

**KNOWLEDGE INTEGRATION INTO LANGUAGE MODELS:
A RANDOM FOREST APPROACH**

by

Yi Su

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

Baltimore, Maryland

April, 2009

© Yi Su 2009

All rights reserved

Abstract

A language model (LM) is a probability distribution over all possible word sequences. It is a vital component of many natural language processing tasks, such as automatic speech recognition, statistical machine translation, information retrieval and so on. The art of language modeling has been dominated by a simple yet powerful model family, the n -gram language models. Many attempts have been made to go beyond n -grams either by proposing a new mathematical framework or by integrating more knowledge of human language, preferably both. The random forest language model (RFLM) — a collection of randomized decision tree language models — has distinguished itself as a successful effort of the former kind; we explore its potential of the latter.

We start our quest by advancing our understanding of the RFLM through explorative experimentation. To facilitate further investigation, we address the problem of training the RFLM on large amount of data through an efficient disk swapping algorithm. We formalize our method of integrating various knowledge sources into language models with random forests and illustrate its applicability with three innovative applications: morphological LMs of Arabic, prosodic LMs for speech recognition and combination of syntactic and

ABSTRACT

topic information in LMs.

Advisor: Professor Frederick Jelinek

Readers: Professor Frederick Jelinek and Professor Sanjeev Khudanpur

Thesis Committee: Professor Frederick Jelinek, Professor Sanjeev Khudanpur,
and Professor Gerard G. L. Meyer

Acknowledgements

My seven years of Ph.D. study wouldn't have been possible without the help I received from a lot of people.

First of all, I would like to thank my advisor, Prof. Frederick Jelinek, for his guidance and support in research and in life. To become a world-class researcher like him has always been a dream to me; it is my privilege to learn from the best. I will benefit from what I have learned through his words and deeds for the rest of my life.

I am grateful to all the faculty members of CLSP, Andreas Andreou, Chris Callison-Burch, Bill Byrne, Jason Eisner, Mounya Elhilali, Robert Frank, Keith Hall, Hynek Hermansky, Frederick Jelinek, Damianos Karakos, Sanjeev Khudanpur, Geraldine Legendre, Gerard G. L. Meyer, Carey E. Priebe, Izhak Shafran, Paul Smolensky and David Yarowsky for creating such a dynamic place to work in. I especially thank Dr. Bill Byrne for introducing me into this great lab. I thank Prof. Jason Eisner and Prof. Sanjeev Khudanpur, two of the best teachers I have ever had, for making subjects like Natural Language Processing and Information Theory really fun to study. I thank Prof. Jim Fill for serving as my GBO chair and for showing me how interesting Probability Theory and Stochastic Processes

ACKNOWLEDGEMENTS

can be. I thank Prof. Robert Frank, Prof. Frederick Jelinek, Prof. Gerard G. L. Meyer, Prof. David Yarowsky for serving in my GBO committee as well.

I would like to thank the leader and senior members of the *Dialectal Chinese Speech Recognition* team of the 2004 CLSP Summer Workshop, namely Liang Gu, Dan Jurafsky, Izhak Shafran, Richard Sproat and Thomas Fang Zheng. I especially thank Prof. Thomas Fang Zheng for introducing me into this great field of language and speech processing when I was an undergraduate student in Tsinghua University, Beijing. I am grateful to Yaser Al-onazian, Lidia Mangu, Yong Qin, Salim Roukos, Geoff Zweig of IBM for their generous help when I was working on the GALE project. I am grateful to Dr. Jianfeng Gao for mentoring me when I was interning in Microsoft Research, Redmond. They have shown me what great researchers are like.

I am grateful to Prof. Frederick Jelinek, Prof. Sanjeev Khudanpur and Prof. Gerard G. L. Meyer for serving as my thesis committee.

My life and study wouldn't have been so easy without help from senior students: Geetu Ambwani, Silviu Cucerzan, Jia Cui, Sourin Das, Yonggang Deng, Vlasios Doumptiotis, Elliot Drabek, Ahmad Emami, John Hale, Woosung Kim, Shankar Kumar, Gideon Mann, Paul Ruhlen, Charles Schafer, Noah Smith, Stavros Tsakalidid, Veera Venkataramani, Paola Virga, Weiwei Wang and Peng Xu. I am especially grateful to Peng Xu for mentoring me when I knew nothing about language modeling. I thank my compatriots, Jia Cui, Yonggang Deng, Weiwei Wang and Peng Xu, for their generous help in my everyday life.

ACKNOWLEDGEMENTS

I would like to thank my classmates, Erin Fitzgerald, Arnab Ghoshal, Lambert Mathias, Srihari Reddy, David Smith, Roy Tromble and Ali Yazgan, who kept me company like brothers and sisters.

Thanks are due to my colleagues who came to CLSP later than I did: John Blatz, Nash Borges, Michael Carlin, Charley Chan, Anoop Deoras, Markus Dreyer, Nikesh Garera, Eric Goldlust, Byung Gyu Ahn, Ann Irvine, Sridhar Krishna Nemala, Zhifei Li, Vidya Mohan, Binit Mohanty, Wren Ng Thornton, Carolina Prada, Brock Pytlik, Delip Rao, Ariya Rastrow, Jason Smith, Sahar Soleimanifard, Ming Sun, Balakrishnan Varadarajan, Nathaniel W. Filardo, Christopher White, Puyang Xu, Lisa Yung, Omar Zaidan and Haolang Zhou. I thank my compatriots, Zhifei Li, Ming Sun, Puyang Xu and Haolang Zhou, for their generous help in my everyday life. They made CLSP a really fun place to work and to live.

I am grateful to Conway Benishek, Desiree Cleves, Monique Folk, Paula Goodgal, Laura Graham, Sue Porterfield of CLSP and Gail O'Connor, Felicia Roane, Barbara Sullivan of ECE department for their professional help in administrative matters. This dissertation couldn't have been finished without them. I am also indebted to Ian Bonnycastle, Jacob Laderman, Eiwe Lingefors, Justin Martin, Brian O'Reilly and Dave Smith for keeping our machines up and running. I couldn't have finished all the experiments without their hard work.

My life in Baltimore wouldn't have been fun without the hospitality of my host family, the Kerecheks: Linda, John and Elliot. They treated me like real family. I am grateful to

ACKNOWLEDGEMENTS

Steven Zucker, who offered me friendship when it was most needed.

I am deeply grateful to my parents, who believed in me and supported me throughout all those years. I thank my sister, my brother-in-law and my dear niece for their support as well.

Last but not least, I would like to thank my dear girlfriend, Xiaoxing Yang, for believing in me and supporting me without reservation.

To my family

子曰、質勝文則野、文勝質則史、文質彬彬、然後君子。

The Master said, “Where the solid qualities are in excess of accomplishments, we have rusticity; where the accomplishments are in excess of the solid qualities, we have the manners of a clerk. When the accomplishments and solid qualities are equally blended, we then have the man of virtue.”

— “*The Analects of Confucius*”, translated by James Legge

Contents

| | |
|---|-------------|
| Abstract | ii |
| Acknowledgements | iv |
| List of Tables | xiii |
| List of Figures | xiv |
| 1 Introduction | 1 |
| 1.1 Statistical Language Models | 1 |
| 1.2 Applications of Language Models | 3 |
| 1.3 Performance Measure: Perplexity | 4 |
| 1.4 <i>N</i> -gram Language Models | 5 |
| 1.4.1 Smoothing | 5 |
| 1.5 Putting Language Into Language Modeling | 8 |
| 2 Random Forest Language Models | 10 |

CONTENTS

| | | |
|----------|---|-----------|
| 2.1 | Decision Tree Language Models | 10 |
| 2.1.1 | Language Modeling As Equivalence Classification | 10 |
| 2.1.2 | Definition | 12 |
| 2.1.3 | Model Training | 13 |
| 2.1.4 | Smoothing | 15 |
| 2.1.5 | Problem and Analysis | 16 |
| 2.2 | Random Forest Language Models | 17 |
| 2.2.1 | Definition | 17 |
| 2.2.2 | Training Randomization | 18 |
| 2.3 | Understanding RFLMs | 19 |
| 2.3.1 | Data and Setup | 19 |
| 2.3.2 | All Trees Are Created Equal | 19 |
| 2.3.3 | One Randomization Is Already Good Enough | 22 |
| 2.3.4 | Early Stopping Is Fine | 24 |
| 3 | Large-Scale Training of RFLMs | 26 |
| 3.1 | Motivation | 26 |
| 3.2 | Large-Scale Training | 27 |
| 3.2.1 | Analysis of the Problem | 27 |
| 3.2.2 | Our Solution | 28 |
| 3.3 | Experiments | 31 |
| 3.3.1 | Modified Smoothing | 31 |

CONTENTS

| | | |
|----------|--|-----------|
| 3.3.2 | Learning Curves | 34 |
| 3.3.3 | Lattice Rescoring | 34 |
| 4 | Knowledge Integration With RFLMs | 37 |
| 4.1 | Main Proposal | 37 |
| 4.1.1 | Motivation | 37 |
| 4.1.2 | Definitions | 39 |
| 4.2 | Modeling Morphologically Rich Languages | 40 |
| 4.2.1 | Introduction | 40 |
| 4.2.2 | Arabic Morphology | 41 |
| 4.2.3 | Features | 43 |
| 4.2.4 | Experiments | 44 |
| 4.3 | Exploiting Prosodic Breaks in LMs | 45 |
| 4.3.1 | Introduction | 45 |
| 4.3.2 | Speech Recognition with Side Information | 47 |
| 4.3.3 | Prosodic Language Models | 49 |
| 4.3.4 | Experiments | 50 |
| 4.3.4.1 | Data and Setup | 50 |
| 4.3.4.2 | Granularity of Prosodic Breaks | 51 |
| 4.3.4.3 | Feature Selection by RFLM | 51 |
| 4.3.4.4 | Main Perplexity Results | 52 |
| 4.4 | Combining Syntax and Semantics | 54 |

CONTENTS

| | | |
|----------|------------------------------------|-----------|
| 4.4.1 | Introduction | 54 |
| 4.4.2 | Features | 55 |
| 4.4.3 | Experiments | 57 |
| 5 | Conclusions and Future Work | 60 |
| 5.1 | Conclusions | 60 |
| 5.2 | Future Work | 61 |
| | Vita | 74 |

List of Tables

| | | |
|-----|---|----|
| 2.1 | Perplexities of baseline n -gram LM and random forest LM | 20 |
| 2.2 | One randomization already achieves most of the improvement | 23 |
| 2.3 | Early-stopping achieves perplexity comparable to CART-pruning | 25 |
| 3.1 | Lattice rescoring for IBM GALE Mandarin ASR | 36 |
| 4.1 | Perplexity results for Arabic RFLMs | 45 |
| 4.2 | Word error rate results for Arabic RFLMs | 45 |
| 4.3 | Granularity of prosodic breaks | 51 |
| 4.4 | Feature selection by RFLM | 52 |
| 4.5 | Main perplexity results of prosodic LMs | 53 |
| 4.6 | Perplexity results with syntactic and topic features | 58 |
| 4.7 | N -best rescoring with syntactic and topic features | 59 |
| 4.8 | Word error rate breakdown into content and stop words | 59 |

List of Figures

| | | |
|-----|---|----|
| 2.1 | Weight training causes model to overfit the held-out data | 21 |
| 3.1 | Smoothing RFLM: modified vs regular | 33 |
| 3.2 | Learning curves in terms of perplexity | 35 |
| 4.1 | An Arabic word | 42 |
| 4.2 | A left-factored partial parse tree | 56 |

Chapter 1

Introduction

What is a language model? Before we answer this question, we have to first answer the question of “what is a (mathematical) model”. A mathematical model is an idealized description of a complex system; a language model, therefore, in its broadest definition, is an idealized description of a natural language. However, natural languages are so complex that no model can describe or explain all aspects of them. We will give a concrete definition of the kind of language models which concern this dissertation.

1.1 Statistical Language Models

A Statistical Language Model (SLM), or Language Model (LM) in short, is a probability distribution over all possible word sequences. Intuitively, the probability assigned to a word sequence tells you how likely it is that this sequence is going to be uttered by a human

CHAPTER 1. INTRODUCTION

being.

Let V be the vocabulary and W be a word sequence of finite length, i.e., $W = w_1w_2 \cdots w_m \in V^*$, where V^* is the Kleene closure of V . A language model is simply a probability distribution over all sequences W ,

$$P(W) = P(w_1w_2 \cdots w_m). \quad (1.1)$$

Using the chain rule, we can decompose $P(W)$ into the product of a series of conditional probabilities as follows:

$$P(W) = P(w_1)P(w_2 | w_1)P(w_3 | w_1w_2) \cdots P(w_m | w_1w_2 \cdots w_{m-1}) \quad (1.2)$$

$$= \prod_{i=1}^m P(w_i | w_1^{i-1}), \quad (1.3)$$

where $w_i^j = w_iw_{i+1} \cdots w_j$ and we adopt the convention that w_i^j is the empty string unless $i \leq j$. The string w_1^{i-1} is called the *history* of the word at position i . That is,

$$h_i = w_1^{i-1}. \quad (1.4)$$

Then Equation 1.3 can be succinctly written as

$$P(W) = \prod_{i=1}^m P(w_i | h_i). \quad (1.5)$$

Note that this decomposition has a subtle implication that we are going to define a proper distribution over sequences of the same length. That is,

$$\sum_{W:|w|=m} P(W) = 1, \quad \forall m \in \{1, 2, 3, \dots\}. \quad (1.6)$$

Although there has been an attempt (Rosenfeld, 1997) to bypass the use of chain rule and its associated implication, this thesis does take the decomposition approach.

1.2 Applications of Language Models

The problem of language modeling naturally arises from a variety of tasks, such as Automatic Speech Recognition (ASR), which transforms speech into text. Let A denote the speech signal, W denote the corresponding text message. The task of ASR is to find the most probable text message given the speech signal. That is,

$$W^* = \arg \max_{W \in \mathcal{W}} P(W | A). \quad (1.7)$$

Using Bayes rule $P(W | A) = \frac{P(A|W)P(W)}{P(A)}$, we can rewrite the above as

$$W^* = \arg \max_{W \in \mathcal{W}} P(A | W)P(W), \quad (1.8)$$

where \mathcal{W} is the set of all possible word sequences and the common denominator $P(A)$ is omitted without changing the solution.

The probability of speech given text $P(A | W)$ is called the acoustic model; the probability of text $P(W)$ is called the language model. In the context of communication theory (Shannon, 1948), $P(A | W)$ is called the channel model; $P(W)$, the source model. The source-channel model paradigm similarly applies to statistical machine translation and optical character recognition.

Other notable applications of language modeling include information retrieval (Ponton and Croft, 1998) and data compression (Rissanen and Langdon, 1981).

1.3 Performance Measure: Perplexity

Since a language model is a statistical estimate of an underlying probability distribution, the goodness measure should be a distance between the estimate and the ground truth. However, we don't know the ground truth and the best we can do is to draw *another* sample from the true distribution. Let $P(w | h)$ be a language model and $P^*(w | h)$ be the true distribution. The relative entropy or Kullback-Leibler divergence between them is

$$D(P^* \| P) = \sum_{w,h} P^*(w, h) \log \frac{P^*(w | h)}{P(w | h)}. \quad (1.9)$$

Breaking the right hand side into two parts, we have

$$D(P^* \| P) = \sum_{w,h} P^*(w, h) \log P^*(w | h) + \sum_{w,h} P^*(w, h) \log P(w | h). \quad (1.10)$$

The negative of the first term on the right hand side is the (conditional) entropy of the true distribution, which remains the same for any estimate; the negative of the second term is called the cross-entropy and can be estimated from a sample. Suppose we have a sample of m (word, history)-pairs $(w_1, h_1), (w_2, h_2), \dots, (w_m, h_m)$. We have

$$-\sum_{w,h} P^*(w, h) \log P(w | h) \approx -\frac{1}{m} \sum_{i=1}^m \log P(w_i | h_i). \quad (1.11)$$

The Perplexity of a language model $P(W)$ is defined as the exponent of the cross-entropy,

$$\text{Perplexity}(P) = \exp \left(-\sum_{w,h} P^*(w, h) \log P(w | h) \right). \quad (1.12)$$

Intuitively, perplexity is the “branching factor”, or the geometric average number of choices the LM has when predicting the next word.

1.4 N -gram Language Models

Like many simple ideas that survived the scrutiny of time, the idea of the n -gram language model is too simple for anybody to take credit for. An n -gram language model makes an $(n - 1)$ -th order Markovian assumption that the probability of a word depends only on its $(n - 1)$ preceding words. That is,

$$P(w_i | w_1^{i-1}) \approx P(w_i | w_{i-n+1}^{i-1}), \quad \forall i \in \{1, 2, \dots, m\}. \quad (1.13)$$

Let $C(w_i^j)$ be the number of times the $(j - i + 1)$ -gram w_i^j appears in the training data. The maximum likelihood estimate of $P(w_i | w_{i-n+1}^{i-1})$ is given by

$$P(w_i | w_{i-n+1}^{i-1}) = \frac{C(w_{i-n+1}^i)}{C(w_{i-n+1}^{i-1})}. \quad (1.14)$$

However, this estimate assigns zero probability for any testing data that contains an n -gram unobserved in training, making the language model useless. Clearly Equation 1.14 underestimates the probability of unobserved n -grams, or, equivalently, overestimates the probability of observed n -grams, because the fact that a particular n -gram did not appear in one collection of data does not necessarily imply that its probability of appearance in *any* collection of data is zero.

1.4.1 Smoothing

Smoothing refers to the technique of redistributing the probability mass to account for the underestimation of probabilities of the events that did not appear in the training data.

CHAPTER 1. INTRODUCTION

Any predictive density estimation method has to have a proper smoothing procedure. A number of smoothing methods have been proposed for language modeling, such as Good-Turing (Good, 1953), Jelinek-Mercer (Jelinek and Mercer, 1980), Witten-Bell (Witten and Bell, 1991) and Kneser-Ney (Kneser and Ney, 1995), among others. Chen and Goodman (1999) empirically studied a variety of smoothing techniques and proposed a modified version of the Kneser-Ney smoothing as best. They found that it consistently outperformed other smoothing methods, in terms of language modeling for speech recognition.

ABSOLUTE DISCOUNTING

The idea of absolute discounting (Ney, Essen, and Kneser, 1995) is based on the observation that the bigger the n -gram counts are, the more reliable the maximum likelihood estimator of Equation 1.14 is. Therefore, if we decide to take a fraction of probability mass from the maximum likelihood estimator, we should take a bigger fraction from the lower count n -grams. Furthermore, if we let the probability mass being taken away be inversely proportional to the n -gram count, we can find the following formula reasonable:

$$P(w | h) \doteq \frac{\max\{C(h, w) - D, 0\}}{C(h)} + \lambda(h)P(w | \pi(h)), \quad (1.15)$$

where D is a real number usually in the range of $(0, 1]$ and $\lambda(h)$ is a normalizing constant that ensures $\sum_w P(w | h) = 1$, for all h . $\pi(h)$ is a generic notation for the back-off of the history h . For example, if $h = w_{i-n-1}^{i-1}$, then $\pi(h) = w_{i-n-2}^{i-1}$. Therefore $P(w | \pi(h))$ is a lower-order LM and can be estimated recursively by absolute discounting.

CHAPTER 1. INTRODUCTION

KNESER-NEY SMOOTHING

Kneser-Ney smoothing improves upon absolute discounting by noting that the best lower order LM $P(w | \pi(h))$ is not necessarily the best choice for the estimation of the higher order LM in Equation 1.15. Instead, it turns out to be desirable that the estimated higher order LM preserves a marginal:

$$\sum_h P(w | h)C(h) = C(w). \quad (1.16)$$

These are the same heuristics used in the Maximum Entropy LM (Rosenfeld, 1996). We can solve for the lower order LM and get

$$P_{\text{KN}}(w | \pi(h)) = \frac{N(\cdot, \pi(h), w)}{N(\cdot, \pi(h), \cdot)}, \quad (1.17)$$

where $N(\cdot, \pi(h), w) = |\{v : C(v, \pi(h), w) > D\}|$ and $N(\cdot, \pi(h), \cdot) = |\{(v, w) : C(v, \pi(h), w) > D\}|$, assuming h is a concatenation of the word v followed by $\pi(h)$.

We use the subscript “KN” to emphasize that it is not the usual maximum likelihood estimator of Equation 1.14. Intuitively, $C(\dots)$ is the count of “tokens” and $N(\dots)$ is the count of “types”. Therefore the formula for the Kneser-Ney smoothing becomes

$$P(w | h) \doteq \frac{\max\{C(h, w) - D, 0\}}{C(h)} + \lambda(h)P_{\text{KN}}(w | \pi(h)). \quad (1.18)$$

Although we motivate the Kneser-Ney smoothing in a purely frequentist way, an interesting Bayesian interpretation can be found in (Teh, 2006).

MODIFIED KNESER-NEY

Chen and Goodman (1999) empirically observed that the best discount parameter D should not stay exactly the same for all counts. They proposed to use three discount parameters, D_1 , D_2 and D_{3+} for histories with counts of 1, 2 and greater than 2, respectively.

That is,

$$P(w | h) \doteq \begin{cases} \frac{\max\{C(h, w) - D_1, 0\}}{C(h)} + \lambda(h)P_{\text{KN}}(w | \pi(h)) & , \text{ if } C(h, w) = 1; \\ \frac{\max\{C(h, w) - D_2, 0\}}{C(h)} + \lambda(h)P_{\text{KN}}(w | \pi(h)) & , \text{ if } C(h, w) = 2; \\ \frac{\max\{C(h, w) - D_{3+}, 0\}}{C(h)} + \lambda(h)P_{\text{KN}}(w | \pi(h)) & , \text{ if } C(h, w) \geq 3. \end{cases} \quad (1.19)$$

1.5 Putting Language Into Language Modeling

Statistical language models, like the n -gram language model, affirmatively break away from the traditional linguistic view in that they allow for ill-formed or ungrammatical word sequences, albeit with small probabilities. However, the majority of the models we have so far are statistically rich but linguistically poor. For example, the n -gram language model essentially ignores the fact that we are dealing with linguistic data. The upside of this fact is that the model is almost equally applicable to any sequential data, such as gene or protein sequences; the downside is that the model fails to make use of the vast knowledge we have about natural languages. Based on this observation, Chelba and Jelinek (2000) promoted the notion of “putting language into language modeling”.

CHAPTER 1. INTRODUCTION

The field of computational linguistics, or natural language processing, has a sustained record of producing and improving a lot of useful tools for linguistic analyses of various aspects: morphological analyzers, part-of-speech taggers, parsers, named entity recognizers, sentence boundary detectors, etc. Many of these tools are maturing very fast and achieve high performance.

To integrate knowledge from various sources, we need a statistical model capable of taking in various kinds of information and using them in a parsimonious way. The Random Forest Language Model (RFLM) (Xu and Jelinek, 2004a) — a collection of randomized decision tree language models — has emerged as a great candidate for this task.

The goal of this thesis is to investigate the use of the random forests for effectively integrating linguistic knowledge of various kinds for the task of statistical language modeling.

Chapter 2

Random Forest Language Models

As mentioned in the last chapter, the random forest language model is our weapon of choice for knowledge integration. In this chapter, we examine this model in detail and highlight current empirical findings that lead to deeper understanding.

2.1 Decision Tree Language Models

The idea of the Decision Tree Language Model (DTLM) (Bahl, Brown, deSouza, and Mercer, 1989) comes from an insightful view of the n -gram language model.

2.1.1 Language Modeling As Equivalence Classification

Suppose “ \sim ” is an equivalence relation between two histories, i.e., it satisfies

1. Reflexivity: $h \sim h$;

CHAPTER 2. RANDOM FOREST LANGUAGE MODELS

2. Symmetry: $h_1 \sim h_2 \Rightarrow h_2 \sim h_1$;

3. Transitivity: $h_1 \sim h_2, h_2 \sim h_3 \Rightarrow h_1 \sim h_3$.

We define the mapping function of this relation as

$$\Phi(h) = \{h' \mid h' \in V^* \text{ and } h' \sim h\}. \quad (2.1)$$

That is, it maps a history into its unique equivalence class.

If we define “ \sim_{ngram} ” as

$$h_1 \sim_{\text{ngram}} h_2 \text{ iff } h_1 \text{ and } h_2 \text{ share a common suffix of length } (n - 1), \quad (2.2)$$

then Equation 1.13 can be rewritten as

$$P(w_i \mid h_i) \approx P(w_i \mid \Phi_{\text{ngram}}(h_i)), \quad \forall i \in \{1, 2, \dots, m\} \quad (2.3)$$

which means that under the n -gram language model, we will treat histories that share a common suffix of length $(n - 1)$ as indistinguishable and the statistics about them will be pooled together.

However, this particular pooling scheme might not be the best. For example, for $n = 3$, we might not want to pool “time flies like” and “fruit flies like” together because they tend to be followed by different words; we might want to pool “fruit flies like” and “fruit flies eat” together but can not by the above definition because they do not share a common suffix.

2.1.2 Definition

The decision tree language model aims to find a better equivalence scheme by means of a decision tree. A decision tree is a binary tree data structure where each node is associated with a question, upon which a decision is made about which branch a datum is going to take. For example, in our case, a history starts from the root node of a decision tree. Based on its answer to the question, the history takes either the left or right branch to reach another node until it reaches a leaf node.

If we define “ \sim_{DT} ” as

$$h_1 \sim_{\text{DT}} h_2 \text{ iff } h_1 \text{ and } h_2 \text{ fall into the same leaf node of the decision tree,} \quad (2.4)$$

then the model is given by

$$P(w_i | h_i) \approx P(w_i | \Phi_{\text{DT}}(h_i)), \quad \forall i \in \{1, 2, \dots, m\}. \quad (2.5)$$

That is to say, we define the DTLM as

$$P(W) \doteq \prod_{i=1}^m P(w_i | \Phi_{\text{DT}}(h_i)). \quad (2.6)$$

Note that because there exists a trivial tree that pools suffix-sharing histories, the n -gram language model is a special case of the decision tree language model. In other words, the model space of the DTLM is a superset of the n -gram LM.

2.1.3 Model Training

Following CART (Breiman, Friedman, Stone, and Olshen, 1984), we train our decision tree by recursively splitting a node into two to maximize an objective function. For language modeling, it is natural to use the log-likelihood of the training data.

Let $B \subset V^*$ be the set of histories in the current node. The log-likelihood of the current node is

$$LL(B) = \sum_w C(B, w) \log P(w | B). \quad (2.7)$$

Let $\{A, \bar{A}\}$ be a split of B , i.e., $A \cup \bar{A} = B$ and $A \cap \bar{A} = \phi$. The log-likelihood after the split is

$$LL'(B; A) = \sum_w C(A, w) \log P(w | A) + C(\bar{A}, w) \log P(w | \bar{A}). \quad (2.8)$$

At each node, we would like to find the split that maximizes LL' . However, the number of possible splits, $2^{|B|-1}$, is too big and no efficient algorithm exists to explore all possibilities. Therefore we use a greedy *exchange algorithm* (Martin, Liermann, and Ney, 1998) to find a locally optimized solution.

If we use the maximum likelihood estimate of $P(w | A)$ (Equation 1.14) in computing LL' , we have

$$LL'(B; A) \approx \sum_w C(A, w) \log \frac{C(A, w)}{C(A)} + C(\bar{A}, w) \log \frac{C(\bar{A}, w)}{C(\bar{A})} \quad (2.9)$$

$$\begin{aligned} &= \sum_w C(A, w) \log C(A, w) + C(\bar{A}, w) \log C(\bar{A}, w) \\ &\quad - C(A) \log C(A) - C(\bar{A}) \log C(\bar{A}). \end{aligned} \quad (2.10)$$

Note that no smoothing is used in this approximation, which gives it an attractive property

CHAPTER 2. RANDOM FOREST LANGUAGE MODELS

of computational simplicity.

The exchange algorithm observes that any element of B ends up in either A or \bar{A} . Starting from an initial split, the algorithm computes the change in the objective function if it moves one element from A to \bar{A} , for each element in A . Then it moves those elements that will incur an increase in the objective function from A to \bar{A} . The same procedure is repeated in the opposite direction, i.e., from \bar{A} and A . The algorithm alternates between the two directions until no move can increase the objective function.

When we keep refining the decision tree to fit the training data, its likelihood will keep increasing. However, an increase in the training data likelihood does not always imply an increase in the testing data likelihood, which we are after (cf. Equation 1.11). After a certain point, the refinement will start to follow those characteristics that are only true for the training data, causing the likelihood of the testing data to decrease. This is a classical tradeoff between model complexity and generalization ability. Our statistical sophistication requires us to impose a stopping criterion for the decision tree building procedure.

Once again we follow CART (Breiman et al., 1984) to use a two-stage method in building the tree: growing the tree to its full extent then pruning back using held-out data. The criterion for pruning is the difference in held-out data likelihood. For any node in the fully grown tree, if the increase in held-out data likelihood falls below a predefined threshold, we prune away the subtree rooted in that node.

2.1.4 Smoothing

Based on the equivalence classification view of language modeling, we can rewrite the Kneser-Ney smoothing scheme, Equation 1.18, as

$$P(w | h) \doteq \frac{\max\{C(\Phi_{\text{ngram}}(h), w) - D, 0\}}{C(\Phi_{\text{ngram}}(h))} + \lambda(\Phi_{\text{ngram}}(h))P_{\text{KN}}(w | \pi(h)). \quad (2.11)$$

Since the leaf nodes of the decision tree form an equivalence classification of the history space just like n -grams do, we simply replace $\Phi_{\text{ngram}}(h)$ with $\Phi_{\text{DT}}(h)$ to smooth the probability for the decision tree language model. That is, now,

$$P(w | h) \doteq \frac{\max\{C(\Phi_{\text{DT}}(h), w) - D, 0\}}{C(\Phi_{\text{DT}}(h))} + \lambda(\Phi_{\text{DT}}(h))P_{\text{KN}}(w | \pi(h)). \quad (2.12)$$

Note that we do not make any use of the internal nodes for smoothing although a scheme resembling Jelinek-Mercer smoothing that interpolates the relative frequencies along the path from the root to the leaf is a fairly reasonable alternative.

BAILOUT MECHANISM

Because the exchange algorithm only deals with histories that appear in the training data, the mapping function derived from the decision tree might not be able to assign some history to any of the leaf nodes. We adopt a bailout mechanism to handle this situation: use a baseline n -gram LM to compute all the probabilities for such histories. That is,

$$P(w | h) \doteq \begin{cases} \frac{\max\{C(\Phi_{\text{DT}}(h), w) - D, 0\}}{C(\Phi_{\text{DT}}(h))} + \lambda(\Phi_{\text{DT}}(h))P_{\text{KN}}(w | \pi(h)) & , \text{ if } \Phi_{\text{DT}}(h) \neq \phi; \\ P_{\text{ngram}}(w | h) & , \text{ otherwise.} \end{cases} \quad (2.13)$$

Note that all seen histories will reach a leaf node, as well as many unseen histories; only some unseen histories will invoke a bailout. Therefore although a binary decision tree node has only two branches, conceptually we have a third branch that leads to the bailout LM. Since we need the lower order LM $P_{\text{KN}}(w \mid \pi(h))$ for smoothing anyway, the bailout mechanism does not increase the complexity of our practical implementation. For the sake of brevity, we omit this aspect in the following presentation.

2.1.5 Problem and Analysis

As promising as it sounds, the decision tree language model failed to outperform the n -gram language model (Potamianos and Jelinek, 1998). Although the training and smoothing methods we presented above are better than those used in Potamianos and Jelinek (1998)'s study, their conclusion is still valid.

Statistically speaking, because the model space of the DTLM is bigger than that of the n -gram LM, we have the benefit of a lower bias. Due to the use of the greedy exchange algorithm in searching the combinatorially large space of decision trees, however, we incur a higher variance. The benefit of a lower bias is offset by the penalty of a much higher variance. Therefore the key to success is to control the variance.

2.2 Random Forest Language Models

Variance reduction for decision trees has been actively pursued for years. The idea of using an ensemble of randomized trees to control the variance emerged from several studies (Breiman, 1996; Amit and Geman, 1997; Ho, 1998) and is summarized in the definitive work of Breiman (2001).

2.2.1 Definition

A Random Forest Language Model (RFLM) (Xu and Jelinek, 2004a) is an ensemble of randomized decision tree language models. The probability of the RFLM is the average of probabilities from the $M > 1$ member DTLMs. That is to say, we define the RFLM as

$$P(W) \doteq \prod_{i=1}^m \left(\frac{1}{M} \sum_{j=1}^M P(w_i | \Phi_{\text{DT}_j}(h_i)) \right). \quad (2.14)$$

Each decision tree is trained independently with the same randomization procedure therefore the computation of the RFLM can be trivially parallelized.

ORDER OF A RFLM

The question that the basic RFLM of Xu and Jelinek (2004a) asks is whether the i -th previous word in an $(n - 1)$ -word history h belongs to some appropriate set of words and $1 \leq i < n$. We will refer to n as the *order* of the RFLM.

2.2.2 Training Randomization

Following Xu and Jelinek (2004a), we can randomize our decision tree building procedure of Section 2.1.3 in three ways:

- 1. Random Initialization** The exchange algorithm we used to split a node is greedy in the sense that it improves the objective in each iteration until it reaches a local optimum. Without additional knowledge, it is reasonable to start from a random split.
- 2. Random Selection** When splitting a node, we can ask questions on any of the history positions. Without randomization, questions on any positions compete against each other in terms of log-likelihood reduction. We can randomize this part by randomly selecting a subset of positions and pick the best question from among only the selected positions.
- 3. Random Sampling** We can use a random sample of the training data for the construction of each tree.

Previously, Xu (2005, Section 4.1.3, pages 67–69) found that random sampling (3. above) did not improve the test data perplexity when the training data is small but suggested it as a way to circumvent the difficulty of training on large amount of data (Xu and Mangu, 2005).

2.3 Understanding RFLMs

We conducted a series of experiments to further our understanding of the RFLM.

2.3.1 Data and Setup

We used the Wall Street Journal portion of the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) in our experiments. Following (Xu and Jelinek, 2004a), we preprocessed the text by lowercasing words, removing punctuations and replacing numbers with one symbol (“N”) to produce a speech-like text. Sections 00–20 (929,564 words), sections 21–22 (73,760 words) and sections 23–24 (82,430 words) were used as training, held-out and testing data, respectively. The vocabulary size was 10,000, including a special token for unknown words.

2.3.2 All Trees Are Created Equal

In this experiment, we would like to explore the possibility of tuning mixture weights of the RFLM.

Let $\mu_j = \frac{1}{M}, \forall j \in \{1, 2, \dots, M\}$. We can rewrite Equation 2.14 as

$$P(W) \doteq \prod_{i=1}^m \left(\sum_{j=1}^M \mu_j P(w_i | \Phi_{DT_j}(h_i)) \right). \quad (2.15)$$

Notice that as long as $\sum_{j=1}^M \mu_j = 1$ and $0 \leq \mu_j \leq 1, \forall j \in \{1, 2, \dots, M\}$, the language model above is properly defined. Therefore we can search for the best mixture weights

CHAPTER 2. RANDOM FOREST LANGUAGE MODELS

$\{\mu_j\}_{j=1}^M$ to maximize the likelihood via Expectation Maximization (EM) (Dempster, Laird, and Rubin, 1977).

Starting from a initial assignment $\{\mu_j^{(0)}\}_{j=1}^M$, the EM algorithm iteratively updates the mixture weights by

$$\mu_k^{(t+1)} = \frac{1}{m} \sum_{i=1}^m \frac{\mu_k^{(t)} P(w_i | \Phi_{DT_k}(h_i))}{\sum_{j=1}^M \mu_j^{(t)} P(w_i | \Phi_{DT_j}(h_i))}, \quad \forall t \geq 0, \quad (2.16)$$

until convergence.

RESULTS

First we built a baseline trigram LM with Kneser-Ney smoothing and a 100-tree forest of order 3 on Penn Treebank data. The perplexity results, including the mean and standard deviation of those from individual trees, are shown in Table 2.1.

Table 2.1: Perplexities of baseline n -gram LM and random forest LM

| Model | KN.3gm | RF-100 | Indv. Trees |
|------------|--------|--------|-------------|
| Perplexity | 143.9 | 129.7 | 160.5±0.9 |

Then we trained the mixture weights on *another* held-out data, which are distinct from the held-out data used to prune the decision trees, using Equation 2.16 and plotted perplexities of both the held-out and the testing data against iterations in Figure 2.1.

Although we could slightly lower the perplexity of the held-out data, we could not lower that of the testing data. In fact, the perplexity of the testing data went up — a sure sign of overfitting. Further examination of the trained weights showed that none of them

CHAPTER 2. RANDOM FOREST LANGUAGE MODELS

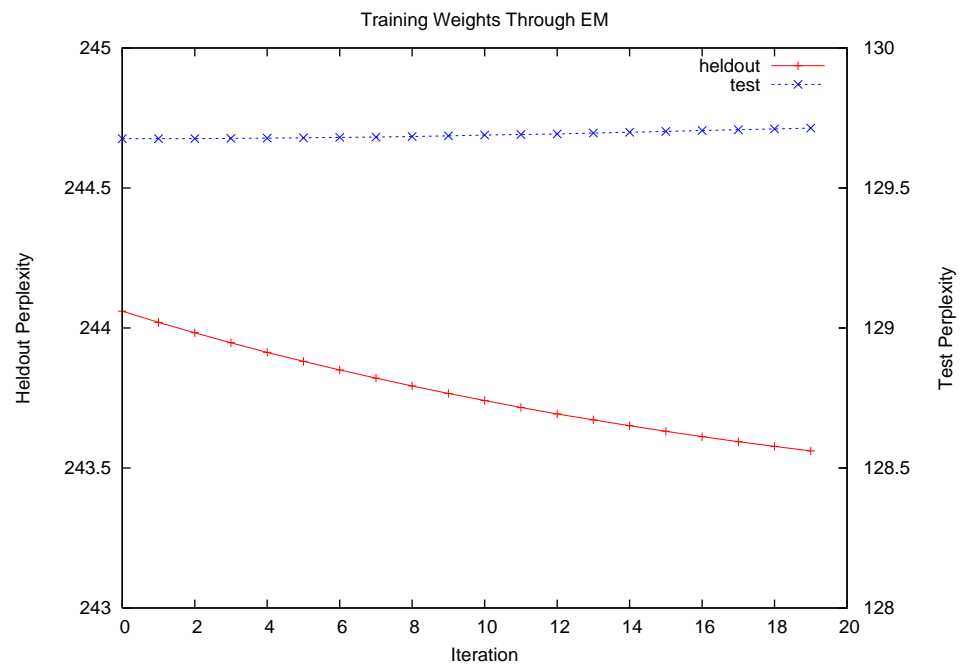


Figure 2.1: Weight training causes model to overfit the held-out data

differed much from their original values of $\frac{1}{M} = 0.01$ at all. Starting from a random point on the simplex yields similar results. The reason, we believe, is that because each tree is independently built from an identical distribution, they have similar *overall* performance, even though they differ from each other in areas of strength and weakness. Therefore we conclude that weight training for the RFLM is fruitless because “all trees are created equal”.

2.3.3 One Randomization Is Already Good Enough

In Section 2.2.2, we described three kinds of randomization, among which the last one, i.e., random sampling of training data, was found ineffective by Xu (2005). We would like to investigate the effectiveness of the first two.

We first built a forest with random initialization of the exchange algorithm while “turning off” the random selection of questions, which means that we always consider all history positions and choose the best position and best question among them. Then we built a forest with random selection of questions while “turning off” the random initialization of the exchange algorithm, which means that we use a deterministic scheme to construct the initial split. In particular, we put words with even number ids to the left; odd number ids, right.

RESULTS

To our surprise, the forest with only random initialization or random selection performed almost as well as the baseline forest, which randomized both the initialization and

CHAPTER 2. RANDOM FOREST LANGUAGE MODELS

the question selection, as shown in Table 2.2.

Table 2.2: One randomization already achieves most of the improvement

| Model | KN.3gm | RF-100 | Rand. Init. Only | Rand. Select. Only |
|------------|--------|--------|------------------|--------------------|
| Perplexity | 143.9 | 129.7 | 130.6 | 130.4 |

In the case of random initialization only, because of the greediness of the exchange algorithm, a random start is a natural choice: we have no reason to prefer one start from another. That is to say, even the earliest version of the DTLM had a built-in nature of randomness. In fact, the common practice is to run the exchange algorithm from a number of random starts and choose the best result among those runs. If we focus on only one particular tree, this practice has only a minor impact. However, as soon as we keep many instances of trees and average the probabilities, we discover the power of ensemble. This point is also supported by the study of Emami and Jelinek (2005a).

In the case of random selection only, although we use a deterministic scheme to construct every initial split, the variation caused by random selection of questions is already enough to form a good forest. Further examination showed that individual trees from this experiment had slightly higher perplexities than those in the first one. However, after we averaged the probabilities, as a forest, they were just as good.

Therefore we conclude that it is the use of ensemble that established the superiority of the RFLM and either random initialization or random selection alone is good enough for the success of the RFLM.

2.3.4 Early Stopping Is Fine

The CART-style tree building method, which grows the tree to its full extent and prunes away spurious branches, has a computational drawback: the need to compute and to store a potentially very large tree. This problem becomes severer when the training data becomes bigger.

One of the simplest alternative strategies is early-stopping, which stops tree growing once the number of leaf nodes has reached a predetermined threshold (or according to the amount of data). Note that in this case, it matters which node is split first while it doesn't when the tree is grown to its full extent.

We propose to grow the tree in a breadth-first fashion, which results in a tree of the shortest height to match our intuition that the tree represents a sophisticated smoothing scheme. At one extreme, a tree with only one node, the root, is equivalent to a unigram LM; at the other end, a fully grown tree of order n is essentially an n -gram LM. (Indeed, as we mentioned in Section 2.1.2, the n -gram LM is exactly one of the fully grown trees.) Therefore the tree growing process can be regarded as a path of refinement from the crudest to the finest. Our task is to pick a desirable compromise in the middle.

The threshold of tree size in terms of number of leaf nodes can be determined on held-out data, as the pruning threshold of the old method is.

RESULTS

We implemented the early-stopping strategy and built forests with various thresholds. We then interpolated the forests with the baseline trigram LM. The numbers are tabulated in Table 2.3.

Table 2.3: Early-stopping achieves perplexity comparable to CART-pruning

| Threshold | Perplexity | Interpolated | Weight (on RF) |
|-----------|------------|--------------|----------------|
| 2 | 369.5 | 143.2 | 0.067 |
| 10 | 265.2 | 140.8 | 0.156 |
| 100 | 180.0 | 135.3 | 0.323 |
| 1000 | 143.3 | 131.2 | 0.512 |
| 5000 | 133.7 | 130.5 | 0.674 |
| 10000 | 132.2 | 130.9 | 0.754 |
| RF-100 | 129.7 | 129.6 | 0.900 |
| KN.3gm | 143.9 | - | - |

Surprisingly, the simple early-stopping strategy worked quite well: with only 1000 leaves per tree, the forest was already comparable to the baseline trigram LM; after interpolating with the baseline trigram LM, the perplexity is approaching that of the RFLM built using the old method. A tree built using the old method had around 20000 leaves. The refinement after 1000 nodes clearly suffered the law of diminishing returns and seemed to duplicate the effort of the baseline trigram LM. Since the RFLM always needs a background n -gram LM, it makes sense to save the effort, especially when the computational resource, e.g., CPU time or memory, is at a premium.

Chapter 3

Large-Scale Training of RFLMs

The training procedure of the RFLM as described in the last chapter has a practical limitation of space complexity. We discuss the importance of addressing this problem and propose a divide-and-conquer solution.

3.1 Motivation

As we mentioned in the abstract, the art of language modeling has been dominated by the simple yet powerful n -gram LM. Nearly all commercial applications of statistical language modeling still use the n -gram LM despite the introduction of many interesting alternatives, such as Maximum Entropy Language Models (Rosenfeld, 1996; Khudanpur and Wu, 2000), Structured Language Models (Chelba and Jelinek, 2000; Charniak, 2001; Roark, 2001; Wang and Harper, 2002), Neural Probabilistic Language Models (Bengio,

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

Ducharme, and Vincent, 2001; Emami and Jelinek, 2005b; Schwenk, 2007), etc.

The popularity of the n -gram LM can be attributed to the following two reasons:

- Availability of huge amount of training text;
- Simplicity of the model itself.

Because the performance of any statistical model depends on the amount of data on which it is trained, a simple model like the n -gram LM can outperform a complicated model by using more data (Banko and Brill, 2001). If the cost of collecting more data is lower than that of using a complicated model, which is increasingly true with the rapid expansion of the Internet, it is sensible to prefer the simple model. For example, the Google n -gram corpus (Brants and Franz, 2006) is publicly available through the Linguistic Data Consortium for a nominal fee. On the other hand, the utility of a complicated model will be limited unless it scales up to a comparable level of the simple model in terms of accommodating training data, which, for many models, is not a trivial task (Wu and Khudanpur, 2000; Schwenk, 2007).

3.2 Large-Scale Training

3.2.1 Analysis of the Problem

The problem that the RFLM faces is space complexity. A straight-forward implementation of the training algorithm as described in Section 2.1.3 would store everything in the

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

memory. The most memory-consuming objects are:

1. Training n -gram counts;
2. Held-out n -gram counts;
3. Background n -gram LM;
4. Decision tree.

Once loaded in the memory, the first three items do not change. The last one, however, keeps growing as the training algorithm keeps splitting nodes to build the tree. When we have a large amount of training data, the first three items would occupy a big chunk of the memory, leaving the available space for the decision tree small. What makes the situation worse is our choice of CART-style pruning, which grows the tree to its full extent then prunes it back: we might not be able to accommodate the full tree even if we manage to deal with the pruned tree.

The guiding principle of any solution to this problem is the trade-off between time and space, or the “No Free Lunch” principle. Since no lunch is free, our goal is to buy it at a good price. That is, to overcome the space complexity problem for a cost in time complexity that is as low as possible.

3.2.2 Our Solution

We propose an efficient disk-swapping algorithm to solve the problem (Su, Jelinek, and Khudanpur, 2007). The main trade-off is between memory space and input/output (I/O)

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

time.

First we make two assumptions:

- The training n -gram counts can be loaded into the memory;
- The decision tree, full or pruned, is too big to be loaded into the memory as a whole.

The first assumption is justifiable since our goal is to make the RFLM possible for roughly the same amount of training data as the n -gram LM, which needs to load the training n -gram counts into the memory as well. The second assumption is exactly the problem we want to solve and is especially true in the case of a full tree.

Since we assume that the decision tree cannot be loaded in whole, we consider the scenario that only part of the tree is in the memory at a given time. The questions we have to answer are:

- Is it possible to build the tree piece-by-piece?
- Is it possible for the tree to perform its functionality efficiently?

For the first question, we notice that our tree growing algorithm has a locality property: the node-splitting depends only on the histories that the node contains. In other word, the splitting result remains the same whether the node is constructed in memory by a previous splitting or is loaded from the disk. Therefore it is possible to build the tree piece-by-piece as far as the growing stage is concerned. The pruning stage is a little trickier because conceptually it is a bottom-up process. As we will discuss later, however, as long as we store the right statistics, the pruning can also be performed in a local fashion.

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

For the second question, we notice that the most important functionality that a tree performs is to classify a given history into one of its leaf nodes. Because each inquiry involves swapping in and out parts of the tree from the disk several times, which is very time-consuming, it is no longer reasonable to perform it one-by-one. Instead, we perform the classification in *batch mode*, sharing the overhead of disk swapping.

ALGORITHM

The DTLM growing and pruning stages of the original algorithm in (Xu, 2005) need to be modified as follows, and one more step needs to be added to achieve the goal described above:

Growing: When the number of nodes exceeds a threshold, stop growing the tree and *swap out*; Add any unfinished nodes to an agenda (priority queue); *Swap in* the top unfinished nodes according to the agenda, and continue growing.

Pruning: Move the computation of held-out data likelihood and size to the growing stage to save one pass of disk swapping; Rewrite node-potentials as functions of held-out data likelihood and size; Compute the potential for each node recursively in a bottom-up fashion and prune away nodes whose potentials are below an empirically chosen threshold.

Clean-Up: Go through the whole tree one more time to remove any internal nodes that were not pruned during the “embedded” pruning stage described above (due to temporary unavailability of descendant node-potentials).

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

The first part of the algorithm (“Growing”) is illustrated in Algorithms 1, 2 and 3. Algorithm 3 is the same as Algorithm Node-Split(p) in Xu (2005, page 44, Figure 2.5) and reproduced here for the sake of completeness.

```
Input: Data  $D$   
Output: Fully grown tree  $T$   
1 begin  
2   Initialize agenda with the root node  $N$ ;  
3   while agenda not finished do  
4     Take node  $n$  out of agenda;  
5     Swap in data  $d$  for node  $n$  from disk;  
6      $(t, \{n_i\}, \{d_i\}) \leftarrow \text{GrowSubtree}(n, d)$ ;  
7     Write subtree  $t$  to disk file  $T$ ;  
8     Swap out data  $\{d_i\}$  for nodes  $\{n_i\}$  to disk;  
9     Put unfinished nodes  $\{n_i\}$  in agenda;  
10  end  
11  Return tree  $T$ ;  
12 end
```

Algorithm 1: Grow(D)

3.3 Experiments

3.3.1 Modified Smoothing

In the first experiment we would like to verify our hypothesis that the modified Kneser-Ney smoothing (Chen and Goodman, 1999) improves the performance of the RFLM as it does that of the n -gram LM. We used Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993), as described in Section 2.3.1. We contrasted the regular with the modified Kneser-Ney smoothing scheme when the forest consisted of y trees, where

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

Input: Node N and its data D

Output: Tree T , unfinished nodes and their data $(\{n_i\}, \{d_i\})$

```

1 begin
2   Initialize agenda with node  $N$ ;
3    $K \leftarrow 1$ ;
4   while agenda not finished and  $K \leq \text{threshold}$  do
5     Take node  $n$  out of agenda;
6     if SplitNode( $n, d$ ) succeeds then
7       Put children nodes  $n_L$  and  $n_R$  in agenda;
8        $K \leftarrow K + 2$ ;
9     end
10  end
11  Return tree  $T$ , remaining nodes  $\{n_i\}$  in agenda and their data  $\{d_i\}$ ;
12 end

```

Algorithm 2: GrowSubtree(N, D)

Input: Node N and its data D

Output: Children nodes $\{N_L, N_R\}$ and their data $\{D_L, D_R\}$

```

1 begin
2   Group histories in  $D$  into basic elements and put them in  $B$ ;
3   Initialize sets  $A$  and  $\bar{A}$  to be a split of set  $B$ ;
4   while any element moved do
5     foreach element  $x$  in  $A$  do
6       Compute likelihood increase if  $x$  were moved from  $A$  to  $\bar{A}$ ;
7       if increase  $> 0$  then
8         Move  $x$  from  $A$  to  $\bar{A}$ ;
9       end
10    end
11    foreach element  $y$  in  $\bar{A}$  do
12      Compute likelihood increase if  $y$  were moved from  $\bar{A}$  to  $A$ ;
13      if increase  $> 0$  then
14        Move  $y$  from  $\bar{A}$  to  $A$ ;
15      end
16    end
17  end
18  Split  $N$  into  $N_L$  and  $N_R$ ,  $D$  into  $D_L$  and  $D_R$  according to  $A$  and  $\bar{A}$ ;
19  Return children nodes  $\{N_L, N_R\}$  and their data  $\{D_L, D_R\}$ ;
20 end

```

Algorithm 3: SplitNode(N, D)

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

$y = 1, 2, 3, 4, 5, 7, 10, 20, 50$ and 100 . Because of the random nature of the RFLM, each experiment was repeated 10 times. The mean and standard deviation of the test data perplexities were plotted as error bars in Figure 3.1.

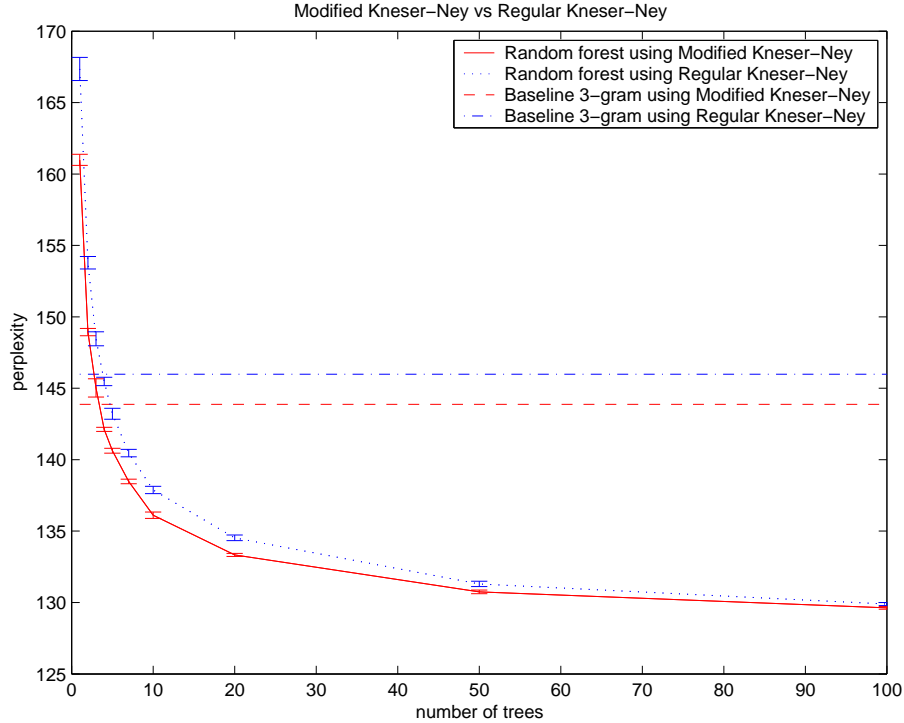


Figure 3.1: Smoothing RFLM: modified vs regular

We conclude from Figure 3.1 that the modified Kneser-Ney smoothing did slightly improve upon the regular, as expected. More interestingly, the gap decreased as the number of trees in the forest increased, supporting the claim in (Xu, 2005) that the RFLM beats the n -gram LM by better smoothing. Because the additional computation incurred when switching from regular to modified smoothing was negligible, we used the latter henceforth.

3.3.2 Learning Curves

Our second experiment was to compare the learning curves of the Kneser-Ney-smoothed 4-gram language model with the RFLM of the same order. For this purpose we chose to work with the 525 million words English Web data from University of Washington, which were also used in (Xu and Mangu, 2005). The first x million words of the Web data, where $x = 1, 2, 4, 8, 16, 32$ and 64 , were used as training data and 2 million words from the Fisher corpus (Cieri, Miller, and Walker, 2004) were used as held-out. (The largest amount of training text we could accommodate, without any counts cut-off, was about 100M words, given that the addressable space for a single process in a 32-bit machine is 4GB.) Another 2 million words from the Fisher corpus were used for testing. The vocabulary contained 30k word types. The number of trees per forest was 100.

From Figure 3.2 we can see that the RFLM scaled at least as well as the n -gram LM, while keeping a more than 10% lead in perplexity.

3.3.3 Lattice Rescoring

Building a RFLM for the IBM GALE Mandarin ASR system was exactly the challenge this work was trying to take. The acoustic model of this recognizer was trained on 585 hours of speech with Perceptual Linear Predictive (PLP) features (Hermansky, 1990) by first building a speaker-independent model, then adapting it to speaker clusters using Vocal Tract Length Normalization (VTLN) (Cohen, Kamm, and Andreou, 1995), feature-space

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

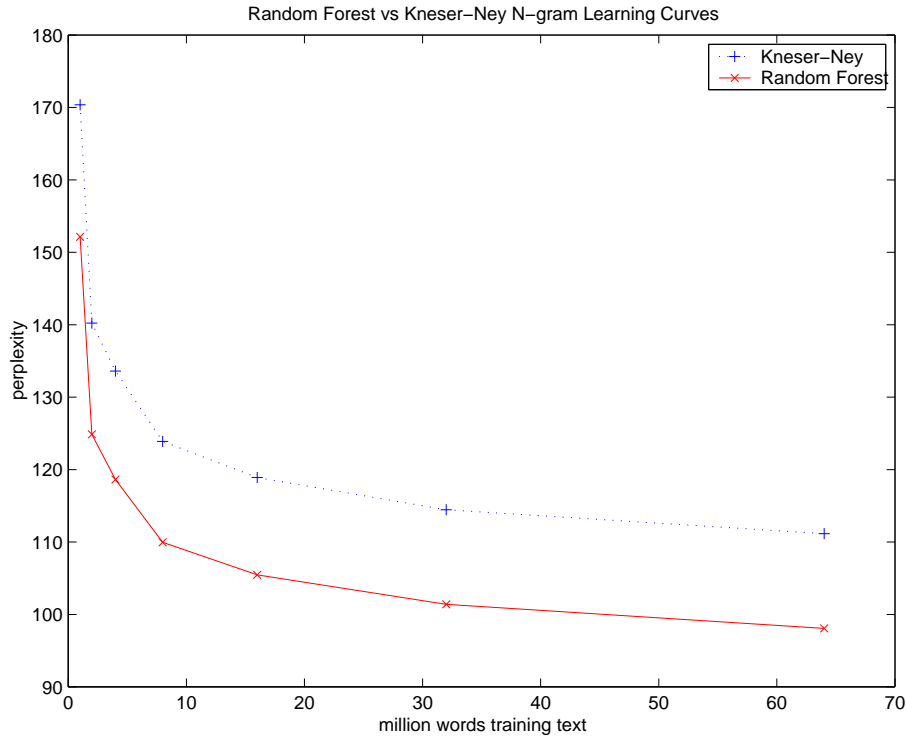


Figure 3.2: Learning curves in terms of perplexity

Maximum Likelihood Linear Regression (fMLLR) (Gales, 1998) and feature-space Minimum Phone Error (fMPE) training (Povey, Kinsbury, Mangu, Saon, Soltau, and Zweig, 2005).

Because the content of this task was mainly broadcast news (BN) and broadcast conversation (BC), the available amount of text for training a language model was huge. Specifically the text we used was organized into seven parts: xin.0 and xin.1 (Xinhua News), cna.1.0 and cna.1.1 (Central News Agency), pdcr (People’s Daily and China Radio), tdt (TDT-2,3,4 and HUB-4), gwd (UWashington general web data). Each contained about 100 million words as segmented by a maximum-matching segmenter with respect to the 107k

CHAPTER 3. LARGE-SCALE TRAINING OF RFLMS

vocabulary used by the acoustic model.

We built a 4-gram RFLM with 50 trees for each of the seven parts then interpolated them together to rescore the lattices generated by the IBM system. (We used 50 trees per forest instead of 100, as we usually do, because 50 trees worked as well as 100 on the held-out data but halved the computational effort.) Interpolation weights were determined using the current one-best output on a per show basis. Note that these lattices had already been rescored once by a regular 4-gram LM (“Baseline”), which was trained on 602 million words text from 13 different sources.

Table 3.1: Lattice rescoring for IBM GALE Mandarin ASR

| CER | All | BN | BC |
|----------|------|------|------|
| Baseline | 18.9 | 14.2 | 24.8 |
| RFLM | 18.3 | 13.4 | 24.4 |

As shown in Table 3.1, we got a 0.6 absolute reduction in Character Error Rate (CER), which was statistically significant in a sign test with $p < 0.001$.

Chapter 4

Knowledge Integration With RFLMs

4.1 Main Proposal

4.1.1 Motivation

The LMs we have considered so far only make use of n -gram counts. In fact, besides some smoothing methods tailored for natural languages, those models take into no account of the fact that they are modeling human languages, rather than a general sequences of symbols. Although ignoring that fact does allow the techniques developed in language modeling to benefit a broad range of problems, it does not exploit the subtle characteristics of human languages.

One of the most obvious limitations of the n -gram LM is the lack of long-range dependency. Because the LM cuts off the influence of any word that is more distant than

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

$(n - 1)$ previous positions, the n -gram LM will miss important linguistic information, such as subject-predicate agreement.

In any case, the model does not “know” anything about subject-predicate agreement. Being statistical, it can only discover that somehow the word “I” has a strong correlation with the word “am” but not much correlation with the word “is”. Given more and more data, the model might eventually learn the agreement but we can never get “enough” data because the number of n -grams grows exponentially with n .

Therefore, besides using sophisticated statistical models like maximum entropy, neural networks and random forests, we believe that it is important to keep integrating linguistic knowledge into language models (Khudanpur and Wu, 2000; Emami and Jelinek, 2005b; Xu and Jelinek, 2004b).

Note that unlike the studies of Emami and Jelinek (2005b) and Xu and Jelinek (2004b), we do not focus on how the knowledge is acquired: it can come from either a rule-based system or a data-driven statistical model. Our approach resembles most the study of Khudanpur and Wu (2000) in terms of methodology.

The key insight of our approach is that the decision trees of the RFLM can ask *any* question about the history as long as the information it asks about is available from some knowledge source. In some sense, our engineering task of modeling languages has a scientific core in the form of the following question:

Given the history, what information and/or knowledge would make a person choose the next word, as opposed to any other word?

4.1.2 Definitions

Let $W = w_1w_2 \cdots w_m \in V^*$ be a word sequence, where V is the vocabulary. Let $h_i = w_1^{i-1}$ denote the history of the word w_i .

Definition 4.1.1. A *feature* of a history h_i is a function $f(h_i)$ that maps h_i to an element of a finite set E . That is,

$$f(h_i) : V^* \mapsto E, \quad (4.1)$$

where E is a finite set of feature values.

Definition 4.1.2. A *feature vector* of a history h_i is a vector function $F(h_i)$ whose components are features of h_i . That is,

$$F(h_i) \doteq (f_1(h_i), f_2(h_i), \dots, f_k(h_i)) : V^* \mapsto E_1 \times E_2 \times \cdots \times E_k, \quad (4.2)$$

where $E_j, j \in \{1, 2, \dots, k\}$ is a finite set of feature values for the j -th feature. Therefore $F(h_i)$ is a feature space representation of the history h_i .

Definition 4.1.3. A *word feature* of a history h_i is a function $\text{WORD}_j(h_i)$ that returns the word in the $(i - j)$ position:

$$\text{WORD}_j(h_i) \doteq w_{i-j}. \quad (4.3)$$

Note that the domain of the word features happens to be the vocabulary V .

Definition 4.1.4. A *question* of a history h_i on a feature f is the indicator function $q_S^f(h_i)$

of the set $\{h : f(h) \in S \subset E\}$. That is,

$$q_S^f(h_i) = \begin{cases} 1 & , \text{ if } f(h_i) \in S \subset E; \\ 0 & , \text{ otherwise.} \end{cases} \quad (4.4)$$

The questions we have used so far are questions on word features with $E = V$. Once more information about the history is available via additional features, the RFLM will be able to take advantage of it by asking questions about them in addition to questions about the word features.

4.2 Modeling Morphologically Rich Languages

4.2.1 Introduction

Increasing effort is focused on processing languages with a rich morphology, such as Arabic or Czech. The n -gram LM is less successful in modeling morphologically rich languages than languages with simple morphology for the following reasons:

1. Morphologically rich languages tend to be freer in word order, to which the n -gram LM pays more attention as n becomes bigger.
2. Morphologically rich languages need much larger vocabularies in order to cover a certain amount of text than their morphologically impoverished counterparts do.

A direct consequence of the latter observation is that data sparseness is severer in morphologically rich languages. Take Modern Standard Arabic (MSA) for instance. In the Ara-

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

bic Gigaword corpus, an 847,935-word vocabulary is needed to reach an out-of-vocabulary (OOV) rate of 1.43% (Choueiter, Povey, Chen, and Zweig, 2006); for English, it only takes around 60,000-word vocabulary to achieve the same level of coverage (Afify, Abdou, Guo, Makhoul, Nguyen, and Xiang, 2004).

Since an Arabic word can be decomposed into several morphemes, an obvious solution is to use morpheme as the modeling unit. Although using morphemes does reduce the OOV rate dramatically, it does not help the performance of the LM in either perplexity and WER. The reason is that not every morpheme sequence corresponds to a valid word sequence. For example, a prefix can't appear between two suffixes. But an n -gram morpheme language model still assigns a (presumably low) probability to it. In a sense, the model "wastes" some probability mass on invalid sequences. So the advantage of a small vocabulary is gained at the price of increased model confusedness. In case lots of data are available, the benefit of morpheme-based LM disappears when a very large vocabulary is used (Byrne, Hajic, Ircing, Jelinek, Khudanpur, Krbec, and Psutka, 2001; Afify et al., 2004; Choueiter et al., 2006). Cited BBN work on Arabic LM.

Our task falls into the scenario where large amount of data are available. Therefore we focus on how to utilize morphology to build a better word LM with a very large vocabulary.

4.2.2 Arabic Morphology

To ground our following presentation, we give a very brief introduction to Arabic morphology based on the tutorial of Habash (2005). We focus on Modern Standard Arabic, a

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

widely used dialect in Arabic-speaking world for written and formal oral communication. Figure 4.1 shows an example of an Arabic word, whose romanization is “walilmaktabāt”, meaning “and for the libraries”. Note that Arabic writes from right to left.

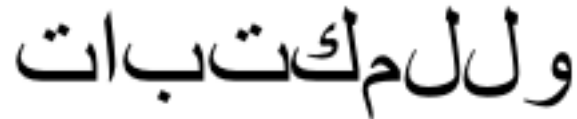


Figure 4.1: An Arabic word

An Arabic word usually consists of a stem preceded by prefixes and followed by suffixes. A stem usually comes from a root applied to a pattern. For example, the Arabic word “walilmaktabāt” can be decomposed as follows:

$$\begin{aligned} \text{(Romanized Arabic)} \quad \text{walilmaktabāt} &= \text{wa} + \text{li} + \text{al} + \text{maktaba} + \text{āt}; \\ \text{(English gloss)} \quad \text{and-for-the-libraries} &= \text{and} + \text{for} + \text{the} + \text{library} + \text{plural}. \end{aligned} \tag{4.5}$$

The stem “maktaba” can be decomposed as follows:

$$\begin{aligned} \text{(Romanized Arabic)} \quad \text{maktaba} &= \text{maCCaCa} * \text{ktb}; \\ \text{(English gloss)} \quad \text{library} &= \text{some-pattern} * \text{notion-of-writing}. \end{aligned} \tag{4.6}$$

We use “*” to denote the operation of applying a pattern to a root. For example, applying the pattern “maCCaCa” to the root “ktb” means to replace three placeholders “C” in the pattern with three letters of the root “ktb” to get “maktaba”.

Like any other linguistic analysis, Arabic morphology is ambiguous. Fortunately this ambiguity is generally mild and a good Arabic morphological analyzer like Darwish (2002)

can generate a ranked list of all possible analyses, with the highest ranked being the most probable.

4.2.3 Features

Let r_i , p_i and s_i denote the root, prefix and suffix of the word w_i , respectively, as provided by a morphological analyzer in the most probable decomposition. In case of more than one prefix (or suffix), we group them together to form a “prefix compound” (or “suffix compound”). For example, if w_i is the word “walilmaktabāt”, then r_i , p_i and s_i are “ktb”, “walil” and “āt”, respectively. Following the definition of the word feature in Equation 4.3, we can define morphological features as follows:

Definition 4.2.1. A *root feature* of a history h_i is a function $\text{ROOT}_j(h_i)$ that returns the root of the word in the $(i - j)$ position:

$$\text{ROOT}_j(h_i) \doteq r_{i-j}. \quad (4.7)$$

The domain of the root features is the vocabulary of roots.

The definitions of the prefix feature $\text{PREFIX}_j(h_i)$ and the suffix feature $\text{SUFFIX}_j(h_i)$ are similar to that of the root feature and are therefore omitted. In addition to the morphological features, we also define the part-of-speech (POS) feature $\text{POS}_j(h_i)$ as provided by an Arabic POS tagger (Diab, Hacıoglu, and Jurafsky, 2004).

With all the extracted features, the feature vector representation (Equation 4.2) of a

trigram history h_i is

$$\begin{aligned}
 F(h_i) = & (\text{PREFIX}_2(h_i), \text{PREFIX}_1(h_i), \text{SUFFIX}_2(h_i), \text{SUFFIX}_1(h_i), \\
 & \text{ROOT}_2(h_i), \text{ROOT}_1(h_i), \text{POS}_2(h_i), \text{POS}_1(h_i), \\
 & \text{WORD}_2(h_i), \text{WORD}_1(h_i))
 \end{aligned} \tag{4.8}$$

$$= (p_{i-2}, p_{i-1}, s_{i-2}, s_{i-1}, r_{i-2}, r_{i-1}, t_{i-2}, t_{i-1}, w_{i-2}, w_{i-1}), \tag{4.9}$$

where t_j denote the POS tag of the word w_j , $j = \{1, 2, \dots, m\}$.

4.2.4 Experiments

We used 3M words broadcast news (BN) and 4M words broadcast conversation (BC) transcripts from the LDC GALE data releases. We built a 4-gram LM with modified Kneser-Ney smoothing as baseline. The vocabulary size was 350K.

We preprocessed the training, held-out and testing data by running a morphological analyzer (Darwish, 2002) and a POS tagger (Diab et al., 2004) to extract the root, prefix, suffix and POS tag of each word. Then we trained random forests of order four with various combination of features and tabulated the results in Table 4.1.

As we see in Table 4.1, we can lower the perplexity of the LM by introducing morphological features although the decrease wasn't as much as we had hoped. We tested the new LMs in the IBM GALE Arabic ASR system and, unfortunately, could not lower the WERs, shown in Table 4.2.

Table 4.1: Perplexity results for Arabic RFLMs

| Model | BN | Reduction | BC | Reduction |
|----------------|------|-----------|------|-----------|
| KN.4gm | 1844 | - | 1425 | - |
| RF-50 | 1721 | 6.67% | 1336 | 6.25% |
| +root | 1698 | 7.92% | 1328 | 6.81% |
| +pos | 1713 | 7.10% | 1334 | 6.39% |
| +root+pos | 1696 | 8.03% | 1329 | 6.74% |
| +root+suffix | 1702 | 7.70% | 1331 | 6.60% |
| +suffix+prefix | 1722 | 6.62% | 1345 | 5.61% |

Table 4.2: Word error rate results for Arabic RFLMs

| Model | dev07 | eval06dev |
|------------|-------|-----------|
| Baseline | 12.5 | 20.9 |
| KN.4gm | 12.6 | 20.7 |
| RF-50 | 12.5 | 20.6 |
| RF-50+root | 12.6 | 20.7 |

4.3 Exploiting Prosodic Breaks in LMs

4.3.1 Introduction

Prosody refers to a wide range of suprasegmental properties of spoken language units, such as tone, intonation, rhythm, stress and so on. It has been used for a number of spoken language processing tasks, such as disfluency and sentence boundary detection (Stolcke, Shriberg, Bates, Ostendorf, Hakkani, Plauche, Tür, and Lu, 1998), topic segmentation (Hirschberg and Nakatani, 1998), spoken language parsing (Hale, Shafran, Yung, Dorr, Harper, Krasnyanskaya, Lease, Liu, Roark, Snover, and Stewart, 2006), among others. We are mainly interested in using prosody to improve automatic speech recognition (ASR). As a separate knowledge source, prosody has been helpful in all three major components of a modern ASR system: the acoustic model (Shafran and Ostendorf, 2003; Chen, Borys,

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

Hasegawa-Johnson, and Cole, 2003), the pronunciation model (Fosler-Lussier, 1999) and the language model (Stolcke, Shriberg, Hakkani-Tür, and Tür, 1999; Hirose, Minematsu, and Terao, 2002). (For a comprehensive review of prosody models in ASR, see Ostendorf, Shafran, and Bates (2003).) New opportunities of using prosody emerged after the availability of a prosodically labeled speech corpus (Ostendorf, Shafran, Shattuck-Hufnagel, Carmichael, and Byrne, 2001), where tones and breaks were hand-labeled with a subset of ToBI labeling scheme (Silverman, Beckman, Pitrelli, Ostendorf, Wightman, Price, Pierrehumbert, and Hirschberg, 1992). In this work, we focus on exploiting prosodic breaks for language modeling with random forests (Su and Jelinek, 2008).

PROSODIC BREAK INDEX

A *prosodic break index* is a number representing the subjective strength of one word’s association with the next. The ToBI scheme for English assigns numbers from 0 to 4 to indicate the perceived association from the strongest conjoining to the most disjoining. For example, we might hear the sentence “Time flies like an arrow” pronounced in the following way:

$$\text{Time/3 flies/2 like/1 an/0 arrow/4.} \quad (4.10)$$

We use the notation “Time/3” to indicate that the break index between the word “Time” and its follower is 3. However, Expression 4.10 is not the only way of reading this sentence. For example, another person might read this sentence in the following way:

$$\text{Time/1 flies/3 like/2 an/0 arrow/4.} \quad (4.11)$$

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

In contrast to Expression 4.10, which puts a longer break between “Time” and “flies” than between “flies” and “like”, probably implying “Time” itself is a noun phrase, like “Space”, and “flies” is a verb, Equation 4.11 puts a shorter one, probably implying that “Time flies” is a noun phrase, like “fruit flies”, and “flies” is a noun. Therefore prosodic breaks help resolve syntactic ambiguity (Dreyer and Shafran, 2007).

Prosodic breaks also help lexical prediction. For example, consider the following two sentences, omitting break index “0” for brevity:

1. What *sort of* benefits would you like to get from a big company/1.
2. It just *sort of*/2 happens automatically/1.

Although the trigram history, “sort of”, appears in both sentences, they differ a lot in the distribution of immediate following words. Without prosodic break information, a trigram language model would not be able to treat them differently. However, because people tend to insert a longer break when using the phrase “sort of” in an adverbial way, as in the second sentence, a prosodic language model can, in principle, pick up the cue and adjust the prediction accordingly.

4.3.2 Speech Recognition with Side Information

The task of ASR has only two variables: the input acoustic signal A and the output word sequence W . In the source-channel formulation of Equation 1.8, we decompose the model into two: the acoustic model (AM) $P(A | W)$ and the language model (LM) $P(W)$. When

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

additional side information is available, it can be used to improve either the AM or LM, or even both. For example, intuitively, prosody should help the AM even more than the LM. However, we are mainly interested in exploiting the side information for improving the LM.

Let S denote the side information. Depending on its nature, we have the following two proposals of using it:

- If S is hidden, then

$$\begin{aligned} W^* &= \arg \max_W P(W | A) \\ &= \arg \max_W P(A | W)P(W) \\ &= \arg \max_W P(A | W) \sum_S P(W, S). \end{aligned} \quad (4.12)$$

This is the main idea behind various structured language models (Chelba and Jelinek, 2000; Charniak, 2001; Roark, 2001; Wang and Harper, 2002), where S is the hidden syntactic parse tree.

- If S is observable, then

$$\begin{aligned} W^* &= \arg \max_W P(W | A, S) \\ &= \arg \max_W P(A | W, S)P(W, S) \\ &\approx \arg \max_W P(A | W)P(W, S). \end{aligned} \quad (4.13)$$

For the sake of simplicity, we make an independence assumption that $P(A | W, S) \approx P(A | W)$ so that we can keep the AM intact while investigating the effect of S on the LM.

Note that, strictly speaking, only the acoustic signal A is observable in ASR. However, if the side information S can be predicted by the acoustic signal A with high accuracy, we can choose to model it as observable, in which case, Equation 4.13 becomes

$$W^* \approx \arg \max_W P(A | W)P(W, S(A)) = \arg \max_W P(A | W)P(W | S(A)), \quad (4.14)$$

where $S(A) = \arg \max_{S'} P(S' | A)$ is a function of A . At any rate, both Equation 4.12 and 4.13 need to compute the joint probability $P(W, S)$.

4.3.3 Prosodic Language Models

Let $(W, S) = w_1 s_1 w_2 s_2 \cdots w_m s_m$ denote a sequence of words and prosodic breaks in temporal order, where $W = w_1 w_2 \cdots w_m$ is the word sequence of length m and $S = s_1 s_2 \cdots s_m$ is the sequence of prosodic breaks for W , where s_i is the break between w_i and w_{i+1} , $i = \{1, 2, \dots, m\}$; and s_m is the utterance break.

To compute $P(W, S)$, we first make a Markovian assumption similar to that of the n -gram LM:

$$\begin{aligned} P(W, S) &= \prod_{i=1}^m P(w_i, s_i | w_1^{i-1}, s_1^{i-1}) \\ &\approx \prod_{i=1}^m P(w_i, s_i | w_{i-n+1}^{i-1}, s_{i-n+1}^{i-1}). \end{aligned} \quad (4.15)$$

Now for the computation of $P(w_i, s_i | w_{i-n+1}^{i-1}, s_{i-n+1}^{i-1})$, we propose two methods:

- **Tuple N -gram Model:** Let $t_i = (w_i, s_i)$, for all $0 \leq i \leq m$. We have

$$P(w_i, s_i | w_{i-n+1}^{i-1}, s_{i-n+1}^{i-1}) = P(t_i | t_{i-n+1}^{i-1}). \quad (4.16)$$

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

Then we can build an n -gram LM or RFLM of t_i 's, whose vocabulary is the Cartesian product of the word vocabulary and the prosodic break vocabulary.

- **Decomposition Model:** We can decompose the probability as follows:

$$P(w_i, s_i | w_{i-n+1}^{i-1}, s_{i-n+1}^{i-1}) = P(w_i | w_{i-n+1}^{i-1}, s_{i-n+1}^{i-1})P(s_i | w_{i-n+1}^i, s_{i-n+1}^{i-1}). \quad (4.17)$$

Then we can build two n -gram LMs or RFLMs for predicting the word and the break, respectively.

4.3.4 Experiments

4.3.4.1 Data and Setup

We used the ToBI-labeled Switchboard data from Ostendorf et al. (2001). Following Hale et al. (2006), we divided our data into training (665,790 words), development (51,326 words) and evaluation (49,494 words, 55,529 counting the end of sentence symbols). Due to the relatively small size of the corpus, our LMs would only consider up to two words and two breaks in the history, if not specified otherwise. We built 100 trees for each RFLM and the smoothing method for both regular n -gram LMs and RFLMs was always the modified Kneser-Ney (Chen and Goodman, 1999). The vocabulary size was 10k.

4.3.4.2 Granularity of Prosodic Breaks

The decision tree classifier in Hale et al. (2006) provided three degrees of granularity: two-level (break or not), three-level (ToBI indices 1, 4 and p) and continuous-valued (quantized into 12 values, 0.0, 0.1, \dots , 1.0 and -1.0). We built three RFLMs for $P(w_i \mid w_{i-1}, w_{i-2}, s_{i-1}, s_{i-2})$, where the breaks s_{i-1} and s_{i-2} took values of different granularity. The baseline was the word trigram LM, $P(w_i \mid w_{i-1}, w_{i-2})$, with modified Kneser-Ney smoothing. The perplexity results on the evaluation data are tabulated in Table 4.3.

Table 4.3: Granularity of prosodic breaks

| Model | two-level | three-level | cont.-valued |
|--------|-----------|-------------|--------------|
| KN.3gm | 66.1 | 66.1 | 66.1 |
| RF-100 | 65.5 | 65.4 | 56.2 |

From Table 4.3, we concluded that the ToBI indices were not fine-grained enough for the purpose of language modeling. Henceforth our experiments used the continuous-valued breaks.

4.3.4.3 Feature Selection by RFLM

As we mentioned before, from a RFLM’s point of view, the various variables in the history, w_i ’s or s_i ’s, are just features. The model chooses any one of them simply because it has strong correlation, or large mutual information, with the future word. So by asking the RFLM *not* to use one of the variables in the history, we can find out how valuable that feature is. This kind of feature engineering was also used in maximum entropy LMs

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

(Rosenfeld, 1996; Khudanpur and Wu, 2000).

We built RFLMs for $P(w_i | w_{i-1}, w_{i-2}, s_{i-1}, s_{i-2})$ then masked out one of the features in order to see how much it contributed.

Table 4.4: Feature selection by RFLM

| History | Perplexity |
|--------------------------------------|-------------|
| $w_{i-1}, w_{i-2}, s_{i-1}, s_{i-2}$ | 56.2 |
| $w_{i-1}, w_{i-2}, s_{i-1}$ | 55.9 |
| $w_{i-1}, w_{i-2}, s_{i-2}$ | 63.9 |
| w_{i-1}, w_{i-2} | 62.3 |

As we had expected, Table 4.4 showed that the break between the immediately previous word and the future word, s_{i-1} , helped the prediction, while the break between the previous and the proceeding of the previous, s_{i-2} , did not. Adding the latter actually hurt the perplexity a little bit, although that might change if we had more data. Similar experiments can be carried out for $P(s_i | w_i, w_{i-1}, w_{i-2}, s_{i-1}, s_{i-2})$. We omit the details here, but the conclusion was that the most useful features for predicting a break were the previous two words, w_i and w_{i-1} , which was consistent with our intuition.

We also point out here that these kinds of experiments would *not* have been so easy to carry out in the case of regular n -gram LMs with modified Kneser-Ney smoothing.

4.3.4.4 Main Perplexity Results

Having selected the features, we put the two components together (Equation 4.17) to get $P(w_i, s_i | w_{i-1}, w_{i-2}, s_{i-1}, s_{i-2})$ and called it the “decomp.” (decomposition) method in Table 4.5. For comparison, we also followed Equation 4.16 to get the same quantity with

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

a trigram LM of (word, break)-tuples and called it the “tuple 3gm” method in Table 4.5. For each method, we contrasted the modified Kneser-Ney-smoothed n -gram LM (“KN” column) with the RFLM (“RF” column).

Table 4.5: Main perplexity results of prosodic LMs

| Model | Method | KN | RF |
|------------------------------|-----------|------|-------------|
| $P(W, S)$ | tuple 3gm | 358 | 306 |
| | decomp. | 274 | 251 |
| $P(W)$ $= \sum_S P(W, S)$ | tuple 3gm | 69.3 | 67.2 |
| | decomp. | 66.8 | 64.2 |
| $P(W)$ | word 3gm | 66.1 | 62.3 |

As shown in Table 4.5, the best perplexity resulted from the decomposition method using the RFLM in both the model $P(W, S)$, where the prosodic breaks were given, and the model $P(W) = \sum_S P(W, S)$, where the prosodic breaks were hidden.

If we knew nothing about the prosodic breaks, we could still build a trigram LM with the modified Kneser-Ney smoothing or the RF. We called it the “word 3gm” method and put the perplexity results in the last row of Table 4.5. We observed that although our best number (64.2) for the model $P(W) = \sum_S P(W, S)$ was better than a modified Kneser-Ney-smoothed trigram LM (66.1), it was outperformed by the basic RFLM (62.3), as shown in the bottom right corner of Table 4.5. The reason was that although prosodic breaks helped predict the next word, as shown in Table 4.3 as well as in Table 4.4, we had to predict the breaks themselves from preceding words and breaks, if they were not given as input. Unfortunately, the benefit of using the breaks to help word prediction was offset by the mediocre quality of predicting the breaks from history.

Therefore we concluded that given prosodic breaks, we could successfully reduce the LM perplexity by using the decomposition method of Equation 4.17 and the RFLM.

4.4 Combining Syntax and Semantics

4.4.1 Introduction

A human language differs from any collection of sequences of symbols in its syntactic and semantic regularities: a random sequence of words is most likely meaningless and therefore very unlikely to be ever uttered by a human being. Because the n -gram LM implicitly models syntactic and semantic information in an inefficient way, effort has been put in modeling syntax and semantics explicitly and/or taking advantage of syntactically or semantically annotated data. For example, structured language models (Chelba and Jelinek, 2000; Charniak, 2001; Roark, 2001; Wang and Harper, 2002) aim mainly at syntactic information while topic language models (Chen, Seymore, and Rosenfeld, 1998; Gildea and Hofmann, 1999; Florian and Yarowsky, 1999; Iyer and Ostendorf, 1999; Bellegarda, 2000, among others) mainly go after topic/shallow semantic information. After these two kinds of efforts, it is natural to combine them to exploit the synergy between syntax and semantics because humans use both information together after all. The key to successful combination of different knowledge sources is the suitable choice of a mathematical model. For example, Khudanpur and Wu (2000) and Cui, Su, Hall, and Jelinek (2007) use a maximum

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

entropy LM while Wang, Wang, Greiner, Schuurmans, and Cheng (2005) uses Markov random fields.

We propose to use the RFLM for the combination of syntactic and semantic information by exploiting the fact that its member decision trees can ask any question about the history.

4.4.2 Features

SYNTACTIC FEATURES

The syntactic features we use are extracted by the incremental Probabilistic Context-Free Grammar (PCFG) parser of Roark (2001). The parser works in a left-to-right fashion, updating a priority queue of candidate partial parses each time it consumes a word. We take the top candidate and extract the following four features:

- **POS tag:** the part-of-speech tag of the last word of the history;
- **Parent:** the parent NonTerminal (NT) of the last word of the history;
- **Left sibling:** the closest left sibling NT of the last word of the history;
- **C-commander:** the closest c-commanding NT of the last word of the history.

If any of the features doesn't exist, we give it the null value (NULL). For example, suppose that the top candidate partial parse that the parser finds for the word sequence "Spotted the" is shown in Figure 4.2, which is reproduced from Roark (2001, Figure 2, page 253). The POS tag, parent, left sibling and c-commander feature values for predicting the

next word (“ball”) are DT, NP, NULL and VBD, respectively, shown with red circles in Figure 4.2.

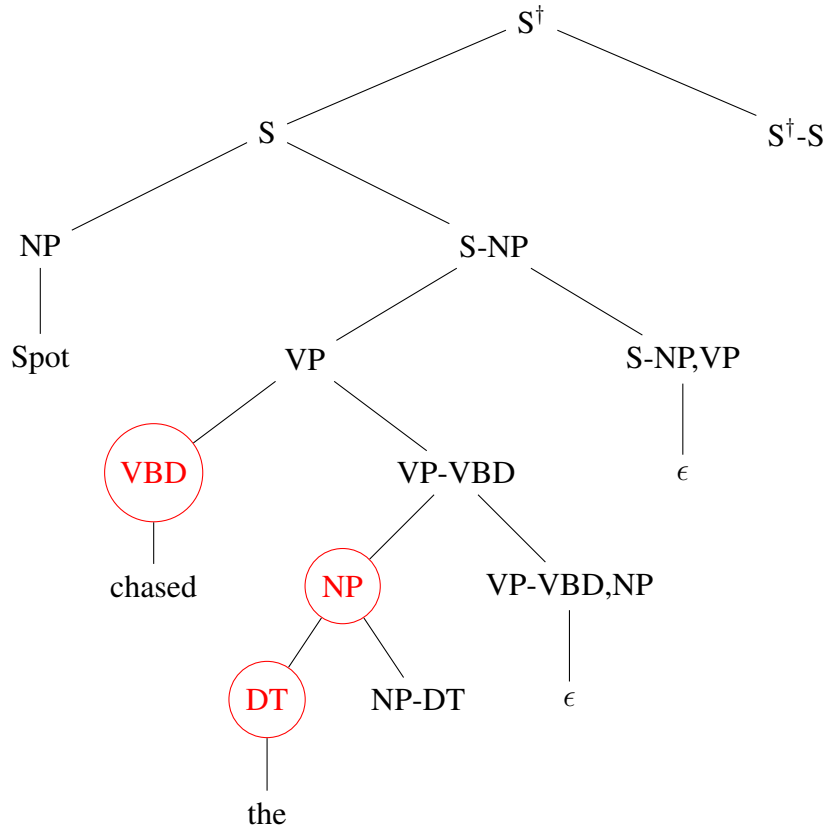


Figure 4.2: A left-factored partial parse tree

Formally speaking, we have defined four new features for our decision tree building algorithm. For example,

Definition 4.4.1. A *c-commander feature* of a history h_i is a function $CC(h_i)$ that returns the closest *c*-commanding NT of the word w_{i-1} , as determined by the current top candidate partial parse from an incremental parser.

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

The requirement of the parser to be incremental is necessary for the proper computation of the perplexity. The other three definitions are similar, therefore omitted. The intuition of choosing these features is detailed in Roark (2001, Section 4.1, pages 257–261).

TOPIC FEATURE

We use the Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan, 2003) to automatically extract the topic feature at the sentence level. Unlike the syntactic features, the topic feature does not change very quickly. In fact, topic is only well-defined at the document level. Therefore, we treat the collection of all previous sentences as a document and update its topic distribution each time we add a new sentence to that collection. Although the LDA returns a distribution of topics, we only use the most likely topic as our feature. That is,

Definition 4.4.2. A *topic feature* of a history h_i is a function $\text{TOPIC}(h_i)$ that returns the most likely topic, as determined by the LDA topic distribution, up to the previous sentence.

4.4.3 Experiments

We used the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993) with our usual preprocessing steps and data partition as described in Section 2.3.1. We built 100 trees per forest. The vocabulary size was 10k.

We trained the Roark (2001) parser on our training data and parsed the training, held-out and testing data incrementally to get the desired syntactic features. We also trained a LDA model with 10 topics on the training data and evaluated it on the training, held-out

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

and test data incrementally to get the topic feature.

We trained a trigram language model with modified Kneser-Ney smoothing as the baseline. We trained RFLMs with different combinations of features and the results are presented in Table 4.6.

Table 4.6: Perplexity results with syntactic and topic features

| Model | Perplexity |
|----------------|------------|
| KN.3gm | 143.9 |
| Roark | 155.6 |
| interp. KN.3gm | 126.0 |
| RF-100 | 129.6 |
| +syntax | 126.1 |
| +topic | 129.3 |
| +syntax+topic | 123.1 |

As shown in Table 4.6, adding syntactic information (“+syntax”) reduced the perplexity of the basic RFLM (“RF-100”), which was already significantly lower than that of the n -gram LM (“KN.3gm”). Interestingly, adding topic information on top of syntactic information (“+syntax+topic”) further reduced the perplexity (“+syntax”) even though adding topic information alone (“+topic”) did not. Syntactic information and topic information worked in a synergetic way, as we had hoped. Compared to the perplexity results in Roark (2001), which were repeated here, the RFLM with syntactic information (“+syntax”), even without interpolation with the trigram LM, was as good as their model, which interpolates the LM from the parser and the trigram LM (“interp. KN.3gm”).

We evaluated our models on the Wall Street Journal DARPA’93 HUB1 task, whose test setup consisted of 213 utterances with a total of 3,446 words. The vocabulary size was 20k and N -best lists with at most 50 per utterance were generated using a standard 3-gram

CHAPTER 4. KNOWLEDGE INTEGRATION WITH RFLMS

model trained on 40M words of Wall Street Journal text. The results are shown in Table 4.7.

Table 4.7: *N*-best rescoring with syntactic and topic features

| Model | WER |
|---------------|------|
| KN.3gm | 15.8 |
| RF-100 | 15.3 |
| +syntax | 15.1 |
| +topic | 15.5 |
| +syntax+topic | 15.1 |

We observed that the WER results basically followed the trend of the perplexity results. However, the synergy between syntactic and topic information did not show up in terms of overall WER. Following Khudanpur and Wu (2000), we computed the WERs on content words and stop words, as defined by a standard list of stop words for information retrieval, and tabulated the results in Table 4.8.

Table 4.8: Word error rate breakdown into content and stop words

| Model | All Words | Content Words | Stop Words |
|---------------|-----------|---------------|------------|
| KN.3gm | 15.8 | 14.7 | 16.9 |
| RF-100 | 15.3 | 14.3 | 16.4 |
| +syntax | 15.1 | 14.2 | 16.0 |
| +topic | 15.5 | 14.2 | 16.8 |
| +syntax+topic | 15.1 | 14.1 | 16.2 |

As shown in Table 4.8, the synergy between syntax and topic did show up in terms of content word error rate, albeit small.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

We have presented a novel method of knowledge integration into LMs using the RFLM. First we introduced the model itself and conducted several revealing experiments to further our understanding about the RFLM. Then we proposed an efficient disk-swapping algorithm to solve the space complexity problem of the original training method, expanding the utility of the model. Finally we presented our proposal of integrating various knowledge sources into the LM by using the RFLM as a general framework and exhibited this central idea in three innovative applications: modeling morphologically rich languages, exploiting prosodic breaks in LMs and combining syntax and semantics in LMs.

5.2 Future Work

It is rather surprising how well a simple-minded model like the n -gram LM can do, given a huge amount of data. A recent study even suggests that smoothing becomes unimportant in case of huge data availability (Brants, Popat, Xu, Och, and Dean, 2007). However, not every language and/or domain is blessed with a huge amount of data. For example, there were less than 200K words of training data available when Kirchhoff, Vergyri, Bilmes, Duh, and Stolcke (2006) built a speech recognizer for Egyptian Colloquial Arabic. Therefore LM adaptation is becoming more important. By comparing and contrasting the decision trees learned from different domains, the RFLM might be able to transfer some useful domain-independent information from one domain to another. For example, it might be reasonable to assume that the decision tree structures learned from a background domain with abundant data are also useful in the adaptation domain, where sufficient data are not available to learn the tree structures. Then we can pour down the data from the adaptation domain through those trees to get the adapted model. Similar languages or dialects of a language can also be treated this way.

Although the independence of member decision trees makes the RFLM ideal for parallel computing, it is interesting to find out whether those trees can be trained in a successive way so that the next tree would complement previous trees' effort. This is exactly the idea of boosting (Freund and Schapire, 1995; Friedman, Hastie, and Tibshirani, 2000). A nice framework of boosting in density estimation has already been laid out in Rosset and Segal (2002).

Bibliography

Mohamed Afify, Sherif Abdou, Frank Guo, John Makhoul, Long Nguyen, and Bing Xiang.

The BBN non-English evaluation systems for broadcast news. In *Proceedings of the RT-04 workshop*, Palisades, NY, November 2004.

Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees.

Neural Computation, 9(7):1545–1588, 1997.

Lalit R. Bahl, Peter F. Brown, Peter V. deSouza, and Robert L. Mercer. A tree-based

statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):1001–1008, 1989.

Michele Banko and Eric Brill. Mitigating the paucity-of-data problem: Exploring the effect

of training corpus size on classifier performance for natural language processing. In *Proceedings of the Conference on Human Language Technology*, 2001.

Jerome R. Bellegarda. Exploiting latent semantic information in statistical languagemod-

eling. *Proceedings of IEEE*, 88(8):1279–1296, 2000.

BIBLIOGRAPHY

- Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *Proceedings of Advances in Neural Information Processing Systems*, 2001.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.
- Thorsten Brants and Alex Franz. Web 1T 5-gram Version 1. Linguistic Data Consortium, Philadelphia, 2006.
- Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, 2007.
- Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.
- William Byrne, Jan Hajic, Pavel Ircing, Frederick Jelinek, Sanjeev Khudanpur, Pavel Krbec, and Josef Psutka. On large vocabulary continuous speech recognition of highly inflectional language – Czech. In *Proceedings of EUROSPEECH 2001: 7th European Conference on Speech Communication and Technology*, volume 1, pages 487–489, 2001.

BIBLIOGRAPHY

- Eugene Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131, Morristown, NJ, USA, 2001.
- Ciprian Chelba and Frederick Jelinek. Structured language modeling. *Computer Speech and Language*, 14(4):283–332, 2000.
- Ken Chen, Sarah Borys, Mark Hasegawa-Johnson, and Jennifer Cole. Prosody dependent speech recognition with explicit duration modelling at intonational phrase boundaries. In *Proceedings of 8th European Conference on Speech Communication and Technology (EUROSPEECH 2003 - INTERSPEECH 2003)*, pages 393–396, 2003.
- Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.
- Stanley F. Chen, Kristie Seymore, and Roni Rosenfeld. Topic adaptation for language modeling using unnormalized exponential models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 681–684, Seattle, Washington, 1998.
- Ghinwa Choueiter, Daniel Povey, Stanley F. Chen, and Geoffrey Zweig. Morpheme-based language modeling for Arabic LVCSR. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 14–19, 2006.
- Christopher Cieri, David Miller, and Kevin Walker. The Fisher corpus: a resource for the

BIBLIOGRAPHY

- next generation of speech-to-text. In *Proceedings of 4th International Conference on Language Resources and Evaluation*, pages 69–71, Lisbon, Portugal, May 2004.
- Jordan Cohen, Terri Kamm, and Andreas G. Andreou. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. *Journal of the Acoustic Society of America*, 97(5):3246–3247, May 1995.
- Jia Cui, Yi Su, Keith Hall, and Frederick Jelinek. Investigating linguistic knowledge in a maximum entropy token-based language model. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, Kyoto, Japan, December 2007.
- Kareem Darwish. Building a shallow arabic morphological analyzer in one day. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–8, Morristown, NJ, USA, 2002.
- Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.
- Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automatic tagging of Arabic text: from raw text to base phrase chunks. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, 2004.
- Markus Dreyer and Izhak Shafran. Exploiting prosody for PCFGs with latent annotations.

BIBLIOGRAPHY

- In *Proceedings of INTERSPEECH 2007: 8th Annual Conference of the International Speech Communication Association*, 2007.
- Ahmad Emami and Frederick Jelinek. Random clusterings for language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 581–584, March 2005a.
- Ahmad Emami and Frederick Jelinek. A neural syntactic language model. *Machine Learning*, 60(1-3):195–227, 2005b.
- Radu Florian and David Yarowsky. Dynamic nonlocal language modeling via hierarchical topic-based adaptation. In *Proceeding of 37th Annual Meeting of the Association for Computational Linguistics*, 1999.
- John Eric Fosler-Lussier. *Dynamic Pronunciation Models for Automatic Speech Recognition*. PhD thesis, University of California, Berkeley, CA, USA, 1999.
- Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, 1995.
- Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28, 2000.
- Mark J.F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12:75–98, 1998.

BIBLIOGRAPHY

- Daniel Gildea and Thomas Hofmann. Topic-based language models using EM. In *Proceedings of European Conference on Speech Communication and Technology*, pages 2167–2170, Budapest, Hungary, September 1999.
- Irving J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.
- Nizar Habash. Introduction to Arabic natural language processing. <http://www1.ccls.columbia.edu/~cadim/TUTORIAL.ARABIC.NLP.pdf>, 2005.
- John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Mathew Lease, Yang Liu, Brian Roark, Mathew Snover, and Robin Stewart. PCFGs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL)*, pages 161–168, 2006.
- Hynek Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.
- Keikichi Hirose, Nobuaki Minematsu, and Makoto Terao. Statistical language modeling with prosodic boundaries and its use for continuous speech recognition. In *Proceedings of 7th International Conference on Spoken Language Processing (ICSLP2002 - INTER-SPEECH 2002)*, 2002.

BIBLIOGRAPHY

- Julia Hirschberg and Christine H. Nakatani. Acoustic indicators of topic segmentation. In *Proceedings of the International Conference on Spoken Language Processing*, 1998.
- Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- Rukmini Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39, 1999.
- Frederick Jelinek and Robert L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceeding of the Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, 1980.
- Sanjeev Khudanpur and Jun Wu. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in lanugage modeling. *Computer Speech and Language*, 14(4):355–372, 2000.
- Katrin Kirchhoff, Dimitra Vergyri, Jeff Bilmes, Kevin Duh, and Andreas Stolcke. Morphology-based language modeling for conversational arabic speech recognition. *Computer Speech and Language*, 20(4):589–608, 2006.
- Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184, 1995.

BIBLIOGRAPHY

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2): 313–330, 1993. ISSN 0891-2017.

Sven Martin, Jörg Liermann, and Hermann Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37, 1998.

Hermann Ney, Ute Essen, and Reinhard Kneser. On the estimation of ‘small’ probabilities by leaving-one-out. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(12):1202–1212, 1995.

Mari Ostendorf, Izhak Shafran, Stefanie Shattuck-Hufnagel, Leslie Carmichael, and William Byrne. A prosodically labeled database of spontaneous speech. In *Proceedings of ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, pages 119–121, 2001.

Mari Ostendorf, Izhak Shafran, and Rebecca Bates. Prosody models for conversational speech recognition. In *Proceedings of the 2nd Plenary Meeting and Symposium on Prosody and Speech Processing*, pages 147–154, 2003.

Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, 1998.

BIBLIOGRAPHY

- Gerasimos Potamianos and Frederick Jelinek. A study of n-gram and decision tree letter language modeling methods. *Speech Communication*, 24(3):171–192, 1998.
- Daniel Povey, Brian Kinsbury, Lidia Mangu, George Saon, Hagen Soltau, and Geoffrey Zweig. fMPE: Discriminatively trained features for speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 961–964, 2005.
- Jorma Rissanen and Glen G. Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, January 1981.
- Brian Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001.
- Roni Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10:187–228, 1996.
- Roni Rosenfeld. A whole sentence maximum entropy language model. In *Proc. IEEE workshop on Automatic Speech Recognition and Understanding*, Santa Barbara, California, December 1997.
- Saharon Rosset and Eran Segal. Boosting density estimation. In *Advances in Neural Information Processing Systems*, 2002.
- Holger Schwenk. Continuous space language models. *Computer Speech and Language*, 21(3):492–518, 2007.

BIBLIOGRAPHY

Izhak Shafran and Mari Ostendorf. Acoustic model clustering based on syllable structure.

Computer Speech and Language, 17(4):311–328, 2003.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical*

Journal, 27:379–423, 623–656, 1948.

Kim Silverman, Mary Beckman, John Pitrelli, Mari Ostendorf, Colin Wightman, Patti

Price, Janet Pierrehumbert, and Julia Hirschberg. ToBI: A standard for labeling english prosody. In *Proceedings of the International Conference on Spoken Language*

Processing, pages 867–870, 1992.

Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Mari Ostendorf, Dilek Hakkani,

Madelaine Plauche, Gökhan Tür, and Yu Lu. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proceedings of the International*

Conference on Spoken Language Processing, volume 5, pages 2247–2250, 1998.

Andreas Stolcke, Elizabeth Shriberg, Dilek Hakkani-Tür, and Gökhan Tür. Modeling the

prosody of hidden events for improved word recognition. In *Proceedings of European*
Conference on Speech Communication and Technology, 1999.

Yi Su and Frederick Jelinek. Exploiting prosodic breaks in language modeling with random

forests. In *Proceedings of 4th Conference on Speech Prosody*, pages 91–94, Campinas, Brazil, May 2008.

Yi Su, Frederick Jelinek, and Sanjeev Khudanpur. Large-scale random forest language

BIBLIOGRAPHY

- models for speech recognition. In *Proceedings of INTERSPEECH 2007: 8th Annual Conference of the International Speech Communication Association*, volume 1, pages 598–601, Antwerp, Belgium, 2007.
- Yee Whye Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, 2006.
- Shaojun Wang, Shaomin Wang, Russell Greiner, Dale Schuurmans, and Li Cheng. Exploiting syntactic, semantic and lexical regularities in language modeling via directed markov random fields. In *Proceedings of the International Conference on Machine Learning*, pages 948–955, Bonn, Germany, 2005.
- Wen Wang and Mary P. Harper. The superary language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 238–247, Morristown, NJ, USA, 2002.
- Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, July 1991.
- Jun Wu and Sanjeev Khudanpur. Efficient training methods for maximum entropy lan-

BIBLIOGRAPHY

- guage modeling. In *Proceedings of Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, Beijing, China, October 2000.
- Peng Xu. *Random forests and the data sparseness problem in language modeling*. PhD thesis, Johns Hopkins University, 2005.
- Peng Xu and Frederick Jelinek. Random forests in language modeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 325–332, Barcelona, Spain, 2004a. Association for Computational Linguistics.
- Peng Xu and Frederick Jelinek. Using random forests in the structured language model. In *Advances in Neural Information Processing Systems*, Vancouver, December 2004b.
- Peng Xu and Lidia Mangu. Using random forest language models in the IBM RT-04 CTS system. In *Proceedings of INTERSPEECH 2005 - Eurospeech: 9th European Conference on Speech Communication and Technology*, pages 741–744, 2005.

Vita

Yi Su was born in Nanxi, Sichuan, People's Republic of China on January 31, 1979. He received a Bachelor of Engineering degree in Computer Science and Technology from Tsinghua University, Beijing in 2000 and enrolled in the Electrical and Computer Engineering Ph.D. program with the Center for Language and Speech Processing at The Johns Hopkins University in 2002. In 2004, he received a Master of Science in Engineering degree in Electrical and Computer Engineering from The Johns Hopkins University.

He participated in the Center for Language and Speech Processing Summer Workshop in 2004 and interned with the Natural Language Processing Group, Microsoft Research, Redmond in summer 2008.

He is interested in statistical modeling of human intelligence as exemplified in the use of language and speech.