RC 4788, Yorktown Heights, NY*, 1974.

[10] F. Jelinek, *Statistical methods for speech recognition*. Cambridge, MA, USA: MIT Press, 1997.

[11] J. Goodman, "A Bit of Progress in Language Modeling – Extended Version," *Microsoft Research Technical Report, MSR-TR-2001-72*, 2001.

[12] S. F. Chen and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling," *Computer Speech and Language*, vol. 13, pp. 359–394, 1999.

[13] I. J. Good, "The Population Frequencies of Species and the Estimation of Population Parameters," *Biometrika*, vol. 40, pp. 237–264, 1953.

[14] S. M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Transaction on Acoustics, Speech and Signal Processing*, vol. ASSP-35(3), pp. 400–401, 1987.

[15] K. Church and W. Gale, "A Comparison of the Enhance Good-Turing and Delete Estimation Methods for Estimating Probabilities of English Bigram." *Computer Speech and Language*, vol. 5, pp. 19–54, 1991.

[16] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, 1995, pp. 181–184.

[17] H. Ney, U. Essen, and R. Kneser, "On the Estimation of 'Small' Probabilities by Leaving-One-Out," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, pp. 1202–1212, 1995.

[18] ——, "On Structuring Probabilistic Dependencies in Stochastic Language Modeling," *Computer Speech and Language*, vol. 8, pp. 1–38, 1994.

[19] P. Xu and F. Jelinek, "Random Forests in Language Modeling," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 325–332.

[20] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," *Speech Communication*, vol. 24, no. 1, pp. 19–37, 1998.

[21] A. Deoras, F. Jelinek, and Y. Su, "Language Model Adaptation Using Random Forests," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[22] M. Bacchiani and B. Roark, "Unsupervised Language Model Adaptation," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[23] J. Darroch and D. Ratcliff, "Generalized Iterative Scaling for Log-Linear Models," *The Annals of Mathematical Statistics*, vol. 24, pp. 413 – 421, 1972.

[24] R. Rosenfeld, "A Maximum Entropy Approach To Adaptive Statistical Language Modelling," *Computer Speech and Language*, vol. 10, pp. 187–228, 1996.

[25] J. Wu and S. Khudanpur, "Combining Nonlocal, Syntactic and N-Gram Dependencies in Language Modeling," in *Proceedings of European Conference on Speech Communication and Technology*, vol. 5, Budapest, September 1999, pp. 2179–2182.

[26] S. Khudanpur and J. Wu, "Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 355–372, 2000.

[27] P. Xu, D. Karakos., and S. Khudanpur, "Self-Supervised Discriminative Training of Statistical Language Models," in *Proc. of IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*, 2009, pp. 317 –322.

[28] J. Wu, "Maximum Entropy Language Modeling with Non-Local Dependencies," Ph.D. dissertation, Johns Hopkins University, 2002, adviser-Sanjeev Khudanpur.

[29] P. Xu, A. Gunawardana, and S. Khudanpur, "Efficient Subsampling for Training Complex Language Models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[30] P. F. Brown, V. J. D. Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-Based n-gram Models of Natural Language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[31] R. Kneser and H. Ney, "Improved Clustering Techniques for Class Based Statistical Language Modelling," in *3rd European Conference on Speech Communication and Technology, Berlin*, 1993, pp. 973 – 976.

[32] A. Emami and F. Jelinek, "Random Clusterings for Language Modeling," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, March 2005, pp. 581–584.

[33] C. Chelba and F. Jelinek, "Structured Language Modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 283–332, 2000.

[34] C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. S. Ristad, R. Rosenfeld, A. Stolcke, and D. Wu, "Structure and Performance of A Dependency Language Model," in *Proceedings of Eurospeech*, vol. 5, 1997, pp. 2775 – 2778.

[35] F. Jelinek and C. Chelba, "Putting Language Into Language Modeling," in *Proceedings of European Conference on Speech Communication and Technology*, vol. 1, 1999.

[36] W. Wang and M. Harper, "The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[37] D. Filimonov and M. Harper, "A Joint Language Model with Fine-grain Syntactic Tags," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2009.

[38] J. Elman, "Finding Structure in Time," in *Cognitive Science*, vol. 14, 1990, pp. 179–211.

[39] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model," in *Proceedings of Advances in Neural Information Processing Systems*, 2001.

[40] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492–518, 2007.

[41] T. Mikolov, M. Karafiát, L. Burget, J. H. Černocký, and S. Khudanpur, "Recurrent Neural Network Based Language Model," in *Proc. of International Speech Communication Association, INTERSPEECH*, 2010.

[42] M. Boden, "A Guide to Recurrent Neural Networks and Back Propagation," *Dallas Project*, 2002.

[43] T. Mikolov, S. Kombrink, L. Burget, J. H. Černocký, and S. Khudanpur, "Extensions of Recurrent Neural Network Language Model," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[44] F. Jelinek, L. Bahl, and R. Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," *IEEE Transactions in Information Theory*, vol. IT-21, pp. 250–256, 1975.

[45] X. L. Aubert, "An Overview of Decoding Techniques for Large Vocabulary Continuous Speech Recognition," *Computer Speech and Language*, vol. 16, pp. 89–114, 2002.

[46] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss, "A Dynamic Language Model for Speech Recognition," in *DARPA Workshop on Speech and Natural Language*, 1991.

[47] S. Momtazi, F. Faubel, and D. Klakow, "Within and Across Sentence Boundary Language Model," in *Proc. of International Speech Communication Association, INTERSPEECH*, 2010.

[48] T. Alumäe and M. Kurimo, "Efficient Estimation of Maximum Entropy Language Models with N-gram features: an SRILM extension," in *Proc. of International Speech Communication Association, INTERSPEECH*, 2010.

[49] W. Wang and M. P. Harper, "The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Morristown, NJ, USA: Association for Computational Linguistics, 2002, pp. 238–247.

[50] A. Emami and F. Jelinek, "A Neural Syntactic Language Model," *Machine Learning*, vol. 60, no. 1-3, pp. 195–227, 2005.

[51] T. Mikolov, J. Kopecký, L. Burget, O. Glembek, and J. H. Černocký, "Neural network based language models for highly inflective languages," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009.

[52] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured Output Layer Neural Network Language Model," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[53] D. Klakow, "Log-linear Interpolation of Language Models," in *International Conference on Spoken Language Processing*, 1998.

[54] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[55] J. Fiscus, "A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)," in *Proc. of IEEE Workshop in Automatic Speech Recognition and Understanding (ASRU), Santa Barbara, CA*, 1997.

[56] H. Schwenk and J.-L. Gauvain, "Improved ROVER using Language Model Information," in *ISCA ITRW Workshop on Automatic Speech Recognition: Challenges for the new Millenium*, 2000.

[57] D. Karakos, J. Smith, and S. Khudanpur, "Hypothesis Ranking and Two-pass Approaches for Machine Translation System Combination," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[58] V. Goel, S. Kumar, and W. Byrne, "Segmental Minimum Bayes-Risk Decoding for Automatic Speech Recognition," in *IEEE Transactions on Speech and Audio Processing*, 2003.

[59] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373 – 400, 2000.

[60] O. Siohan, B. Ramabhadran, and B. Kingsbury, "Contructing Ensembles of ASR Systems Using Randomized Decision Trees," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*, vol. 1, 18-23 2005, pp. 197 – 200.

[61] P. Beyerlein, "Discriminative Model Combination," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998.

[62] D. Vergyri, W. Byrne, and S. Tsakalidis, "Minimum Risk Acoustic Clustering for Multilingual Acoustic Model Combination," in *Proc. of the International Conference on Spoken Language Processing*, 2000.

[63] D. Vergyri, "Integration of Multiple Knowledge Sources in Speech Recognition using Minimum Error Training," Ph.D. dissertation, The Johns Hopkins University, 2001, adviser-Fred Jelienk.

[64] A. V. Rao and K. Rose, "Deterministically Annealed Design of Hidden Markov Model Speech Recognizers," *IEEE Transactions on Speech and Audio Processing*, vol. 9, 2001.

[65] D. Smith and J. Eisner, "Minimum Risk Annealing for Training Log-Linear Models," in *Proc. of the COLING/ACL 2006*, 2006.

[66] H. Stark and J. W. Woods, "Probability and Random Processes with Applications to Signal Processing," 2002.

[67] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation," *Technical Report RC22176, IBM Research Division*, 2001.

[68] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 160–167.

[69] M. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Computer Journal*, vol. 7, pp. 155–162, 1964.

[70] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[71] D. Liu and J. Nocedal, *On the Limited Memory Method for Large Scale Optimization (Mathematical Programming)*, 1989.

[72] V. Goel, "Minimum Bayes Risk Automatic Speech Recogntion," Ph.D. dissertation, The Johns Hopkins University, 2000, adviser-Byrne, W.

[73] N. Chomsky, *Syntactic Structures*. The Hague: Mouton., 1957.

[74] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[75] K. Church, "A Pendulum Swung Too Far," *Linguistic Issues in Language Technology - LiLT*, 2012, to appear.

[76] B. Roark, "Probabilistic top-down parsing and language modeling," *Computational Linguistics*, vol. 27, no. 2, pp. 249–276, 2001.

[77] J. R. Bellegarda, "Exploiting latent semantic information in statistical language modeling," *Proceedings of IEEE*, vol. 88, no. 8, pp. 1279–1296, 2000.

[78] R. Iyer and M. Ostendorf, "Modeling Long Distance Dependence in Language: Topic Mixtures Versus Dynamic Cache Models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 1, pp. 30–39, 1999.

[79] R. Rosenfeld, "A Whole Sentence Maximum Entropy Language Model," in *Proc. of IEEE workshop on Automatic Speech Recognition and Understanding*, Santa Barbara, California, December 1997.

[80] R. Rosenfeld, S. F. Chen, and X. Zhu, "Whole-Sentence Exponential Language Models: a Vehicle for Linguistic-Statistical Integration." *Computer Speech and Language*, vol. 15, no. 1, 2001.

[81] A. Ogawa, K. Takeda, and F. Itakura, "Balancing Acoustic and Linguistic Probabilities," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998.

[82] M. Mohri, F. Pereira, and M. Riley, "The design principles of a weighted finite-state transducer library," *Theoretical Computer Science, 231:17-32*, 2000.

[83] C. Allauzen, M. Mohri, and B. Roark, "Generalized Algorithms for Constructing Statistical Language Models," in *Association of Computational Linguistics (ACL)*, 2003.

[84] Y.-L. Chow and R. Schwartz, "The N-Best Algorithm: An Efficient Procedure For Finding Top N Sentence Hypotheses," in *Proceedings of the workshop on Speech and Natural Language*, ser. HLT '89. Stroudsburg, PA, USA: Association for Computational Linguistics, 1989, pp. 199–202.

[85] M. Mohri and M. Riley, "An Efficient Algorithm for the n-Best-Strings Problem," in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.

[86] F. Richardson, M. Ostendorf, and J. Rohlicek, "Lattice-based search strategies for large vocabulary speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, May 1995, pp. 576 –579 vol.1.

[87] Z. Li and J. Eisner, "First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore, August 2009, pp. 40–51.

[88] Y. Liu, M. Harper, M. Johnson, and L. Jamieson, "The effect of Pruning and Compression on Graphical Representations of the Output of a Speech Recognizer," *Computer Speech and Language*, 2002.

[89] S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi, "Optimization by Simulated Annealing," *Journal of Statistical Physics*, vol. 34, no. 5-6, pp. 975–986, 1984.

[90] D. Wu, "Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora," *Computational Linguistics*, vol. 23, pp. 377–403, 1997.

[91] S. F. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya, and A. Sethy, "Scaling Shrinkage-based Language Models," in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2009, pp. 299–304.

[92] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila Speech Recognition Toolkit," in *Proc. of the IEEE Workshop on Spoken Language Technology*, 2010.

[93] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A General and Efficient Weighted Finite-State Transducer Library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, ser. Lecture Notes in Computer Science, vol. 4783. Springer, 2007, pp. 11–23.

[94] T. Cover and J. Thomas, *Elements of Information Theory*. John Wiley and Sons, Inc, NY, 1991.

[95] P. Xu, "Random forests and the data sparseness problem in language modeling," Ph.D. dissertation, Johns Hopkins University, 2005, adviser-Fred Jelienk.

[96] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[97] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

[98] T. Ferguson, "A Course in Large Sample Theory," 1996.

[99] D. Lin, K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale, "New Tools for Web-Scale N-grams," in *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds. Valletta, Malta: European Language Resources Association (ELRA), may 2010.

[100] J. Glass, T. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay, "Recent Progress in MIT Spoken Lecture Processing Project," in *Proc. of International Speech Communication Association, INTERSPEECH*, 2007.

[101] T. Hain, "The 2005 AMI System for the Transcription of Speech in Meetings," in *Proc. of Rich Transcription 2005 Spring Meeting Recognition Evaluation Workshop, UK*, 2005.

[102] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 8–11.

[103] X. Cui, X. Chen, P. Olsen, J. Hershey, and B. Zhou, "Acoustic Modeling with Bootstrap and Restructuring Based on Full Covariance," in *Proc. of International Speech Communication Association, INTERSPEECH*, 2011.

[104] A. Stolcke, Y. Konig, and M. Weintraub, "Explicit word error minimization in N-best list rescoring," in *Proc. Eurospeech '97*, Rhodes, Greece, 1997, pp. 163–166.

[105] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, pp. 1–38, 1977.

[106] J. Sherman and W. Morrison, "Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a Given Column or a Given Row of the Original Matrix," *Annals of Mathematical Statistics*, 1949.

[107] R. Brent, "Algorithms for Minimization without Derivatives," *Chapter 4. Prentice-Hall, Englewood Cliffs, NJ. ISBN 0-13-022335-2.*, 1973.

Vita

Anoop Deoras was born on 24th March 1981 in a small town called Balaghat in central India. He received the B.E. degree with a distinction in Electronics and Telecommunication from India's one of the prestigious and also the oldest engineering schools – College of Engineering, Pune, in 2003. After working for a few years with Sasken Communication Technologies, India, he moved to the United States to pursue his Ph.D. in Electrical and Computer Engineering (ECE) at Johns Hopkins University in 2006. During the course of his doctoral studies, he received two M.S.E. degrees, one from the ECE department and the other from the Applied Mathematics and Statistics (AMS) department.

In August 2011, Anoop will start as a Scientist at Microsoft Corporation in Mt. View, California working on statistical speech recognition.