STRATEGIES FOR HANDLING OUT-OF-VOCABULARY WORDS IN AUTOMATIC SPEECH RECOGNITION

by

Xiaohui Zhang

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

Baltimore, Maryland
October, 2019

Abstract

Nowadays, most ASR (automatic speech recognition) systems deployed in industry are closed-vocabulary systems, meaning we have a limited vocabulary of words the system can recognize, and where pronunciations are provided to the system. Words out of this vocabulary are called out-of-vocabulary (OOV) words, for which either pronunciations or both spellings and pronunciations are not known to the system. The basic motivations of developing strategies to handle OOV words are: First, in the training phase, missing or wrong pronunciations of words in training data results in poor acoustic models. Second, in the test phase, words out of the vocabulary cannot be recognized at all, and mis-recognition of OOV words may affect recognition performance of its in-vocabulary neighbors as well. Therefore, this dissertation is dedicated to exploring strategies of handling OOV words in closed-vocabulary ASR.

First, we investigate dealing with OOV words in ASR training data, by introducing an acoustic-data driven pronunciation learning framework using a likelihood-reduction based criterion for selecting pronunciation candidates from multiple sources, i.e. standard grapheme-to-phoneme algorithms (G2P) and phonetic decoding, in a greedy fashion. This framework effectively expands a small hand-crafted pronunciation lexicon to cover OOV words, for which the learned pronunciations have higher quality than approaches using G2P alone or using other baseline pruning criteria. Furthermore, applying the proposed framework to generate alternative pronunciations for in-vocabulary (IV) words improves both recognition performance on relevant words and overall acoustic model performance.

Second, we investigate dealing with OOV words in ASR test data, i.e. OOV detection and recovery. We first conduct a comparative study of a hybrid lexical model (HLM) approach for OOV detection, and several baseline approaches, with the conclusion that the HLM approach outperforms others in both OOV detection and first pass OOV recovery performance. Next, we introduce a grammar-decoding framework for efficient second pass OOV recovery, showing that with properly designed schemes of estimating OOV unigram probabilities, the framework significantly improves OOV recovery and overall decoding performance compared to first pass decoding.

Finally we propose an open-vocabulary word-level recurrent neural network language model (RNNLM) re-scoring framework, making it possible to re-score lattices containing recovered OOVs using a single word-level RNNLM, that was ignorant of OOVs when it was trained. Above all, the whole OOV recovery pipeline shows the potential of a highly efficient open-vocabulary word-level ASR decoding framework, tightly integrated into a standard WFST decoding pipeline.

First Reader: Sanjeev Khudanpur

Second Reader: Daniel Povey

Acknowledgments

To my advisor Sanjeev Khudanpur, I am deeply thankful to you for giving me the opportunity to do research at CLSP at Hopkins, and for your constant support throughout my PhD, especially when I was ill in 2016. I will always remember your teaching (I really enjoyed the classes "Information Theory" and "Information Extraction" which you taught!), mentorship, and especially all your insights on proposing smart and proper metrics to evaluate research ideas. To my advisor Daniel Povey, it's you who trained me literally from scratch in ASR, with great patience. Thanks for the countless hours of inspiring discussions and debugging. Thanks for giving me the opportunity to contribute to Kaldi. Thank you for teaching me to enjoy the beauty of coding and drive for engineering excellence, by always implementing the highest standard on code review. Thanks for always replying emails in seconds. Wherever you are, I will cherish the your guidance throughout the 7 years, at JHU and Facebook. To Najim Dehak, Shinji Watanabe and Hynek Hermansky, I would like to thank you for kindly serving on several of my committees in the past few years. Especially, I want to thank Hynek for your leadership at the CLSP, Najim for so many memorable parties at your home, and Shinji for your friendship and emotional support. To Yenda Trmal, Mahsa Yarmohamadi, and Paola Garcia for your support throughout multiple DARPA and IARPA projects, and especially Yenda on our memorable collaboration on IARPA BABEL. Thanks to Professors Johhn Wierman and Carey Priebe at JHU Applied Math and Stats (AMS) department, for teaching important math and statistics knowledge during my study at AMS, which was really helpful for my research. To Ruth Scally, Debbie Race, Dana Walter-Shock and Belinda Blinkoff for the substantial administrative support in the past few years, especially on scheduling the defense.

Thanks to Puyang Xu, Guoguo Chen, Vijay Peddinti, Yuan Cao, Shuai Huang, Xuchen Yao, Harrish Mallidi, Ming Sun for your help as my senior labmates, especially Guoguo and Puyang, whose advice and help were really crucial for me to survive and being fruitful during the early years of my PhD. Thanks to my peers Vimal Manohar, Chunxi Liu, Matthew Wiesner, David Snyder, Yiming Wang, Pegah Ghahrmani, Hang Lyu, Ashish Arora, Hongyuan Mei, Ke Li, Hainan Xu, Keith Kintzley, Zhehuai Chen, Ruizhe Huang, Dongji Gao, Yiwen Shao, Dingquan Wang and many other JHU friends and colleagues, for all the thoughtful advice, discussion, collaboration, and great company along the way. Thanks to Zhouyang Zhang for the amazingly fun time playing duets together. Thanks to Shangsi Wang and Yumo Wang for being my wonderful roommates for 5 years (and Lei Zhang for the 6th year), especially I thank Shangsi for all your help on math and cooking, and Yumo for all the fun time together, and co-parenting our cute cat Leo. Thanks to Qi Huang for the friendship and kindly providing living space every time I went back to Baltimore. Thanks to Mo Yu and Quan Jiao for all the fun time playing ping-pong together. Thanks to Dr. David Cooper and Dr. Jason Prescott from Hopkins Hospital, for treating my thyroid cancer properly. Your superb treatment ensured my speedy recovery, which was crucial for me to continue my PhD.

Thanks to my dear brothers and sisters from JHU-AFC and Hengtong Christian Community Church: Faithful Liao and Uncle. Ji, Ke-yao Wang, Lingling Guo, Yu Du, Shuyuan Wu, Mingming Wu, Brian Hu, Yuanyuan Liu, Meng and Garland Rowland, TJ and Shan-Jen Huang, Paul and Esther Lin, Joseph Lin, Ivy Peng, Guorong Chen, Liqin Zhou, Hanzhi Lin, Yuzhu Wang, John Yu, Xiaolei Liang, Yichuan Tang, Yang Du and many others. I'll always remember your spiritual supports and so many memorable times serving Lord together. Especially I want to thank Faithful Liao, Paul and Esther for your spiritual guidance and lovingly cares in my life. Also thanks to my American brothers and sisters from Grace Community Church, JHU-CRU and JHIF: Dave & Nancy Barlow and Steve & Molly Smith, BJ Hall, Peter & Mimi Hinterkopf, Bryan Nichols and many others, for your abundant love, encouragements and supports to me and my wife.

Thanks to friends I met during my internship at Google: Hang Ma, Tiantian Gao, Lei Tian, Kuan Liu, William Chan, Derek Murray and Daphne Ippolito, for all the fun time together in California, and especially William for your spiritual encouragements. Thanks to Geoffrey Zweig and Yatharth Saraf, for supporting me at Facebook NYC. I feel lucky to have the opportunity to apply what I learned during my PhD study, to real production. Besides, I'd like to sincerely thank my undergraduate professors at BJUT: Dingfang Zeng, Lifang

Wu, Guangmin Sun, Tieliu Wang, Shudong Zhao and Yijian Jiang, who laid a good foundation for me to pursue my PhD in US.

Thanks so much for my CKW (cute and kind wife) Yujiang Chen, for keeping me productive throughout my PhD, by her sincere supports via kind words and prayers, amazing artworks, cooking etc., and also by not meeting me until the 4th year of my PhD. Thanks so much for my dear parents Lianshou Zhang and Huichun Xu who gave me life. Thanks for enthusiastically and constantly loving me, nurturing me and supporting me until today. All my successes were impossible without you. Last but not least, I want to thank my Lord Jesus. Thanks for gracefully saving me from sin, and sending so many wonderful people in my life, to support me, educate me and help me grow. I want to dedicate my whole life to you.

Dedication

This thesis is dedicated to my wife Yujiang, and my parents Lianshou and Huichun.

Table of Contents

List of Tables

List of Figures

List of Acronyms

G2P	Grapheme-to-Phoneme
ASR	Automatic Speech Recognition
AM	Acoustic Model
LM	Language Model
BIC	Bayesian Information Criterion
SAT	Speaker Adaptive Training
TDNN	Time Delay Neural Networks
LF-MMI	Lattice-free Maximum Mutual Information
PD	Phonetic Decoding
pp	Pronunciation Probability
TMR	Token Miss Rate
NN	Neural Network
DARPA	Defense Advanced Research Projects Agency

EM	Expectation-Maximization
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IARPA	Intelligence Advanced Research Projects Activity
IV	In-Vocabulary
KWS	Keyword Search
LDC	Linguistic Data Consortium
LSTM	Long Short-Term Memory
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel-Frequency Cepstral Coefficient
NIST	National Institute of Standards and Technology
OOV	Out-of-Vocabulary
SGD	Stochastic Gradient Descent
WER	Word Error Rate
WFST	Weighted Finite State Transducer
E2E	End-to-End
P2G	Phoneme-to-Grapheme
HLM	Hybrid Lexical Model

Chapter 1

Introduction

An automatic speech recognition (ASR) system takes an audio stream as input and turns it into a text transcription, which is a sequence of words. It usually comprises of three major components: an acoustic model (AM), a pronunciation lexicon, and a language model (LM). Here are some concepts which are helpful to understand the roles of these components, especially the lexicon:

- Phone: the smallest pre-defined discrete segment of sound in a stream of speech, e.g. `p`, `b`, `d`, and `t` in the English words **pad**, **pat**, **bad**, and **bat**.
- Grapheme: In linguistics, a grapheme is the smallest unit of a writing system of any given language. In English it's a letter (a to z).
- Pronunciation: the phonetic transcription of a word, represented by a sequence of phones[1].
- Acoustic model: a statistical model which represents the relationship

[1] Pronunciation of a word could also be represented by larger sub-word units, like syllables. In this work we only consider phones.

between acoustic sounds of a language and the underlying sequence of phones.

- Language model: a probability distribution over sequences of words drawn from the vocabulary of a language, independent of the acoustics.

- Lexicon: a pronunciation dictionary (usually hand-crafted), which lists the pronunciation(s) of each word (e.g. **speech** `s p iy ch`) in the vocabulary. Each word could have more than one pronunciation variant. The vocabulary covered by the lexicon is the same as the vocabulary used in the language model. Sample lexicon entries are as follows:

  ...

speech	`s p iy ch`
speechless	`s p iy ch l ih s`
speed	`s p iy d`

  ...

- Grapheme-to-Phoneme (G2P): the process of converting a letter string (word) like "cake" into a phone string (pronunciation) like `k ey k`, based on statistical modeling. The probabilistic relationships between words and pronunciations are learned from an existing lexicon.

In a conventional hybrid ASR system, a lexicon with a fixed vocabulary is always used in the training/test phase (see Figure 1.1 and Figure). This is called closed-vocabulary ASR. i.e. only words within the vocabulary of the lexicon can be recognized in test phase. Recent research progress on grapheme-based end-to-end (E2E) models has shown that it's possible to remove the need for a separate expert-curated pronunciation lexicon [3], enabling open-vocabulary ASR. However the relative performance of grapheme-based ASR over conventional phone-based ASR is highly language-dependent, i.e. for irregularly spelled languages grapheme ASR may only work well when we have a large amount of data. Taking English for an example: given large amount of training data (e.g. more than $10,000$ hours), graphemic systems performs better than phonemic systems in terms of Word Error Rate (WER) as reported in [3]. But given only hundreds of hours of data (e.g. in academic datasets like Switchboard/WSJ/Librispeech), it has been consistently observed that using a lexicon gives better performance[4, 5, 6]. Therefore, based on these observations, it is important to address the OOV problem for closed-vocabulary hybrid ASR.

In the ideal case, in the training phase for closed-vocabulary ASR, a hand-crafted lexicon should cover all words in speech transcripts being used to train the acoustic model, i.e. we should know how to pronounce all words in the acoustic training data. The reason is that, we align acoustic data to phoneme sequences to get "alignments" which are training examples for the acoustic phonetic models. Missing or wrong pronunciations of words in training data results in poor alignments and contaminates the training examples, thereby affecting acoustic modeling performance. In the test phase, the lexicon should cover all the words in test data, so that the ASR at least has the potential to recognize all the words being spoken.

However, in practice, in training phase, the pronunciation lexicon doesn't necessarily cover all the words in the transcripts, and are Out-of-Vocabulary

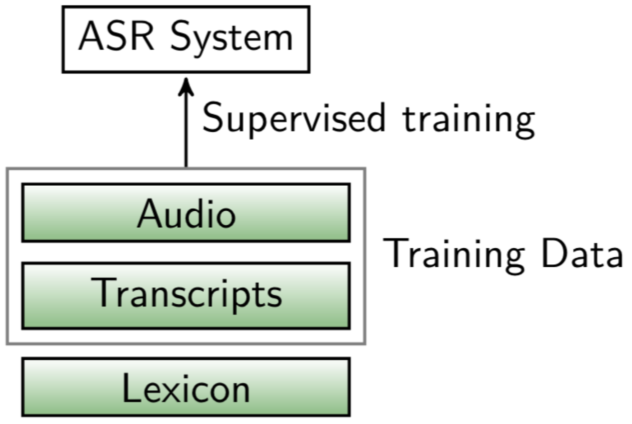


Figure 1.1: ASR framework –training phase

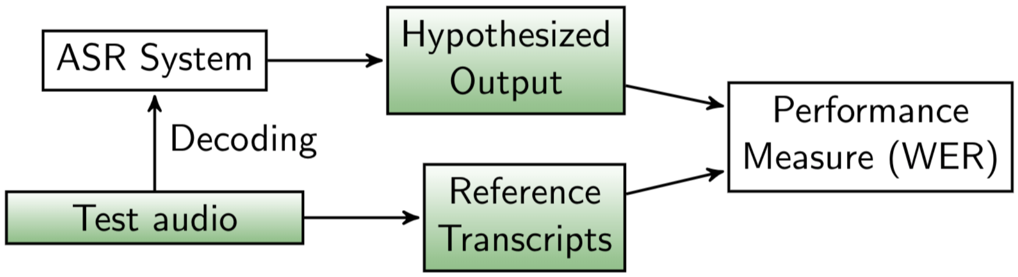


Figure 1.2: ASR framework –test phase

(OOV) words, or the pronunciations provided by the expert lexicon don't agree well with the acoustics because of mismatched domain/accents. In the test phase, there could also be lots of words that cannot be anticipated while building the ASR system, and mis-recognition of these OOV words will affect recognition performance of its in-vocabulary neighbors. Both of these are big challenges for closed-vocabulary ASR, and this dissertation is dedicated to investigating some novel strategies to tackle the OOV challenges.

1.1 Problem Statement

Generally speaking, the OOV word problems in closed-vocabulary ASR fall into two categories: Handling OOV words in training and test phases.

1.1.1 Handling OOV words in ASR training phase

In closed-vocabulary ASR, the quality of lexicon affects acoustic modeling performance. However, building a large expert (hand-crafted) lexicon is expensive. Therefore, we are interested in how to expand a small expert lexicon to cover pronunciations of OOV words, i.e. learning OOVs' pronunciations from their written forms and/or acoustics examples. And we call this problem **pronunciation learning**. The main challenge here is: the quality of the learned pronunciations is crucial to:

— The training of the acoustic model (learning the correct probabilistic relationship between acoustics and phonemes)[2].

— The capability to correctly recognize these words in test data.

1.1.2 Handling OOV words in ASR test phase

In closed-vocabulary ASR, OOV words in test data can't be recognized, and they affect the recognition performance of their surrounding in-vocabulary (IV) words. In the following example, the phrase in reference text "as AIRCOA", where AIRCOA is an OOV word, was wrongly recognized as "a circle"

[2]In some sense, the training-time problem should be better described as an Out-of-Lexicon (OOL) problem, since we know the spelling of those words. But we'll continue to call it an OOV problem to remain consistent with extant literature

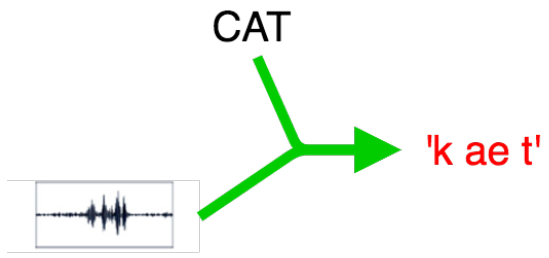


Figure 1.3: Dealing with OOVs in ASR training phase — lexicon learning

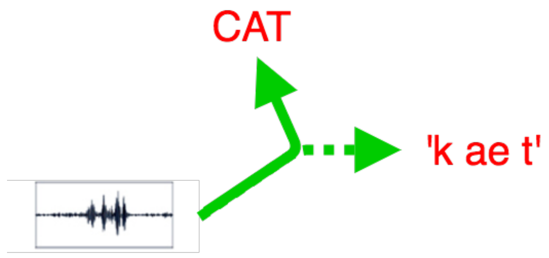


Figure 1.4: Dealing with OOVs in ASR test phase — OOV detection & recovery

in hypothesis text, i.e. even though only "AIRCOA" is OOV, its IV neighbor "as" was wrongly recognized as "a".

Ref: ... *associated inns known as AIRCOA* ...

Hyp: ...*associated inns known a circle*...

Therefore, the problem here is **OOV detection and recovery** in hypothesis text. In many cases, this issue is of particular interest, since many OOVs are proper nouns, which could be very important for the downstream task.

Figure 1.3 summarizes the two problems in an intuitive way.

1.2 Contributions

This dissertation makes the following contribution to the body of knowledge in OOV handling for ASR systems.

1.2.1 Lexicon learning

- We propose a novel acoustic data-driven framework for pronunciation learning. It uses a likelihood-based pruning criterion modeling confusability between pronunciations in an efficient greedy framework for pronunciation selection. The framework is computationally scalable (no iterative lattice generation) and easy to parallelize (word level EM procedure for pronunciation selection).

- We evaluate the proposed pronunciation learning framework in various **lexicon expansion** experiments, and show that:

  — Using acoustic information in pronunciation candidate generation is important (compared to only using orthographic and lexicon information).

  — In terms of pronunciation candidate selection/pruning strategy, the proposed framework performs better than G2P/pronunciation probabilities (pp)/Bayesian information criterion (BIC) based criteria, in both WER and lexicon size.

  — Evaluation on individual words recognition performance reveals that the performance gain brought by our method increases as we have more training acoustic examples.

- We evaluate the proposed pronunciation learning framework in various **lexicon adaptation** experiments, and show that:

  — Adapting the pronunciation of of words to acoustic evidence helps with overall acoustic model (AM) training performance.

— It also improves recognition performance on individual words just by re-decoding with the adapted lexicon.

1.2.2 OOV detection and recovery

- We show that our implementation of a hybrid lexical model (HLM) approach for OOV detection outperforms baselines (hybrid LM+classification, single-phone) in both OOV detection and first pass OOV recovery.

- We investigate a Grammar-decoding framework for efficient second pass decoding. We show that it significantly improves OOV recovery/overall decoding performance compared to first pass decoding, while avoiding re-compiling the whole decoding graph.

- We empirically investigate different schemes of estimating OOV LM probabilities, with findings: Phonemic/Character LM scores give the best performance; Incorporating empirical frequency helps when the OOV rate is high.

- We propose an open-vocabulary word RNNLM re-scoring framework, enabling N-best/lattice re-scoring on top of lattices containing recovered OOVs with a single word RNNLM ignorant of OOVs when it was trained. Experiments have shown that it improves both OOV recovery and overall decoding performance v.s. decoding with N-gram LMs.

- We explore potential performance gain when given oracle OOV spellings, and show that HLM can bring further gain by learning OOV pronunciations from acoustic evidence.

- Using the whole proposed pipeline (HLM + grammar decoding + open-vocab RNNLM rescoring), we show the potential of a highly efficient open-vocabulary word-level ASR decoding framework, tightly integrated into a standard WFST decoding pipeline.

1.3 Outline

The rest of this dissertation is organized as follows. In Chapter 2, we first discuss motivation and related work on acoustic data-driven lexicon learning in Section 2.1. Then we discuss how we generate pronunciation candidates in Section 2.2, and how we collect acoustic evidence from training data in Section 2.3. Then we introduce our likelihood-based pronunciation pruning strategy and the greedy pruning framework in Section 2.4. Experimental results on various ASR lexicon expansion and lexicon adaptation tasks, and analysis of the number of acoustic examples versus performance of lexicon learning are presented in Section 2.5. We conclude in Section 2.6.

In Chapter 3, we first discuss motivation and related work on OOV detection and recovery in Section 3.1. Then we revisit hybrid lexical model (HLM) based OOV detection, candidate generation and first pass recovery, and introduce our implementation in Section 3.2. Experimental results comparing of the HLM approach for OOV detection and the IBM (hybrid LM + classification) and single-phone-based OOV detection are also presented in Section 3.2. Then, we introduce a grammar-decoding framework for efficient second pass decoding, and an open-vocabulary word RNNLM re-scoring framework for re-scoring lattices containing recovered OOVs in Section 3.3. We also discuss different schemes of estimating OOV unigram probabilities there. In Section 3.3, we report experiments on Spanish, and read and conversational English to study the performance of OOV recovery and overall decoding of first pass, second pass (grammar) decoding and open-vocab RNNLM rescoring. We summarize the results in Section 3.4.

In Chapter 4, we summarize all the developed components of this dissertation and discuss potential directions for future work.

Chapter 2

Handling OOVs in Training Time

This chapter is about handling OOVs in training time, i.e. lexicon learning. In the past few years there has been an growing interest in investigating acoustic data-driven lexicon learning for ASR systems. "acoustic data-driven lexicon learning" is a broad concept. Generally, in the ASR context[1], it means automatically obtaining pronunciations of words for which pronunciations are not available or not good enough, but for which transcribed acoustic data exists, from which we can learn pronunciations (hence "acoustic data-driven").

There are two goals we want to achieve by learning pronunciations from acoustic data:

- When training an AM, the quality of alignments partially relies on quality of pronunciations of words. So improving pronunciations will improve performance of acoustic modeling.
- When decoding, the recognition performance of relevant words (with

[1] Acoustic data-driven lexicon learning has been investigated in more specific tasks like name recognition [7] [8]. What we are interested in is the more general Large Vocabulary Continuous Speech Recognition (LVCSR) setting.

improved pronunciations) from test data will be improved.

The following are the specified statements of the two scenarios of lexicon learning we are interested in:

Learning pronunciations of Out-of-Vocabulary (OOV) words with limited lexicon resources. In order to develop ASR systems under limited lexicon resources, one solution is to adopt a graphemic lexicon [9] [10] or acoustic unit discovery methods [11] [12][2], which totally eliminate the expert efforts for developing a phonetic pronunciation lexicon. In real applications, especially in languages like English and Chinese, however, a more common scenario is that we already have a phonetic inventory, and a small expert lexicon for a specific language. Our work focuses on this case, i.e. given a small expert lexicon, we want to derive pronunciations for Out-of-Vocabulary (OOV) words, for which we know their spelling and have acoustic examples. We call this task "lexicon expansion".

Adapting pronunciations of In-Vocabulary (IV) words given acoustic data Suppose we have an expert (reference) lexicon following an existing convention (e.g. CMUDict), where some words have wrong pronunciations (e.g. **FDA** : 'f d a' from the `Cantab` lexicon of Tedlium corpus), or pronunciations mismatched with the given acoustic data, because of acoustic variants like accents. Letting a human find these words from the lexicon and correct their pronunciations causes too much efforts. Given acoustic examples of these words,

[2]The main reason it hasn't been widely used is that acoustic unit discovery doesn't lead to very good ASR performance

we want to adopt lexicon learning process to learn better pronunciations of these words. We call this task "lexicon adaptation". Proposed modification of the reference lexicon will be presented in a format as an "lexicon edit" file, allowing linguists to make modification on the pronunciations or change the accept/reject decisions.

2.1 Motivation and existing methods

In the LVCSR setting, in order to develop ASR systems under limited lexicon resources, one solution is to adopt a graphemic lexicon [9, 10] or acoustic unit discovery methods [11, 12], which totally eliminate the expert efforts for developing a phonetic pronunciation lexicon. In real applications, however, a more common scenario is that we already have a phonetic inventory, and a small expert lexicon for a specific language. Given a small expert lexicon, the most straightforward way to generate pronunciation candidates for OOV words is to train a Grapheme-to-Phoneme (G2P)[13] model using the seed lexicon and apply it on these OOV words [14, 15, 16]. But G2P cannot guarantee giving satisfying pronunciation variants if the language is not very phonemic (meaning words are not regularly spelled), like English. Also for proper nouns and abbreviations G2P usually cannot give correct pronunciations, because they are relatively rare in the G2P training examples, and their pronunciations are usually inconsistent with linguistic rules. That is the reason why people have found it helpful to utilize acoustic examples besides conventional G2P methods, in order to improve lexicon learning. Given there are various kinds of attempts in incorporating acoustic data into lexicon learning in different aspects, e.g. acoustic data can be used to generate pronunciation variants or/and estimate weights of the G2P/human-generated pronunciation variants, our basic motivation is to investigate the role of acoustic data in existing lexicon learning methods, and propose new ways to efficiently incorporate acoustic data into lexicon learning.

Generally, there are two types of acoustic-data driven lexicon learning approaches:

Modeling the acoustics and grapheme-to-phoneme relationship jointly In these type of approaches, acoustics are incorporated into the training phase of a grapheme-to-phoneme model [17] [7] and a seed lexicon is not needed,.e.g. training a joint model of acoustics and grapheme-phoneme relationships (phonemes are treated as hidden variables), by maximizing the joint/conditional likelihood of the words and the acoustic examples [7]. This model could be interpolated with a conventional G2P model trained using the seed lexicon alone, or we can use this model to generate pronunciation variants for words with acoustic examples, and we combine these word-pronunciation pairs with the seed lexicon to train a conventional G2P model, or we can simply combine pronunciations generated from this model with pronunciations given by a conventional G2P model trained on a seed lexicon [17]. This type of approaches is elegant in terms of modeling the acoustic and grapheme-to-phoneme relationship jointly even without using a seed lexicon. They are particularly suitable for recognition tasks for a specific domain, like name recognition. In the general ASR setting, however, it's potentially hard to train such a model robustly, because of the data-sparsity issue (usually we only have a few acoustic examples in training data for each word).

Generating alternative pronunciations from a phonetic decoder With acoustic examples of words which are interested in, we can generate pronunciation variants directly from the phonetic transcription obtained with a phonetic decoder composed of an acoustic model and a phonemic language model. These pronunciation variants could either be added into G2P training examples [15] [18] [19], or be combined them with the G2P generated pronunciations [20]. After collecting pronunciation candidates from both G2P and phonetic decoding, we could collect acoustic evidence for all pronunciation candidates, based on which we then prune them based on some criterion like pron-prob (pronunciation probabilities) [18] [14] [16] [15] [19], or, as proposed in [20], it's possible to decode acoustic training data iteratively to filter pronunciation candidates.

Our method using candidates from both G2P and phonetic decoding, falls into the second type of approaches. The aspect of the problem that we focus on is candidate pruning (selection). That is, given a set of pronunciation candidates from G2P and phonetic decoding (and maybe some from a manually created lexicon), which subset should we keep? Keeping all the pronunciations is impractical because it would make decoding slow, and also because too many pronunciations tend to hurt ASR performance, even when pronunciation probabilities are used [21].

Previous work on candidate pruning has relied on estimated pronunciation probabilities to determine which candidates should be cut [18, 14, 16, 15, 19].

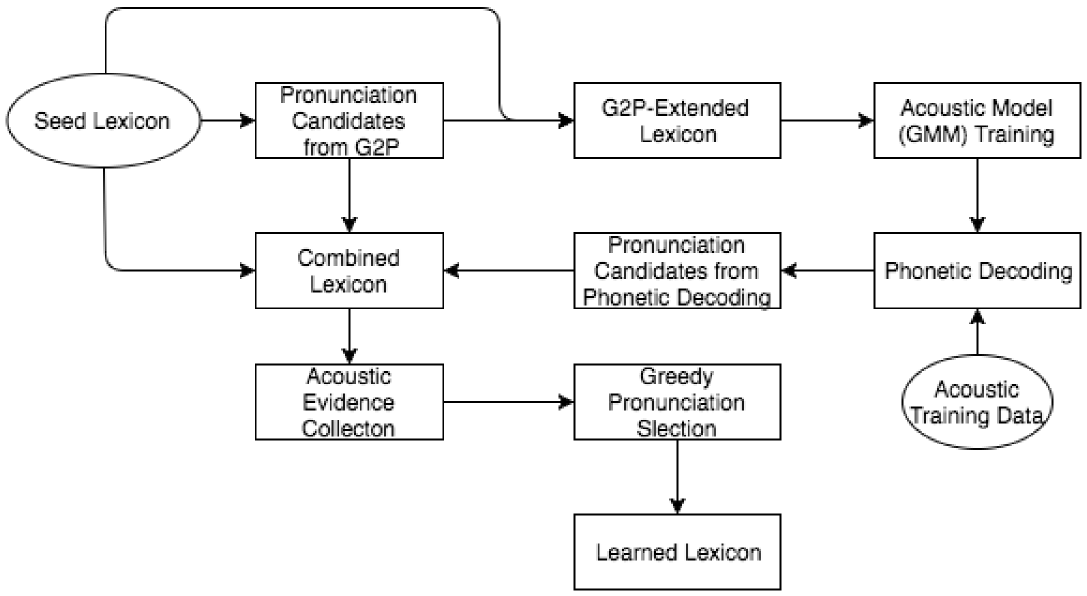

Seed Lexicon
Pronunciation Candidates from G2P
G2P-Extended Lexicon
Acoustic Model (GMM) Training
Combined Lexicon
Pronunciation Candidates from Phonetic Decoding
Phonetic Decoding
Acoustic Evidence Collecton
Greedy Pronunciation Slection
Acoustic Training Data
Learned Lexicon

Figure 2.1: The proposed framework of acoustic-data driven lexicon learning.

The main defect with this is that for words with multiple pronunciations, it tends to give us too many minor pronunciation variants (e.g. reflecting co-articulation effects), which is undesirable for ASR. If we rely on pronunciation probabilities alone it is hard to discard those types of variants while keeping variants that come from different meanings of the word.

The core contribution of our work is a likelihood-based criterion applied in a greedy framework for pronunciation candidates pruning, that naturally keeps only pronunciaiton candidates that are "far apart". Figure 2.1 is an overview of the proposed framework, which will be explained in detail in the following sections.

2.2 Collecting pronunciation candidates from multiple sources

There are many ways to generate pronunciation variants using acoustic examples. The first option is to simply use a phonetic decoder to generate pronunciation variants for words with acoustic examples, optionally combine them with the seed lexicon and then train a G2P model with the combined lexicon, and re-generate pronunciation candidates for all OOV words [19]. This eliminates the necessity of a seed lexicon and could help reduce the noise effect at the boundary of phonetic-decoding generated pronunciations, but may not produce appropriate pronunciations for foreign words whose pronunciation rule is relatively rare, and the G2P model performance might be sensitive to the quality of acoustic examples. A second option is to solely rely on phonetic-decoding generated pronunciations, and only use G2P generated pronunciations to produce alignment information (word boundaries) for generating pronunciations from phonetic-decoding. This ensures that the initial pronunciation candidates agree highly with the acoustics, but ignores the fact that phonetic-decoding generated pronunciations are much more noisy than G2P generated ones. To balance the two options, we choose to combine pronunciations from both G2P and phonetic-decoding (and the seed lexicon for IV words), into a large candidate pool, on top of which we will collect acoustic evidence for further selection.

In our framework, like [20], we first extend the seed lexicon to include OOV words in the training data, using a G2P model trained on the seed lexicon, and then train an acoustic model (AM) using the G2P-extended lexicon. Then we generate alignments for all training data, based on which we then train a bi-gram phonemic language model (PLM). Using this PLM and the AM, we construct a phonetic decoder and use it to generate phonetic transcription of training data. For each individual word token in the transcript, we can align it with a phone sequence using timing information from the alignments and phonetic transcriptions. Then for each specific word w, we can compute the relative frequency of each phone sequence being aligned to it, by normalizing each phone sequence's count by the most frequent phone sequence's count. Then we filter out those phone sequences whose relative frequency is too low (e.g. smaller than 0.1) and keep the remaining ones as the alternative pronunciations generated from phonetic decoding.

Then we combine these alternative pronunciation candidates with the G2P-extended lexicon into a large lexicon (called combined lexicon). For each word w from the combined lexicon, let $\mathcal{B}$ denote the set of pronunciation candidates collected from multiple sources, and b denote one pronunciation (baseform) candidate. The source of b (denoted as $s(b)$) could be one of the three: G2P/phonetic-decoding/reference. In the next section we will specify how we collect acoustic evidence for all pronunciation candidates in $\mathcal{B}$.

2.3 Acoustic evidence collection

"Acoustic evidence collection" itself is a vague concept. In the context of lexicon learning, it means collecting some statistics using acoustic data and the combined lexicon we have, which could be used to evaluate the relative "correctness" of all the pronunciation candidates for each word. Existing methods either collect acoustic evidence by counting from decoding results of the acoustic training data [20], or counting from alignments [14, 22], or from ASR lattices/N-best lists [16, 14]. We choose to collect acoustic evidence from lattices, cause the statistics are represented in soft counts, which provide richer information for pronunciation candidate selection.

First we introduce some notations. Let $\mathbf{O} = \{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_M\}$ denote acoustic sequences; M_w denote the number of utterances in $\mathbf{O}$ which contain the word w [3]; Then we further define θ_{wb} as the pronunciation probability of a pronunciation b for a word w ($\sum_{b \in \mathcal{B}} \theta_{wb} = 1$), and $\boldsymbol{\theta}_w \triangleq \{\theta_{wb} : b \in \mathcal{B}\}$ as the pronunciation model for word w. We define $\tau_{uwb} \triangleq p(\mathcal{O}_u | w, b)$ as the data likelihood given the pronunciation of w being b, which is determined by the acoustic model[4]. This is the "acoustic evidence" we want to derive from lattice statistics, which is needed by our pronunciation selection algorithm.

With the combined lexicon and an existing AM (the one we used for phonetic decoding in the candidate collection phase), we generate lattices for each training utterance. This lattice generation treats distinct pronunciations of words as distinct symbols for the purposes of lattice determinization, unlike our standard procedure described in [23]. This is achieved by putting both phone symbols and word symbols as the input sequence on the FST prior to lattice determinization. From the lattices, we can obtain per-utterance lattice pronunciation-posterior statistics $\gamma_{uwb} \triangleq p(w, b | \mathcal{O}_u)$.

[3]we assume that each word appears in each utterance's transcript at most once. In practice, if a word appears multiple times in an utterance, we divide the utterance into sub-utterances where each one only contains one token of the word.

[4]To be more precise, $p(\mathcal{O}_u | w, b)$ actually stands for $p(\mathcal{O}_u | \boldsymbol{\theta}_w = b)$ since $\boldsymbol{\theta}_w$ is the only parameter here. And it's the same for the posterior we'll introduce later: $p(\boldsymbol{\theta}_w = b | \mathcal{O}_u)$ v.s. $p(w, b | \mathcal{O}_u)$.

When the lattices were generated, we assign uniform priors over all pronunciation candidates of each word in the combined lexicon. By Bayes' rule, we can directly use the posterior statistics γ_{uwb} as the likelihoods τ_{uwb} [5]. Because lattices are pruned, a posterior γ_{uwb} could be zero even if w actually appears in a utterance u. So we always floor γ_{uwb} to a small positive scalar δ (In practice it's set between 10^{-7} and 10^{-5}), so that we have $\tau_{uwb} \geq \delta, \forall u, w, b$.

Based on γ_{uwb}, we can obtain another useful statistic, the average pronunciation posterior $\gamma_{wb} \triangleq \frac{1}{M_w} \sum_u \gamma_{uwb}$, where the summation $\sum_u$ is only taken over those utterances where the word w actually appears. It provides more information than the pronunciation probabilities estimated from the alignments [18], since lattices represent many alternative pronunciations for each acoustic example.

After the lattices were dumped, for each word, we prune away its pronunciations whose average posterior γ_{wb} is too low (e.g. only keeping the top 10), construct a new combined lexicon, and then re-generate the lattices and re-collect acoustic evidence in the same way. We found this pruning is always helpful as it improves accuracy of posteriors.

2.4 Data-likelihood-reduction based greedy pronunciation selection

The reason why we're interested in pronunciation selection, i.e. why we don't just allow all the pronunciations into the lexicon and learn the pronunciation

[5] Strictly speaking, Bayes' rule only gives us $\tau_{uwb} \propto \gamma_{uwb}$, i.e. τ_{uwb} can only be treated as γ_{uwb} up to a constant, but the constant doesn't affect the objective (2.5) we want to optimize.

probabilities, is that, using a big un-pruned lexicon would allow the lexicon to model variations due to co-articulation, accent variations, reduction and so on. However, the accepted opinion in the ASR community is that this hurts performance, and that it is better to allow the acoustic model (AM) and the context-dependent phones to model these types of effects [21]. We will also experimentally verify this argument.

Given we have acoustic examples of the words in concern from the training transcripts, the most obvious approach to pronunciation selection, which is adopted by most existing approaches, would be to estimate pronunciation probabilities (pron-prob) from acoustic data by counting from alignments [22], running EM using lattice statistics [14], .etc, and select pronunciations based on these probabilities. The main defect of using pronunciation probabilities is that for words with multiple pronunciations, it will tend to give us too many minor pronunciation variants (e.g. reflecting co-articulation effects), which can easily be modeled by each other, and too few pronunciations for totally different variants of a word. Consider the word "us", where the original and most common pronunciation is "uh s", and there are two alternatives present: "ax s" (representing a reduction of the word), and "uw eh s", representing the acronym "U.S.". We don't want the pronunciation representing the reduction, but we do want the acronym version. But in practice, the reduction gets a higher pron-prob than the acronym.

To tackle this challenge, we develop a framework for pronunciation selection which incorporates the acoustic data likelihood estimated from lattices. Basically, we formulate the pronunciation selection process as a greedy model selection procedure, with data-likelihood-reduction as the selection criterion, taking advantage of side information such as the source of those pronunciations, e.g. to favor human/G2P-generated candidates over those that arise from phonetic decoding. In the following section, we'll first specify how to compute the optimal data likelihood given a set of pronunciation candidates using EM and propose a pronunciation selection criterion based on likelihood reduction, and then use an illustrative example to compare the proposed selection criterion against other criteria. At last we talk about some practical issues in our algorithm, and summarize the whole iterative framework of pronunciation selection. Note that even if we have an iterative framework for pronunciation selection, we only need 1 or 2 rounds of lattice generation where we collect the acoustic evidence, which is very efficient.

2.4.1 A pronunciation selection criterion based on per-utterance likelihood reduction

Given a set of pronunciation candidates for a specific word w, and the conditional likelihood τ_{uwb} (acoustic evidence) for each utterance $\mathcal{O}_u$, we want to maximize the total data likelihood over the pronunciation model $\boldsymbol{\theta}_w$ [6]:

$$\mathcal{L}(\boldsymbol{\theta}_w) = \sum_u \log \left(\sum_b \tau_{uwb} \theta_{wb} \right) \tag{2.1}$$

where the summation $\sum_u$ is only taken over utterances where the word w actually appears. Since maximizing this objective doesn't have a closed form

[6]When we optimize the pronunciation probabilities for a specific word, we consider the pronunciation probabilities for other words as fixed.

solution, like [16], we use EM which maximizes the following auxiliary function instead (n stands for the iteration index, $\lambda_{uwb}^{n} \triangleq p(w, b|\mathcal{O}_u, \boldsymbol{\theta_w}^n)$ is the pronunciation posterior computed at the nth iteration)

$$Q(\boldsymbol{\theta}_w^{n+1}, \boldsymbol{\theta}_w^{n}) = \sum_u \sum_b \lambda_{uwb}^{n} \log \theta_{wb}^{n+1} \tag{2.2}$$

Maximizing the above function with the constraint $\sum_b \theta_{wb}^{n+1} = 1$ gives the M-step:

$$\theta_{wb}^{n+1} \leftarrow \frac{\sum_u \lambda_{uwb}^{n}}{\sum_u \sum_b \lambda_{uwb}^{n}} \tag{2.3}$$

According to Bayes' rule, we compute the updated posteriors λ_{uwb}^{n+1} as the following:

$$\lambda_{uwb}^{n+1} \leftarrow \frac{\tau_{uwb}\theta_{wb}^{n+1}}{\sum_b \tau_{uwb}\theta_{wb}^{n+1}} \tag{2.4}$$

which is the E-step. By running (2.3) and (2.4) iteratively until convergence, we can find an optimal pronunciation model $\boldsymbol{\theta}_w^{*}$, and evaluate the optimal log-likelihood (2.5) $\mathcal{L}(\boldsymbol{\theta}_w^{*})$ (denoted as $\mathcal{L}^{*}$ for simplicity).

In order to evaluate the importance of a specific pronunciation, say, b, we remove b from the pronunciation candidate set $\mathcal{B}$, re-initialize the pronunciation model $\boldsymbol{\theta}_w^{'}$ on top of $\mathcal{B}\backslash b$, and run EM to optimize (2.5) with the model $\boldsymbol{\theta}_w^{'}$. Writing the likelihood at convergence after removing b as $\mathcal{L}_b^{*}$, we can compute the per-utterance likelihood reduction associated with the pronunciation b as:

$$\overline{\Delta\mathcal{L}}_b \triangleq \frac{\Delta\mathcal{L}_b}{M_w} = \frac{\mathcal{L}^{*} - \mathcal{L}_b^{*}}{M_w},$$

This metric reflects the contribution of each pronunciation to the total data likelihood. With this metric, we can iteratively remove least important pronunciations in a greedy fashion, which is efficient. The pruning process is terminated if the smallest $\overline{\Delta\mathcal{L}_b}$ among all pronunciation candidates is larger than a user-specified threshold T. In summary, we look for an "optimal" subset of pronunciations for a word w, in a greedy fashion, by removing the "least important" pronunciation at each iteration, with the hope that the resulting subset balances the number of pronunciation candidates and the total data likelihood. The complete iterative framework is given in Section 2.4.5.

2.4.2 An illustrative example

Here we show an example to illustrate the advantage of pronunciation selection based on the per-utterance log likelihood reduction $\overline{\Delta\mathcal{L}_b}$ over the learned pronunciation probabilities $\boldsymbol{\theta}_w^*$, in terms of dealing with confusability of pronunciation variants.

In Table 2.1, we listed the pronunciation candidates, average pronunciation posteriors, learned pronunciation probabilities, and the per-utterance log likelihood reduction of two English words 'machine' and 'us' taken from the TED-LIUM [1] training corpus. Note that the two pronunciations of 'machine' only differ in one vowel, while the two pronunciations of 'us' represent two distinct meanings.

We want a selection criterion under which it's possible to put a threshold to rule out the reduction 'M IH SH IY N' (generated from phonetic-decoding) in the 'machine' case, while keeping the acronym 'Y UW EH S' in the 'us' case. Looking at the average posteriors γ_{wb} and the learned pronunciation

w	’machine’	’us’
$\mathcal{B}$	[’M AH SH IY N’, ’M IH SH IY N’]	[’AH S’, ’Y UW EH S’]
γ_{wb}	[0.839, 0.161]	[0.990, 0.010]
$\boldsymbol{\theta}_w^*$	[0.987, 0.013]	[0.992, 0.008]
$\overline{\Delta\mathcal{L}_b}$	[3.575, 0.004]	[15.576, 0.034]

Table 2.1: The pronunciation candidate set $\mathcal{B}$, average pronunciation posteriors γ_{wb}, learned pronunciation probabilities $\boldsymbol{\theta}_w^*$, and the per-utterance log likelihood reduction$\overline{\Delta\mathcal{L}_b}$ for two English words ’machine’ and ’us’ from TED-LIUM [1].

probabilities $\boldsymbol{\theta}_w^*$, both of them give lower values for ’Y UW EH S’ than ’M IH SH IY N’, and thereby cannot serve as the criterion we need. However, the per-utterance log likelihood reduction $\overline{\Delta\mathcal{L}_b}$ of ‘AH S’ is much larger than ‘M IH SH IY N’ (0.034 v.s. 0.004). Thus it’s possible to set a proper threshold on $\overline{\Delta\mathcal{L}_b}$ to keep ‘AH S’ and remove ‘M IH SH IY N’.

The underlying reason is that the confusability between pronunciations is reflected in the sharpness of the per-utterance pronunciation posteriors γ_{uwb}. In the ‘us’ case, the two pronunciation variants cannot easily model each other, and therefore the posteriors are very sharp for most examples. Thereby removing the minor pronunciation ‘Y UW EH S’ would result in a greater reduction in the data likelihood. Thus, beyond reflecting the relative frequency, the proposed criterion $\overline{\Delta\mathcal{L}_b}$ is capable of modeling the confusability between pronunciation candidates, which is preferable from the Maximum Likelihood point of view and therefore could help us to select an informative set of pronunciations.

2.4.3 Refining the pronunciation selection criterion $\overline{\Delta\mathcal{L}_b}$

One difficulty of directly using $\overline{\Delta\mathcal{L}_b}$ in an iterative pronunciation selection framework is that, we need to develop an interpretable threshold T in order to decide when to stop removing pronunciations. However, we notice the upper bound of $\overline{\Delta\mathcal{L}_b}$ can be achieved in an extreme case, where we remove an absolutely dominating pronunciation p (meaning: the observed conditional likelihoods satisfy: $\tau_{uwp} = 1, \quad \tau_{uwb} = \delta, \quad \forall b \neq p$). Before removing p, it's obvious from (2.5) that the maximum $\mathcal{L}(\boldsymbol{\theta}_w^*) = 0$ can be reached with $\boldsymbol{\theta}_w^*$ being a one-hot vector s.t. $\theta_{wp} = 1$. After removing p, with the constraint $\sum_{b\in\mathcal{B}\setminus p} \theta'_{wb} = 1$, the log-likelihood is a constant: $\mathcal{L}(\boldsymbol{\theta}'_w) \equiv M_w \log\delta$. Then we have: $\overline{\Delta\mathcal{L}_p} = (0 - M_w \log\delta)/M_w = -\log\delta$. According to this, we scale this upper bound by a scalar α between $[0,1]$ to get an interpretable threshold: $T = -\alpha\log\delta$, where $\alpha = 1$ corresponds to the above extreme case, which means, for a pronunciation to be not removed, it would have to be present with probability 1 in 100% instances of the word, and $\alpha = 0$ means we will never remove any pronunciation candidates. In practice, it's set between 0.005 and 0.2. We also make α dependent on the source $s(b)$ of the pronunciation, which enables us to use a more conservative threshold for selecting pronunciations from a source where the candidates' quality is lower in general, like phonetic-decoding (pd), e.g. by setting $\alpha_{g2p} = 0.02, \alpha_{pd} = 0.01$. So, we define the "score" of a pronunciation candidate as "how far away" its $\overline{\Delta\mathcal{L}_b}$ is to the corresponding threshold, i.e.:

$$q_b \triangleq \overline{\Delta\mathcal{L}_b} - T_{s(b)} = \frac{\Delta\mathcal{L}_b}{M_w + \beta_{s(b)}} + \alpha_{s(b)}\log\delta$$

In our framework we iteratively prune the pronunciation with the lowest score and terminate pruning when all pronunciation have positive scores. Note that the count M_w is smoothed with a source-dependent scalar $\beta_{s(b)}$ (5-15 in practice). The purpose is to keep the score from being to high when M_w is small, so that in general we select fewer pronunciations if we only have a few acoustic examples of a word.

2.4.4 An alternative selection criterion: Bayesian information criterion (BIC)

In statistics, the Bayesian information criterion (BIC) is a popular criterion for model selection among a finite set of models, where the model with the lowest BIC score is preferred. In our case, the BIC of a pronunciation model $\boldsymbol{\theta}_w$ is

$$\text{BIC}(\boldsymbol{\theta}_w) = \log(M_w)|\mathcal{B}_w| - 2\mathcal{L}(\boldsymbol{\theta}_w) \tag{2.5}$$

If we want to use BIC to select one pronunciation candidate to remove from the set $\mathcal{B}_w$, one naturally would choose the one results in the largest BIC reduction. So we can re-define the "quality score" q_b as the BIC change caused by removing this pronunciation:

$$\begin{aligned} q_b^{\text{BIC}} &\triangleq (\log(M_w)(|\mathcal{B}_w| - 1) - 2\mathcal{L}_b^*) - (\log(M_w)|\mathcal{B}_w| - 2\mathcal{L}^*) &(2.6)\\ &= 2\Delta\mathcal{L}_b - \log(M_w) &(2.7)\end{aligned}$$

Note that it's similar as the previously proposed quality score (adjusted likelihood reduction) q_b, in a sense that they all have a positive term of the likelihood reduction $\Delta\mathcal{L}_b$, though the scale is different. Similarly as the proposed greedy framework where we in each iteration we remove the pronunciation associated with the lowest q_b, we can just replace q_b with q_b^{BIC}, and we can still use the same stopping condition: terminate pruning when all pronunciation have positive scores, i.e. we can never reach a lower BIC by further removing any pronunciation. Note that the greedy framework cannot guarantee an globally optimal subset. For efficiency, it always makes sense to adopt a greed framework in case the candidate set is large. The drawback is that with the BIC criterion it's not straightforward to meaningfully incorporate prior information of the pronunciation quality of a specific source (G2P or PD).

2.4.5 Summary: an iterative framework

The proposed pronunciation selection algorithm, which iteratively prunes pronunciations from the initial candidate set $\mathcal{B}$, is summarized as Algorithm 1 ($\mathcal{B}_t$ stands for the selected subset of pronunciation candidates at iteration t). As mentioned in the last section, if we replace q_b with q_b^{BIC}, we get a similar greedy pronunciation pruning framework, which seeks a subset of pronunciations on top of which observed acoustic data have the locally lowest BIC score. This framework serves as a reasonable baseline. In the experiments part, we will compare the proposed framework with this baseline.

2.5 Experiments

We design two tasks to evaluate the performance of the proposed lexicon learning framework respectively, on estimating pronunciations for OOV words (lexicon expansion) and correcting/adapting pronunciations for in-vocabulary

Algorithm 1 Greedy pronunciation selection

```
set $t = 0$, $\mathcal{B}_0 = \mathcal{B}$.
While true:
    Initialize $\boldsymbol{\theta}_w$ uniformly on $\mathcal{B}_t$.
    Run EM on $\mathcal{B}_t$ to get $\boldsymbol{\theta}_w^*$ and the optimal log-likelihood $\mathcal{L}^*$.
    For $b$ in $\mathcal{B}_t$:
        Initialize $\boldsymbol{\theta}_w^{'}$ on $\mathcal{B}_t \backslash b$ and run EM to get the optimal log-likelihood
$\mathcal{L}_b^*$.
        Compute $\Delta\mathcal{L}_b = \mathcal{L}^* - \mathcal{L}_b^*$
        Compute $q_b = \frac{\Delta\mathcal{L}_b}{M_w + \beta_{s(b)}} + \alpha_{s(b)} \log \delta$
    If $\min_{b \in \mathcal{B}_t} q_b \geq 0$:
        Output $\mathcal{B}_t$ as the optimal pronunciation subset.
        Break.
    Else:
        $\hat{b} = \text{argmin}_{b \in \mathcal{B}_t} q_b$:
    $\mathcal{B}_{t+1} = \mathcal{B}_t \backslash \hat{b}$
    $t = t + 1$
```

(IV) words (lexicon adaptation) given acoustic data.

2.5.1 Lexicon expansion

2.5.1.1 WER performance analysis

In order to evaluate the performance of the proposed lexicon learning framework, a small seed lexicon is built by randomly sampling a small portion (5%) of words from the vocabulary of the expert lexicon of each task. With the seed lexicon, we train a G2P model using Sequitur [13] and apply it to all OOV (w.r.t the seed lexicon) words in the vocabulary of the expert lexicon, to get the "G2P-extended" lexicons.[7] A baseline system called G2P-ext is built using

[7]In this chapter we focus on lexicon learning for alphabetic languages. Thereby a G2P model trained with a small seed lexicon is able to generate pronunciations for most words in the expert lexicon.

a G2P-extended lexicon with the optimal number of variants per-word tuned on dev data, and another baseline system called G2P-1best is built using a G2P-extended lexicon where we only take the top G2P pronunciation for each word. With this G2P model and acoustic training data for each task, we can build a learned lexicon using the proposed framework, and then train an ASR system called “Lex-learn". Besides, we have an ASR system trained using the full expert lexicon as the “Oracle" system. Note that the training recipes of three ASR systems (G2P-ext, G2P-1best, Oracle, and Lex-learn) for each task only differ in the lexicons (with the same vocabulary). All experiments were done with Kaldi [24].

Table 2.2: ASR Performance on Librispeech (WER on the test-clean set (tuned on WER of LF-MMI systems on dev-clean, without 4-gram LM rescoring) with different lexicon conditions (the average # pronunciations per word for in-vocab words from training transcripts, are shown in parentheses). The vocab of the full expert lexicon (a subset of CMUDict) has 200K words.

	WER			
	Oracle (1.08)	G2P-ext (5.05)	G2P-1best (1)	Lex-learn (1.42)
SAT	11.32 %	13.11 %	14.57 %	11.53 %
LF-MMI	6.44 %	6.76 %	7.15 %	6.64 %

We first conduct experiments on the Librispeech-460 task [25]. For each lexicon condition, we use the 460h training data subset to build speaker-adaptive trained GMM (SAT) models (the same AM training recipe as the "SAT 460" from [25]), on top of which we then train sub-sampled time-delay neural networks (TDNNs) [26] with the lattice-free MMI (LF-MMI) [27] criterion. The WERs are shown in Table 2.2. It can be seen that the learned lexicon performs better than G2P-extended lexicons, and is close to the oracle lexicon. And the LF-MMI systems are much more robust to the lexicon quality than SAT systems, i.e. the G2P-extended and learned lexicons perform closer to the expert lexicon. The learned lexicon closes 88% (SAT)/ 36%(LF-MMI) of the WER gap between the G2P-ext system and the oracle system. Also, looking at the average number of pronunciations per word, the learned lexicon (1.42) is much more compact than the G2P-extended lexicon (5.05), and is very close to the G2P-1best lexicon (1), though it performs much better than the G2P-1best lexicon by a large gap: 20.9% (SAT) / 7.1%(LF-MMI) relatively in WER.

Table 2.3: ASR Performance of lexicon learning for lexicon extension on Babel (WER of SAT systems on the Dev10h set).

Language (vocab. size)	WER		
	Oracle	G2P-ext	Lex-learn
Bengali (24.3K)	63.2%	64.4%	64.0%
Pashto (17.6K)	57.7%	59.6%	59.0%
Turkish (38.3K)	57.0%	57.9%	57.6%
Tagalog (21.1K)	55.4%	57.6%	57.0%

The proposed lexicon learning framework is also investigated in Babel ASR tasks, where we have much less acoustic training data available. We build SAT systems for Bengali, Pashto, Turkish and Tagalog, which are trained on an IARPA-provided full language pack (FullLP) containing 80 hours of transcribed speech for each language. The WERs were measured on IARPA-provided 10 hours of transcribed speech as the development set[8]. From the

[8] Bengali, `IARPA-babel103b-v0.4b`;

results shown in Table 2.3 we can see that, lexicon learning brings 0.3% − 0.6% absolute WER improvements over the G2P-extended lexicon. For languages like Turkish, where we have a relatively larger expert lexicon, the 5% seed lexicon contains enough words so that the G2P-ext system already performs close to the Oracle system, and thereby the absolute improvements brought by lexicon learning is relatively smaller.

2.5.1.2 Individual word recognition performance analysis

The performance comparison in WERs has given us an idea about the benefits of utilizing acoustic information in lexicon learning. To have a deeper understanding about the benefit, we will design experiments to answer the following questions:

- Does the performance gain come from better recognizing words whose pronunciations were touched (learned) by our method?
- For a specific word seen in training data, Does more acoustic examples in training data result in larger performance gain in test?

Our assumption is that on words seen in training data, the proposed algorithm would bring a larger performance improvement over a pure G2P-based lexicon expansion method, and the more frequent they occur in training data, the larger improvement we will get.

Pashto, `IARPA-babel104b-v0.4bY`;
Turkish, `IARPA-babel105b-v0.4`;
Tagalog, `IARPA-babel106b-v0.2g`.

2.5.1.2.1 Evaluation metric Here we propose a simple metric which measures the recognition performance on a specific set of tokens from test data, based on LVCSR results:

- Using timing information from the CTM from decoding and the CTM obtained with reference transcriptions, we can align the ASR hypothesis with the reference on token level.
- Then we compute the overlap between each reference token and each hypothesis token.
- For each reference token, we pick the hypothesis token which has the maximum overlap with it (could be none).
- Mark one reference token as a **"miss"**, if the max-overlapping hypothesis token is not the reference token.
- On a set of tokens, we define Token Miss Rate (TMR) as:

$$\text{Token Miss Rate} = \frac{\#\ \textit{Missed Tokens}}{\#\ \textit{Tokens}} \times 100\% \tag{2.8}$$

Then we apply the proposed metric to the best ASR outputs of the Oracle/G2P-nbest/Lex-learn systems shown before in the Librispeech-460 task (Table 2.2). The test-clean dataset contains 63.9K tokens in total, and 52K of them are seen at least once in training transcripts. So we measure the token miss rate of each system on set of tokens seen/unseen in training data, and also on the whole set.

We present both absolute and relative (to the Oracle system) token miss rates on test in Table. 2.4. It could be seen that, on the "Seen" partition, lexicon learning considerably lowers the token miss rate, reducing the relative degradation (compared with Oracle) from 15.9% to 3.7%, which is about 6 times larger than the overall improvement which is from 3.3% to 1.1%. This confirms our earlier assumption. The relative performance change on the "Unseen" partition is not statistically significant, because the absolute miss rates are already very high. This implies it's still hard to recognize words unseen in training transcripts.

We realize that, out of the 52 tokens in test data which are seen in training data also, 49.6K of them have their pronunciations touched by our method, and of course for all words unseen in training data have their pronunciation untouched by our method. To better support our assumption, we split the all tokens in test data by the "touched"/"untouched" (by our method) criterion, and we want to check whether most improvements really come from the "touched" words, proving the effectiveness of our lexicon learning method. It can be seen Table. 2.5 that, the overall relative improvement of lex-learn over G2P, which is 2.1, indeed comes from the improvement on words of the "touched" condition, which is 11.3. And the tendency is very similar as Table. 2.4, verifying that doing analysis on the "seen/unseen" partitions, which is generally easier to conduct, is informative enough.

Similar results on dev data (seen/unseen partitions) are presented in Table 2.6. The relative gains are more significant, since the lexicon parameters were tuned on dev data.

Next, for a specific token, we want to investigate the relationship between

Table 2.4: Individual word recognition performance on Librispeech based on best Nnet ASR outputs on test-clean, for words seen/unseen in training transcripts.

Token set	Token Miss Rate (%)		
(size)	Oracle	G2P-nbest	Lex-learn
Seen (52K)	6.2	7.2	6.4
Unseen (11.9K)	98.2	98.1	98.6
Total (63.9K)	23.4	24.1	23.6

Token set	Token Miss Rate (relative % to Oracle)		
(size)	G2P-nbest	Lex-learn	Improvement
Seen (53.8K)	+15.9	+3.7	12.2
Unseen (11.6K)	-0.12	+0.38	-0.4
Total (65.5K)	+3.3	+1.1	2.2

Table 2.5: Individual word recognition performance on Librispeech based on best Nnet ASR outputs on dev-clean, for words whose pronunciations were touched/untouched by our method.

Condition	#. words by type	#. words by token	Token Miss Rate (%)			
			Oracle	G2P-nbest	Lex-learn	Impr.
Touched	7.4K	49.6K	6.1	7.1	6.3	11.3
Untouched[9]	0.6K	14.2K	83.3	83.2	83.7	-0.6
Overall	8.0K	63.9K	23.4	24.1	23.6	2.1

the recognition performance in test data and the amount of its acoustic examples in training data, answering the second question:"For a specific word seen in training data, Does more acoustic examples in training data result in larger performance gain in test?" We focus on all the 52K tokens seen in training data, break down this token set into several bins according to their word frequency in training transcripts, and show the relative performance degradation of the baseline and our method, and compute the performance improvement. It can be seen from Table. 2.7 that there is a positive correlation between the frequency and the performance improvement, i.e. the more acoustic example we have in training data, the larger relative performance gain we can get, in

Table 2.6: Individual word recognition performance on Librispeech based on best Nnet ASR outputs on dev-clean, for words seen/unseen in training transcripts.

Token set (size)	Token Miss Rate (%)		
	Oracle	G2P-nbest	Lex-learn
Seen (53.8K)	5.9	6.9	6.0
Unseen (11.6K)	98.3	98.1	98.5
Total (65.5K)	22.3	23.1	22.5

Token set (size)	Token Miss Rate (relative % to Oracle)		
	G2P-nbest	Lex-learn	Improvement
Seen (52K)	+18.2	+2.7	15.5
Unseen (11.9K)	-0.2	+0.3	-0.5
Total (63.9K)	+3.8	+0.8	3.0

terms of recognizing this word in test data.

Table 2.7: Individual word recognition performance on Librispeech based on best Nnet ASR outputs, for words with different frequencies in training transcripts

Frequency in (training data)	#. words by type	#. words by token	Token Miss Rate (%)			
			Oracle	G2P-nbest	Lex-learn	Impr.
$[0,5)$	0.5K	0.6K	26.1	28.4	27.9	1.8
$[5,20)$	1.2K	1.4K	14.0	15.1	14.1	6.5
$[20,100)$	2.7K	4.2K	7.9	8.8	8.2	6.8
$[100,\infty)$	3.0K	43.4K	5.4	6.4	5.6	12.7

2.5.1.3 Investigating different pronunciation generation and selection approaches

In this part we plan to investigate how pronunciations derived from acoustics and acoustic-evidence based pronunciation selection methods, including our proposed method, affect the lexicon learning performance. In Table. 2.8, we compare the proposed framework with various baseline lexicon expansion approaches, on the Librispeech-460 task (WER of SAT systems).

2.5.1.3.1 Evaluation metrics There are two evaluation metrics. The first one is WER (4th column) on test data, which reflects the overall quality of the expanded lexicon. The second one is the average number of pronunciations per word in the expanded lexicon (3rd column), which reflects the lexicon size. A smaller lexicon size means we'll have a smaller decoding graph in test time, which implies faster decoding.

2.5.1.3.2 Evaluation conditions and baseline methods In terms of seed lexicon, in order to make the performance gap between different systems more noticeable, we take a smaller seed lexicon containing only $1\%(2K)$ randomly sampled words from the same expert lexicon as before.

In terms of pronunciation candidate pool (1st column), we compare G2P over G2P+PD (which means we put candidates from phonetic decoding (PD) together with G2P generated candidates before candidate pruning).

In terms of candidate pruning criterion (2nd column), "G2P-score", as described before, is a baseline built with a G2P-extended lexicon, taking n-best candidates ranked by the G2P-score (with n tuned on dev data). "pp" means pronunciation-probability-based pruning on G2P candidates". Basically we first align acoustic training data with a large G2P-extended lexicon containing all G2P generated candidates (up to 10 candidates per word, and we can optionally add phonetic-decoding (PD) generated pronunciation candidates), and then use max-normalized pronunciation probabilities [18] to prune those candidates for each OOV word, with a tuned threshold (0.4). "BIC" means we use the same pronunciation candidate pool as pp, but using the Bayesian information criterion (BIC), as described before, as the pruning (model selection) method. The last system "likelihood-reduction" is the proposed framework (i.e. the "Lex-learn" systems listed before). For fair comparison, under different lexicon conditions, the acoustic models were re-trained on top of the same acoustic model (the one used in the G2P-score based pruning baseline system). Results are presented in Table. 2.8.

2.5.1.3.3 Observation First, it can be seen that even if we only use G2P based pronunciation candidates, acoustics-based pronunciation pruning could still help a lot in WER (from 13.72% to 13.06%) and also in reducing the relative lexicon size (from 6.57 to 3.77), compared with a G2P score based selection criterion with a tuned threshold. This confirms the finding from earlier papers like [18] [14]. Second, in terms of candidate generation, it can be seen that adding PD candidates to the candidate pool is crucial to the lexicon quality (0.82% WER gain), which implies the importance of using acoustic information in the candidate generation stage. Furthermore, in terms of candidate pruning, comparing with the pronunciation probability (pp)-based pruning criterion, the proposed pronunciation selection method solely brings 0.18% WER gain (from 12.24% to 12.06%) and lowers the number of pronunciations per word from 5.43 to 1.59, meaning it results in a smaller lexicon. Also, notice that the proposed method is also better than pronunciation-pruning with BIC, in terms of both lexicon size (average number of pronunciations per word) and WER performance. This confirms that the proposed pronunciation selection framework enables us to achieve better ASR performance with a much more compact lexicon compared with other acoustic data-driven pronunciation selection methods (pp/BIC-based).

Table 2.8: ASR performance (WER of SAT systems on the test-clean set, without LM rescoring) comparison on Librispeech, with different lexicon expansion approaches.

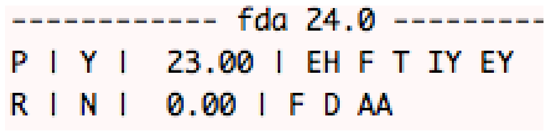
Lexicon condition			WER
Pronunciation candidates	Pruning criterion	#pronunciations per word (avg.)	
G2P	G2P-score	6.57	13.72 %
G2P	pp	3.77	13.06 %
G2P+PD	pp	5.43	12.24 %
G2P+PD	BIC	4.2	12.46 %
G2P+PD	likelihood-reduction	1.59	12.06 %

2.5.2 Experiments: lexicon adaptation

We have just shown results on learning pronunciations for OOV words, i.e. lexicon expansion. Here we want to investigate how the proposed method works for learning pronunciations of IV words, introducing a new task – lexicon adaptation. Basically, we aim at automatically fixing pronunciations of *IV words* from an expert (reference) lexicon, which are **wrong** or significantly **mismatch** with acoustic data, e.g. **FDA** : '`f d a`' from the `Cantab` lexicon (Tedlium).

Here is the procedures of applying the proposed lexicon learning framework for lexicon adapation:

- Generate G2P+PD candidates for words from the reference lexicon, and put them together with reference candidates into a "combined" lexicon.



```
------------ fda 24.0 ---------
P | Y |  23.00 | EH F T IY EY
R | N |  0.00 | F D AA
```

Figure 2.2: One example entry in the lexicon-edit file which presents the lexicon adaptation results.

- Apply our pronunciation pruning framework, with hyper-parameters set to reflect our prior preference to reference candidates (in most cases we want to prune G2P/PD candidates more aggressively than reference candidates, since most reference candidates are correct).
- Present the results in an human-editable "lexicon-edit" file (Figure 2.2).
- Optionally, we can apply inspection/refinements by linguists to lexicon-edit file[10], then it can be applied to the reference lexicon to produce the adapted lexicon.

2.5.2.1 Experimental setup

We test the lexicon adaptation recipe on Tedlium[1] (English TED Talk) corpus. It contains 118 hours of training data and 4 hours of development and test data. The reference lexicon contains 152K words, where 57% words come from CMUDict, and the left come from Festival Speech Synthesis System[11], meaning those pronunciations are not human-annotated and are therefore error-prone.

[10]The size of the edit file is well controlled: e.g. If we are confident about most reference pronunciations, by setting hyper-parameters properly, most words' learned pronunciation are just the original reference candidates, and these words won't appear appear in the edit-file at all. So, e.g. we only propose pronunciation changes to several thousands words for a linguist to review.

[11]http://www.cstr.ed.ac.uk/projects/festival/

We first train a baseline AM (TDNN-LSTM[28]) using the reference lexicon. Then we generate the automatically adapted lexicon using the proposed framework (without human refinements). The edit-file contains 5.6K words, 4.7K of them have their reference pronunciations given by CMUDict, meaning our method suggests changes to lots of pronunciations given by the CMUDict based on acoustic evidence.

2.5.2.2 Evaluation conditions

We basically have two goals to achieve with lexicon adaptation:

- Improve recognition performance on individual words directly, just by re-decoding with the adapted lexicon.
- Improve general AM performance by re-training AM with adapted lexicon.

In order to investigate how lexicon adaptation helps to the above goals, we design the following evaluation conditions:

- ***baseline***: Decode with the **reference lexicon** & **baseline AM**.
- ***re-decode***: Re-decode with **adapted lexicon** & **baseline AM**.
- ***re-train***: Re-train AM with **adapted lexicon** and then decode.

Also, we evaluate ASR performance on both word level (TMR, which is Token Miss Rate. See Eq. (2.8)) and utterance level (WER), in order to understand the effect of lexicon adaptation in depth.

2.5.2.3 Evaluation results — word level

To understand where does the performance improvement come from, we split the Tedlium dev and test data into two subsets: Out of the 45K tokens in $dev + test$ test in total, 3.7K (1.1K by type) have their pronunciation adapted, and we call this subset of T_a. The left 41.3K tokens (4.2K by type, 99% seen in training) constitute T_b. We then compute TMR on T_a and T_b separately.

It can be seen that even if we test on the merged $dev + test$ set, the size of T_a is still small (3.7K). Thanks to the newer version of TED-LIUM data (version-3 [29]), we can now test our method on a much larger dataset. Basically, we take all version-3 training data not present in version-2 training data (which was used to train the acoustic models) to make a new test set called *test.large*. This subset contains 162 hours of data, which is much larger than the 4 hours dev+test data we have. The evaluation scheme is the same as before: Out of the 1536K tokens in the *test.large* set in total, 126K have their pronunciation adapted. Again we call it T_a, and the left 1410K as T_b, and compute TMR on T_a and T_b separately.

Condition	Token Miss Rate (%)	
	T_a	T_b
baseline	7.99	9.74
re-decode	**7.67**	9.78
re-train	7.61	**9.45**

Table 2.9: Token Miss Rate on all tokens from Tedlium test+dev sets, whose pronunciations are adapted (T_a) during training by our method or not (T_b).

From Table. 2.9 we can see the results. On T_a where we adapted pronunciations, 4% relative TMR improvement is achieved simply by re-decoding

Condition	Token Miss Rate (%)	
	T_a	T_b
baseline	9.45	32.3
re-decode	**9.19**	32.3
re-train	8.98	32.3

Table 2.10: Token Miss Rate on all tokens from Tedlium *test.large* set, whose pronunciations are adapted (T_a) during training by our method or not (T_b).

with the adapted lexicon, proving adapting pronunciations directly helps recognize individual words. Re-training gives similar results. The trend in Table. 2.10 is similar, with re-decoding giving around 3% improvement over baseline. But re-training gives around 0.2% absolute improvement further, implying re-training the AM with the adapted lexicon helps with overall AM performance.

On T_b where words don't have their pronunciations adapted, re-decoding doesn't give improvements as expected, but re-training gives some improvements in Table. 2.9, though in Table. 2.10 it's not giving improvement.

Here are some example tokens which were missed in the baseline, but correctly recognized in the "re-train" condition:

Token	Pronunciation	
	reference	learned
schizophrenic	`SH IH Z AH F R EH N IH K`	`S K IH T S AH F R EH N IH K`
harry	`HH EH R IY`	`HH AE R IY`
ipcc	`IH P K`	`AY P IY S IY S IY`

Table 2.11: Example tokens missed in the baseline, but correctly recognized in the "re-train" condition

In Table. 2.11, we can clearly see why lexicon adaptation helps on recognizing particular words. We intended to select examples reflecting three different scenarios. It can be seen that, lexicon adaptation helps correct the wrong syllable "SH IH Z" in the first example. The second examples reflects adaptation of word `harry` from British accent to US accent, and the third example shows it corrected the pronunciation of an acronym "IPCC".

2.5.2.4 Evaluation results — utterance level

Similarly as word level evaluation, we separate test utterances into two subsets according to whether the utterance contains words whose pronunciations are adapted or not (from T_a). Here are the statistics:

For the *dev* + *test* set, out of the 1662 utterance in total, 1333 of them contain at least one token in T_a. We call this subset U_a[12]. And the left 329 utterances constitute subset U_b. And we compute WER on U_a and U_b separately in Table. 2.12.

Likewise, for *test.large* set: out of the 97.7K utterance in *test.large* in total, 65.6K of them contain at least one token in T_a. We call this susbet U_a[13]. And the left 32.1K utterances constitute subset U_b. We also compute WER on U_a and U_b separately, in Table. 2.13.

Condition	WER (%)	
	U_a	U_b
baseline	8.86	11.4
re-decode	8.85	11.4
re-train	**8.72**	**10.7**

Table 2.12: WERs on all tokens from Tedlium *dev* + *test* set, on two subsets U_a (containing at least one word from T_a) and U_b (otherwise).

[12]On average, each utterance in U_a contain 2.8 tokens from T_a
[13]On average, each utterance in U_a contain 1.9 tokens from T_a

Condition	WER (%)	
	U_a	U_b
baseline	15.1	16.4
re-decode	15.1	16.5
re-train	**14.9**	**16.2**

Table 2.13: WERs on all tokens from Tedlium *test.large* set, on two subsets U_a (containing at least one word from T_a) and U_b (otherwise).

From Table. 2.12 and Table. 2.13, we can see that, as the average num. of tokens with changed pronunciations (2.8 in $dev + test$ and 1.9 in *test.large*) is small, we're not able to see improvement on the overall WER be re-decoding on U_a. But we indeed get improvements on both U_a and U_b by re-training the AM on both $dev + test$ and *test.large*, which confirms the adapting the pronunciations to acoustic evidence helps with overall AM performance (ranging from 1% to 6%). Note that in terms of statistical significance of WERs, results on *test.large* is much more believable, where we have 0.2 absolute WER improvements on both U_a and U_b consistently over $dev + test$ and *test.large*.

2.6 Summary

In this chapter, we described an acoustic-data driven lexicon learning framework using a likelihood-based criterion for selecting pronunciation candidates from multiple sources, i.e. G2P and phonetic decoding. With the proposed criterion, the pronunciation candidates are pruned iteratively in a greedy way, based on the acoustic data likelihood reduction caused by removing each candidate. This approach enables us to construct a compact yet informative lexicon for both generating pronunciations for OOVs (lexicon expansion) or improving/adapting pronunciations for IVs (lexicon adaptation). Experiments on various ASR lexicon expansion tasks show that, with the proposed framework, starting with a small expert lexicon (containing $0.88K$ to$10K$ words), we are able to learn a lexicon for OOV words which performs closer to a full expert lexicon in terms of WER performance on test data, than lexicons built using G2P alone, with a pruning criterion based on pronunciation probabilities, or BIC. Individual word recognition experiments have confirmed the WER improvements indeed come from improved recognition performance of words whose pronunciations are learned by the proposed method. Also we have shown there's a positive correlation between word the number of acoustic examples and performance of lexicon learning. Experiments on lexicon adaptation task shows that for adapting the pronunciations of IV words in a reference lexicon to acoustic evidence helps improve both recognition performance on individual words just by re-decoding with the adapted lexicon, and overall AM performance by re-training AM with the adapted lexicon.

Chapter 3

Handling OOVs in Test Time

In the previous chapter, we focused on using acoustic data to improve estimation of OOV words' pronunciations in AM training data. This helps us better utilize the AM training data at hand, and thereby improving ASR performance during test time. In this chapter, we'll focus on dealing with OOVs in ASR test condition, i.e. how to detect and recover spellings of OOV words in test data, given we have neither a word's spelling nor its pronunciation in the lexicon.

The vocabulary of human speech of a particular language is intrinsically infinite. As a language evolves, there are always new words occurring. Therefore it's not possible to cover all words in a language by a closed vocabulary, and in closed-vocabulary ASR, OOV words in test utterances can't be recognized. And the recognizer will output an acoustically similar in-vocabulary (IV) word as the substitution. It mostly won't help with human readability, since there's no guarantee of acoustic/semantic similarity between the substitution and the original OOV word. Also, the occurrence of OOVs affects recognition of their surrounding IV words. This is one of the biggest challenges in ASR and related applications like spoken term detection/keyword search. In many cases, the OOV issue is of particular interest, when OOVs are proper nouns like human/place names, which could be important keywords given a certain context. Different from the keyword search task, in ASR, we don't have OOVs' spelling known in advance, making the problem more challenging. So, research into OOV detection and recovery for ASR has been of particular interest in the ASR community, which is the main focus of this chapter.

Besides, OOV retrieval/keyword search is another important related task. It's generally less challenging than OOV recovery, in a sense that we know the spelling (and even acoustic examples) of the OOV we want to retrieve from test speech in advance, and we need to retrieve the location of such instances from speech. In contrast, in OOV recovery we don't know either the pronunciation or spelling of the OOV in test speech, while we want to recover it's spelling as accurate as possible. So it's fair to say progress in OOV recovery research, which is a harder task, could benefit OOV retrieval as well.

The goal here is to build a OOV detection & recovery pipeline that could be easily integrated to a standard WFST decoding and RNNLM re-scoring framework, with both OOV recovery and overall decoding performance improved by enabling OOV LM probability estimation, efficient 2nd pass decoding, and open-vocabulary word RNNLM re-scoring. Also we aim to minimize additional computational overhead, making it a practical framework for word-based open-vocabulary ASR decoding.

3.1 Motivation and existing methods

3.1.1 OOV modeling and detection

In terms of OOV detection, there are generally two families of approaches being well investigated. The first family models OOVs implicitly with sub-word units.

3.1.1.1 Modeling OOV implicitly

There are two types of approaches within this family:

- Detects OOVs by finding inconsistency between sub-word (e.g. phone) and word recognition results [30, 31, 32]. This type of approach is very intuitive. The rationale is that when there's an OOV, the word recognizer will have to "guess" and approximate the OOV with an IV word, so that phone posterior distribution is relatively flat (because of confusion about predicted phones), and disagrees with the phone posterior distribution from the phone recognizer at the same region. While at region of IVs, the phone posterior distribution is relatively sharp, implying certainty about predicted phones, and it agrees with the phone posterior distribution from the phone recognizer. The disadvantage of this approach is that two separate passes of decoding are always needed.

- Build a hybrid language model combining word and sub-word (e.g. word pieces/fragments) units [33, 34, 35, 36, 37, 2, 38, 39, 40]. For example, one can train an LM on texts where low frequency words are replaced by word pieces/phones [38, 2], thereby combining both word and sub-word level LM information into a single hybrid LM. Then during decoding, at the OOV regions, word pieces will have higher posteriors in the confusion network/lattices.

There's also some work combining the above two types of approaches, e.g. [41]. For most methods in this family, a classifier is built to detect OOVs. One can either design rich features extracted from lattices/confusion networks, e.g. fragment posterior, word entropy [36, 37, 2], or directly feed into "raw" features like phone posteriors into a neural network [32]. The disadvantage of this family of approach is that, labeled data is usually required for building the OOV classifier, and also implicit OOV modeling makes it not straightforward to conduct downstream processing like OOV recovery.

3.1.1.2 Modeling OOV explicitly

This family of approaches was first proposed in [42], while success on a limited domain (weather information) was reported. Then more recently, [43, 44] proved this approach works well for OOV detection in the LVCSR setting. Basically, the goal is to build a filler model/generic word model, with a special structure, to "absorb" OOVs during decoding. The generic word model is usually a sub-word language model, e.g. a bi-gram phonemic language model [42, 43], or graphone (grapheme-phoneme pair) language model (i.e. a G2P model) [45], built to model the OOV pronunciations. It's combined with the pronunciation lexicon, as a hybrid lexical model (HLM), and the generic word model is tagged by the OOV symbol <unk>. Correspondingly, the language model is also open vocabulary, in a sense that the OOV symbol <unk> occurs as a separate unigram.

A variant of this approach is to embed the sub-word language model into the word language model rather than the lexicon, resulting in a hybrid language model [46, 43]. The advantage is that we can use modeling units larger than phone, e.g. phone multi-grams (variable length sequences of phones) in the sub-word language model. In this case, extra entries mapping the sub-word units to phones need to be added in the lexicon. So it's still a hybrid lexical model, modeling the pronunciation of both words and sub-words. Besides, rather than incorporating the sub-word LM into word LM in a hierarchical manner, another option is to built the mixed sub-word + word LM in a flat manner [47, 48]. And this approach can be further combined with the hybrid lexical modeling approach (to be more specific, a graphone LM) to produce a hierarchical [49] language model. Compared with training a sub-word LM from in-vocabulary words (the HLM approach), the data preparation on training LMs on hybrid text data and dealing with word boundaries are more complicated. However this idea is very similar to word-piece (e.g. byte-pair encoding (BPE)) based modeling approaches [50], which are becoming popular recently for both MT and ASR, mostly in an end-to-end setting.

With the hybrid lexical model and open-vocab/hybrid language model, the decoding graph will be a hybrid decoding graph capable of producing both IV words and OOV (<unk>) tokens with their phonetic transcriptions (pronunciation) retrieved from the generic word model. During decoding, the recognizer is free to go through either in-vocabulary words or the generic word model, whichever maximizes the total likelihood of a full lattice path.

Where and how likely <unk> tokens occur in lattices, is decided by the sub-word language model, the unigram probability of <unk> in the word language model, and acoustic observations. For simplicity we will name this family of approaches as the HLM approach.

There are several advantages of modeling OOVs explicitly. First, for OOV detection, a separate recognizer / decoding pass is no longer needed. Neither will we need to build a classifier to detect OOVs. Therefore it could better fit into the standard decoding framework ignorant of the OOV issue. In terms of OOV recovery, as we model OOVs explicitly as a separate lexical unit (the <unk> token), OOV candidates will occur in lattices/confusion networks as separate tokens, with their pronunciations produced by the generic word model, aligned with the word boundary. This makes it straightforward to recover the spelling of OOVs, which is important to down-streaming applications like OOV recovery/retrieval. So in our work, we will follow this idea (with different implementation details, which we will specify). However, to our best knowledge, there has been no serious research done with comparison between this HLM-based OOV detection approach, and the "implicitly modeling OOV" approach, which will be one of the contributions in this dissertation.

3.1.2 OOV recovery

In terms of OOV recovery, as we can find the pronunciation of predicted <unk> tokens from lattices/one-best hypothesis, the most straightforward method for recovering OOVs is to apply a P2G model on the phone sequences to obtain the spelling of OOV candidates, which has been proved to be successful in OOV recovery in an LVCSR setting [46, 44]. Therefore we adopted this approach in our work.

As there are many challenges in recovering OOVs, e.g. phone sequences aligned to <unk> are noisy so that multiple tokens of the same OOV word are recognized as different phone sequences, especially the word boundaries are hard to be aligned accurately,.etc.

Therefore, focuses along two directions have been proposed to improve this situation:

3.1.2.1 Clustering OOV candidates

Along one direction, various similarity scoring and clustering techniques have been proposed. It's shown in [39, 48, 51] that clustering frequent OOVs with linguistic/acoustic features and context information of OOV candidates from the one-best hypothesis helps with OOV recovery performance, and also other tasks like Query-by-Example (QbE). More recently, [44] proposed retrieving and then clustering OOV instances as phone lattices from word lattices, by pair-wise FST composition. This type of approaches is supposed to be able to deal with slightly different acoustic instances of the same OOV more robustly, though computationally expensive, because of the nature of FST operations. Also, this direction assumes that there are lots of recurring OOVs, so it's always applicable and worthwhile to cluster the detected OOV instances. However in our work we focus on infrequent/rare OOVs, that's why we won't pursue this direction. In our understanding, if some OOVs are very frequent, it'll be worthwhile to get human efforts to identify such case and add the word in concern into lexicon.

3.1.2.2 Estimating OOV LM probabilities

Along the other direction, there're attempts to improve OOV recognition performance by extending LM during test time to better estimate LM probabilities of OOVs. [52, 53, 54] investigated estimation of OOV LM probabilities where OOVs are the predicted words, or prediction of IV words' probabilities with OOVs in the history, in both n-gram LM and RNNLM. The methodology is, given the ground-truth spelling of the OOVs, they could estimate the LM probabilities of OOVs by looking at the occurrence statistics from extra corpus containing these OOVs, or use similarity-based approaches[1] to estimate them from probabilities of IV words. These methods tried to incorporate capability of dealing with OOVs into the LM in test time, without having to change the architecture or re-train the LMs. However all these efforts assume we already have access to some extra corpus containing these OOVs, meaning we know the ground-truth spelling of them. In such cases, a better baseline should be LM adaptation-based approaches like [55], which is not investigated in those works. LM adaptation-based approaches should be investigated in such cases, since it's found in [53] that using extra corpus to learn the LM behavior of OOVs is more effective than using the behavior of similar IVs. Though in our research, in most places we assume no access to extra corpus containing these

[1]They have shown that using similar (e.g. closest in the word embedding space) IV words' LM probabilities to estimate OOV LM probabilities gives no WER improvement than treating all OOVs equiprobable. The reason is probably that, there's no guarantee that an OOV, especially for rare name entities, could always find a reliable set of IV neighbors. Therefore it's hard to guarantee estimating LM probabilities based on the IV neighbors is robust.

OOVs, we overall follow this direction. The reasons is that, not like clustering OOV candidates, estimating LM probabilities for recovered OOV candidates is a must. Also this is a relatively new problem, especially for dealing with OOVs in word-level RNNLMs. Note that recently [56] proposed a method for end-to-end ASR decoding with RNNLMs at both the character and word level[2]. Similar as the hierarchy n-gram LM approach, during decoding, hypotheses are first scored with the character RNNLM until hitting a word boundary, then a word RNNLM will be used for re-scoring IV words, while the character RNNLM provide scores for OOV words. Our work adopts the similar idea of estimating OOV LM probabilities by a phone-level/character-level LM. While instead of combining two RNNLMs at decoding time, which is computationally expensive, after collecting OOV candidates from 1st pass decoding, we aim at extending the vocabulary and architecture of the word RNNLM, in order to enable re-scoring hypothesis containing OOVs efficiently, still with a single word-level RNNLM, avoiding extra computational overhead.

3.1.2.3 Dynamic vocabulary expansion within the WFST decoding framework

In OOV recovery, after estimating the LM probabilities of OOV candidates and expanding the lexicon to cover the OOV candidates, a second pass decoding is needed, which provably improves both overall decoding decoding results and OOV recovery results. However, within the WFST decoding framework [57] for conventional hybrid ASR, a closed-vocabulary is pre-determined when

[2]Although using a character LM alone solves the OOV problem in the end-to-end ASR setting, it's generally believed that character LM under-performs relative to word LMs for languages with a small character vocabulary like English [56].

building the decoding graph. Re-building the decoding graph with the new language model and lexicon is too expensive at run-time [3]. Therefore, methods which allow us to dynamically expand the vocabulary in an efficient way, without re-building the whole decoding graph, is highly favored. Generally, with the WFST decoding framework, there are two types of approaches to deal with this problem: The first type approaches compile the decoding graph on-the-fly with context dependency and look-ahead, or conduct vocabulary expansion and language model biasing on-the-fly [59, 60, 61, 62, 63]. The second type of approaches try to compile decoding graphs corresponding to different LM/lexicons separately, and then stitch pre-compiled graphs together dynamically. Compared with the fist type of approaches, it's simpler to implement and generally much faster, because we avoid re-compiling the whole decoding graph at run-time, especially in cases where most of the decoding graph is static, and only small parts in the lexicon/LM FSTs need to be changed. In order to make the computational complexity under control, some constraints on the context is usually needed. In this work we use the "Grammar Decoder" from Kaldi[4], which belongs to the second type of approaches and works with left-biphone models. With the grammar decoding framework, we'll be able to extend the LM and lexicon to include recovered OOV candidates from 1st pass decoding in an efficient way for 2nd pass decoding, with minimum computational overhead.

[3]Except when we apply some strong constraints on the phonetic context, e.g. context-independent phones [58], which makes it possible to simply inject the extra grammar FST representing OOVs/personal contacts into the decoding graph at reserved locations.

[4]The grammar decoder is developed by Daniel Povey. No published paper is available yet. Please refer to `http://kaldi-asr.org/doc/grammar.html` for details

3.2 Hybrid lexical model based OOV detection and candidate generation

In this section, we will review the hybrid lexical modeling (HLM) approach and introduce our implementation. Then we will review one typical method [2] of the "implicitly modeling OOV" approach. We then compare it with the HLM approach in OOV detection and recovery performance, as well as their impact to overall decoding performance, all under an LVCSR setting. Besides, we'll also compare with a trivial baseline approach of dealing with OOVs conventionally used in word based ASR, i.e. using a single (garbage) phone to model the pronunciation of an OOV in lexicon.

3.2.1 Hybrid lexical modeling

In modern ASR, Weighted Finite-State Transducer (WFST) [57] based decoding is the most widely used framework for ASR decoding. In WFST decoding framework, decoding is performed on a search space constrained by a decoding graph as an WFST, composed of acoustic model (AM), language model (LM), and lexicon all as FSTs. To be more specific, in Kaldi [23], the decoding graph is a composition of

$$H \circ C \circ L \circ G \tag{3.1}$$

where H is an FST containing the HMM definitions. Its output symbols represent context-dependent phones: its input symbols are acoustic modeling units (transition-ids); C is an FST representing the context-dependency: its input symbols represent context-dependent phones and its output symbols are phones; L is an FST representing the pronunciation lexicon: Its input symbols are phones and its output symbols are words; G is an FSA representing the language model. Therefore, a path through the decoding graph represents a mapping from a string of acoustic modeling units to a string of words. And the weights on the path encodes scores from HMM transition probabilities and LM, while the HMM emission probabilities will be given by the acoustic model during decoding.

In lexicon, conventionally, we use a single (garbage) phone to model the pronunciation of OOV. This results in a lexicon FST as shown in Figure3.1 for example (In the figure, each loop represents one lexicon entry. The outputs of each loop is a word and the input is its pronunciation. The loop at the bottom represents the single-phone pronunciation <oov> of the OOV token <unk>). Although simple, this approach cannot give good acoustic representation of OOVs, and usually gives poor OOV detection performance (which will be verified in experiments). Also OOV recovery won't be possible.

Following [42, 44, 43], we use the hybrid lexical model (HLM) approach to deal with OOV pronunciations in the lexicon. The first step is to first train a sub-word (here we use phone as the sub-word unit) language model, representing a probabilistic distribution over all possible phone sequences, on the phoneme sequences of all pronunciation entries from the lexicon (where all word-position-dependency has been removed). In order to keep the phonemic language model (PLM) compact, we limit the transitions to seen bi-grams

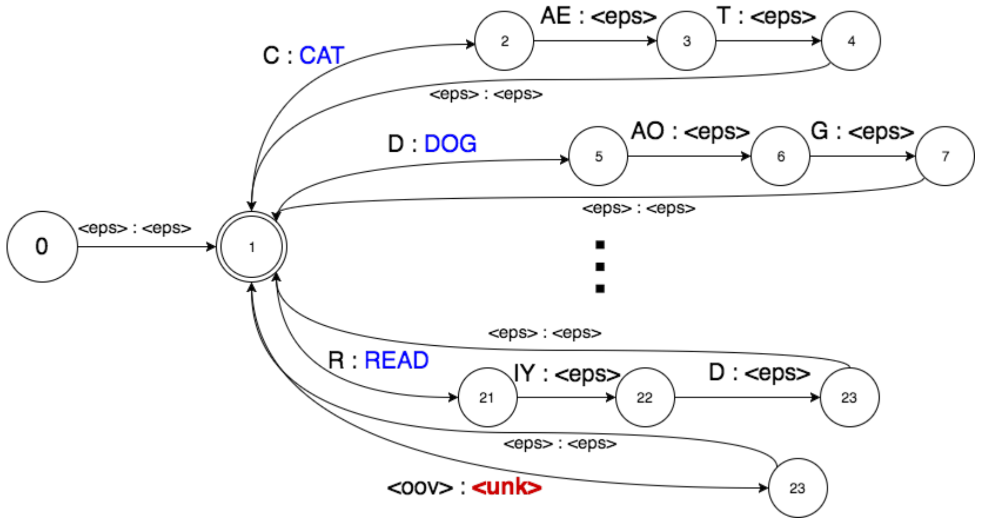


Figure 3.1: A lexicon as a Finite-State-Transducer (FST), with a single-phone OOV model.

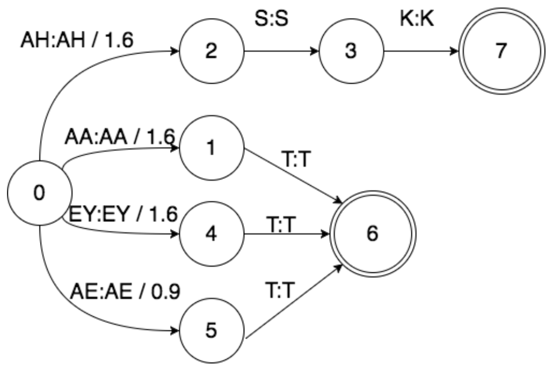


Figure 3.2: A simple phonetic LM represented as an Finite-State-Transducer (FST))

of phonemes. Then, we compile this PLM into an FSA (See Figure 3.2 for example).

However, there are two undesirable properties of this FST. 1. This FSA can accept/generate phoneme sequences with any length. But apparently a single phone doesn't make sense to model the pronunciation of an OOV, and allowing this will cause trouble when we decoding using the hybrid lexical

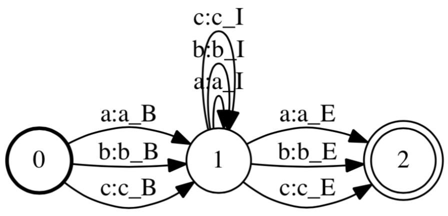


Figure 3.3: An FST accepting only phoneme sequences whose length is at least 2, and adding word-position-independent markers to phonemes

model[5]. 2. The PLM was trained on word-position-independent phonemes to keep PLM small, while we need word-position-dependent phonemes to represent OOV pronunciations. We solve this by composing the PLM FSA with a separate phoneme-sequence-constraint FST encoding 1. the constraint of minimum length (e.g. 2) of the generated sequence; 2. mapping from word-position-independent phonemes to word-position-dependent phonemes. An example is given in Figure 3.3, where the phoneme set composes of only three phonemes "a", "b", "c", the allowed minimum phoneme sequence length is 2, and Kaldi's word-position-dependency markers are used, where "B" stands for "at the beginning of a word", "I" stands for "inside a word", and "E" means "at the end of a word". It can be seen that by composing PLM FSA with this constraint FST, the final PLM FST is guaranteed to generate word-position-dependent phones, with length being at least 2.

Therefore we have created an PLM-FST representing a phonemic language model suitable for modeling OOV pronunciations. Then we can insert this PLM-FST to the L.fst (the lexicon fst), i.e. replacing the pronunciation of the

[5]To be more specific, the consequence of allowing the PLM to generate single phone pronunciations is that, lattices from decoding will contain a lot of <unk>'s with a single phone pronunciation, which is wrong.

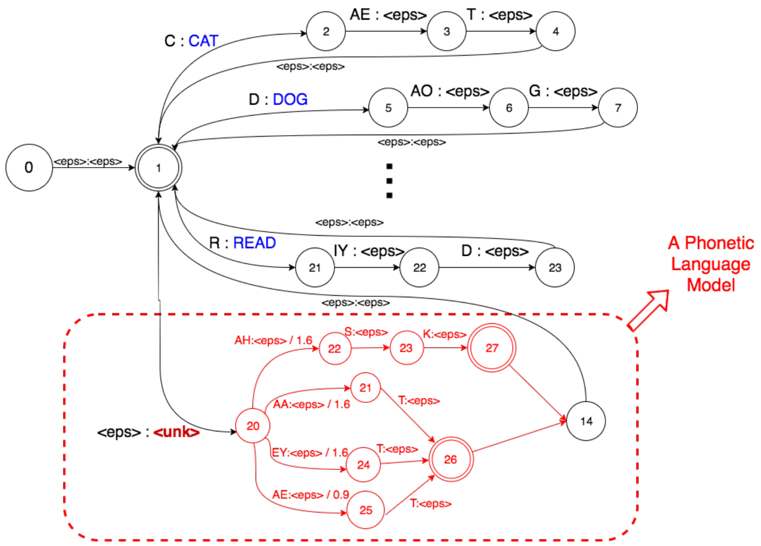


Figure 3.4: Embed the phonetic LM into the lexicon with the `fstreplace` operation.

<unk> word with a PLM (See Figure 3.4). And this is the so-called Hybrid Lexical Model (HLM). Here we use OpenFST's `fstreplace` operation, which perfectly fits this application. It performs the dynamic replacement of an arc in one FST with another FST. Here we replaced the arc representing the single-phone OOV lexicon entry, with the PLM-FST.

To summarize, similar as what's stated in [43] and [42], the decoding graph can be represented as the following FST operations:

$$H \circ C \circ (L \cup (G_p \circ C_p)) \circ G \tag{3.2}$$

where H, C and G are the same as in Eq. (3.1), G_p stands for PLM FSA, and C_p stands for the phoneme-sequence-constraint FST, which is major difference between our implementation and previous approaches, besides using the efficient `fstreplace` operation for building HLM.

Then we adopt this hybrid decoding graph in the standard ASR decoding pipeline. After decoding OOV occurrences will occur as <unk> tokens in the lattice, with their predicted pronunciations occurring as the one-best path of local phone lattices in the word lattice. After applying P2G (i.e. applying a G2P model inversely) to the generated OOV pronunciations, these phone sequences are mapped to character sequences, giving the recovered spelling of OOV candidates. We'll specify in the next part.

3.2.2 First pass decoding for OOV detection

During decoding, the entrance into the phoneme sub-graphs can be controlled by the unigram probability of <unk> in the N-gram language model, and we name it more intuitively as OOV cost (C_{oov}). This cost can be used to balance the contribution of the OOV phonemic grammar to the overall score of the utterance. To be more specific, when decoding with the hybrid decoding graph, boosting C_{oov} helps encourage hypothesis paths to go through the phoneme sub-graph modeling OOVs, and vice-versa. In the lattice generated from decoding, besides acoustic score, the score of a specific instance of OOV candidate is determined both by C_{oov} (the language model score of OOV) and the probability of the best phoneme path given by the phoneme sub-graph (the pronunciation score of OOV). In the proposed pipeline, we adjust C_{oov} according to the estimated OOV rate in validation data. Also we should limit the <unk> history by removing any N-grams where <unk> is the second-to-last word, and remove back-off probability from the lines where <unk> is the predicted word[6]. Then the next step is to compile the decoding graph (HCLG) with the AM, LM FST, and hybrid lexical model (HLM). After building the HCLG decoding graph with HLM, phoneme sub-graphs, which model the OOV pronunciations, occur in the decoding graph.

3.2.3 OOV candidates generation

We align arcs in the lattices with word boundaries, and collect all <unk> arcs with their pronunciations. This way we obtained all phonetic candidates of OOVs. Furthermore, we can get all word candidates by applying a P2G model (trained on the lexicon before decoding) to the pronunciations. There are several aspects that could affect OOV candidates generation:

- **Lattice-determinization** When we dump lattices, we have several choices in terms of determinizing the lattices which affects how many OOV pronunciation candidates we have per instance. If we determinize at the (word+phone) level but not at the word level (–word-determinize-lattice=true and –determinize-lattice=false in Kaldi's terminology), then for each <unk> instance in the word lattice, a phone lattice is left there, and thereby the lattice retains multiple pronunciation variants. With this, we will be able to do some more advanced post-processing like picking these phone lattices and clustering/merging them [44]. The second option is to determinize at both (word+phone) level and at the word level (–word-determinize-lattice=true and –determinize-lattice=true in

[6]This is important to guarantee there won't be too many <unk> arcs in the G.fst. Otherwise, the final HCLG composition may blow-up, since we use a phonemic LM to model pronunciation of <unk> in L.fst, meaning there'll be too many local phone graphs embedded in the word graph if there are many <unk> arcs in G.fst

Kali's terminology). By doing this we always take the one best path of a local phone lattice representing an <unk> instance, so that for each <unk> instance we have a unique pronunciation left. We experimented with both options, and decided to take the second one in our pipeline. Reasons are:

  - Not determinizing at word level could cause a blowup in the following lattice word alignment procedure, and we've found doing this causes slight overall WER performance (considering tuning).
  - Even if we determinize at both word level and word+phone level, we can still obtain <unk> arcs with similar locations but different pronunciations. The reason is that these arcs belong to different <unk> instances, i.e. these <unk> arcs belong to different word sequences in the lattice.

- **P2G**: When applying P2G, we have noticed that taking more variants besides the top variant doesn't help WER performance or OOV recovery performance (from one-best hypothesis). For search-related applications (e.g. OOV keyword search), which is not the focus of this dissertation, taking more variants are usually helpful. [64].

3.2.4 A baseline approach of implicitly modeling OOV: the IBM Hybrid LM + classification method

In this section, we introduce one OOV detection method from the family of "implicitly modeling OOV" approaches involving a hybrid LM and a discriminative classifier [2], which we'll compare with our HLM-based OOV

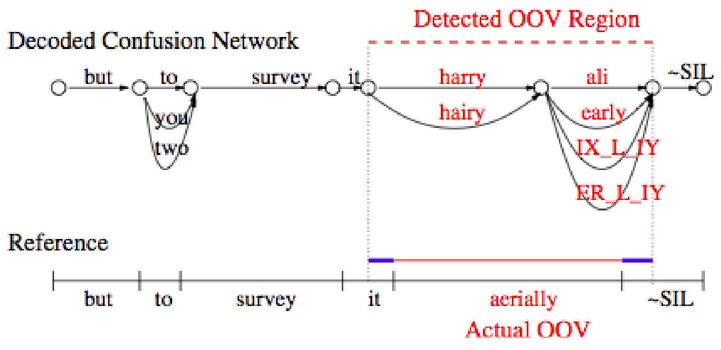


Figure 3.5: Confusion network example (from [2])

detection method in the experiments. Basically, this method decode with a hybrid language model containing both words and fragments (word-pieces). The fragments, as filler models, are variable length phone sequences that are introduced to implicitly modeling OOVs by absorbing them.

After obtaining first pass decoding results on development data as confusion networks (see Table 3.5), for each bin, we extract the following three features:

$$\texttt{Fragment Posterior} = \sum_{f \in t_j} p(f|t_j) \tag{3.3}$$

$$\texttt{Entropy} = -\sum_{w \in t_j} p(w|t_j) \log p(w|t_j) \tag{3.4}$$

$$\texttt{Max. Posterior} = \max_{w \in t_j} p(w|t_j) \tag{3.5}$$

to train a maximum entropy (MaxEnt) classifier, with three categories: OOV, in-vocabulary error (WErr), and in-vocabulary correctly decoded word (WCorr). Training labels are obtained by aligning the hypothesis with the reference text on development data. The rationale is that: An OOV word does not match well with IV words, so that it's more likely to have higher posterior of fragments; Also, there is likely to be more confusion in the bin containing an OOV, so that entropy is higher and maximum posterior is supposed to be smaller. During evaluation, OOV detection is conducted by decoding test data, generating confusion networks and then applying the classifier to each bin to give predictions.

3.2.5 First pass OOV recovery

For each recovered OOV candidate, we assign a unique ID and add it to the word list. Then we replace the word ID of all <unk> arcs by the ID of OOV candidate recovered from applying P2G to its pronunciation. When collecting the <unk> arcs, we indexed each arc by its starting state + ending state. This information is carried along when we map the pronunciation on the arc to a recovered OOV candidate, so that when want to insert back recovered OOVs into lattices, we can easily retrieve a specific arc from lattice and replace its word-ID by the recovered OOVs' ID. On top of this lattice, we can optionally do re-scoring with a large N-gram LM with OOV candidates inserted. Or we can just take the one-best hypothesis from the lattice as decoding result, which contains recovered OOVs.

3.2.6 Experiments

3.2.6.1 Experimental setup

We use Switchboard conversational speech recognition task as the test bed. We use the original 300h training data with the 303K lexicon covering all words in training data, to train an AM (TDNN-LSTM [28]) and an n-gram LM. Here is the information of two test sets:

Test set	# utts.	# Tokens	# utts. w/ OOVs	# OOVs
Eval2000 (3.8h)	4.5K	40K	282	347
RT03 (6.2h)	8.4K	71K	356	455

For our HLM approach, we train the phonemic language model on the 303K lexicon, compile hybrid decoding graph, and decode. To evaluate performance, we align the hypothesis with reference text (Basically, for each word from reference text, we align it with the max-overlapped word in hypothesis text, and regard "<unk>" in hypothesis as positive predictions), and then OOV detection results were reported by measuring precision, recall, false alarm rate, and F1:

$$\begin{aligned} Recall &= \frac{\text{\# OOVs detected}}{\text{\# OOVs in reference}} \times 100\% \\ False\ Alarm\ Rate &= \frac{\text{\# OOVs reported - \# OOVs detected}}{\text{\# IVs in reference}} \times 100\% \\ Precision &= \frac{\text{\# OOVs detected}}{\text{\# OOVs reported}} \times 100\% \\ F1 &= \frac{2 \times Precision \times Recall}{Precision + Recall} \end{aligned}$$

After mapping the <unk> tokens to recovered OOVs by P2G, we can also compute the character error rates (CER) on the OOV tokens to evaluate the 1st pass OOV recovery results:

$$CER = \frac{\text{\#Substitution + \#Deletion + \#Insertion errors}}{\text{\# Characters in all ref. tokens}} \times 100\% \quad (3.6)$$

For the baseline IBM method, we first train a tri-gram phonemic LM using the lexicon, and then sample 1 million fragments from a tri-gram phonemic LM, putting them into both the lexicon and word LM to build a hybrid LM, and we adopt the probabilities of phone sequences assigned by the phonemic LM as the unigram probabilities in the LM [64]. Then we decode development utterances to generate confusion networks, and train the MaxEnt classifier. Then we decode test data and apply the classifier to get predictions on each token. After aligning the hypothesis with reference text we report the same OOV detection metrics as we did for the HLM approach. Note that we can also compute the character error rates (CER) on the OOV tokens to get an idea of how the hybrid LM approach perform in OOV recovery, since each OOV token will be aligned with either an IV word or a fragment. Results on Eval2000 were produced by a MaxEnt classifier trained on RT03's utterances, and vice versa.

We also test with the trivial method of modeling OOVs with a single phone in the lexicon, with the same OOV detection metrics. OOV CERs are not reported since it's not possible to recover OOVs with this approach.

3.2.6.2 Experimental results

3.2.6.2.1 OOV detection OOV detection results as ROC curves are shown in Table 3.6 and Table 3.7. For the Hybrid Lexical Modeling (HLM) approach and the single-phone OOV modeling approach, we sweep through C_{oov} (unigram probability of <unk>) to get ROC curves.

For the IBM method, we sweep through the score threshold of the MaxEnt

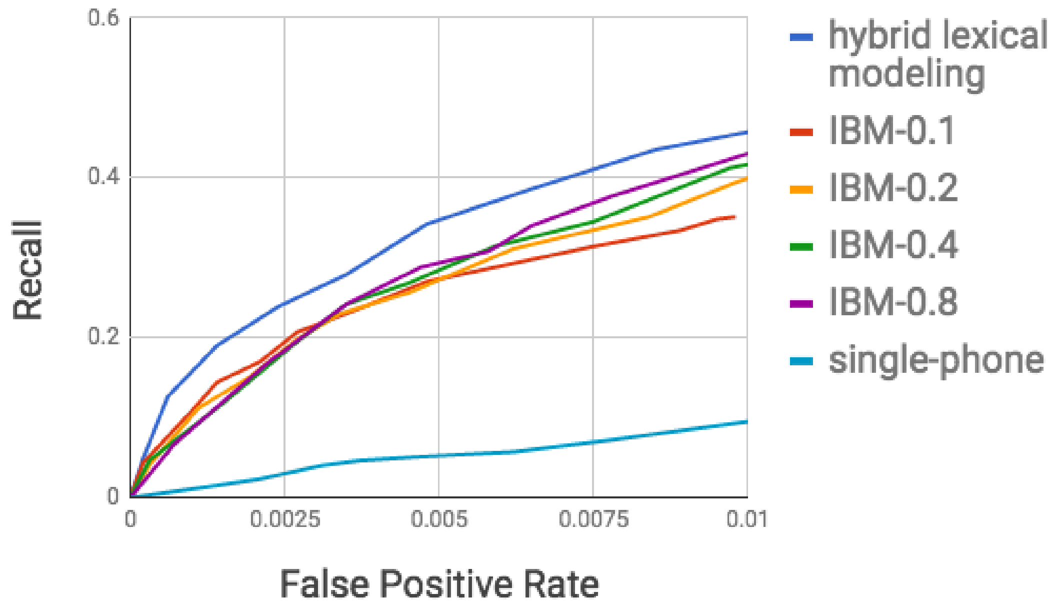


Figure 3.6: OOV detection results on eval2000

classifier to get the ROC curves. We also tune the fraction of unigram probability mass allocated to fragments in hybrid LM, e.g. "IBM-0.1" means the unigram probability mass allocated to fragments is 0.1.

It can be seen that on both data sets, as expected, increasing the fragment fraction in LM results in better OOV detection performance for the baseline IBM method (higher ROC curve). And the HLM approach out-performs the baseline IBM method, which is interesting since this implies a dynamic OOV filler model, which generates OOV hypothesis dynamically, is better than a classifier-based approach using word-pieces as statically-generated OOV fillers, which requires labeled training data. Besides, the trivial single-phone approach does a bad job of detecting OOVs, as expected.

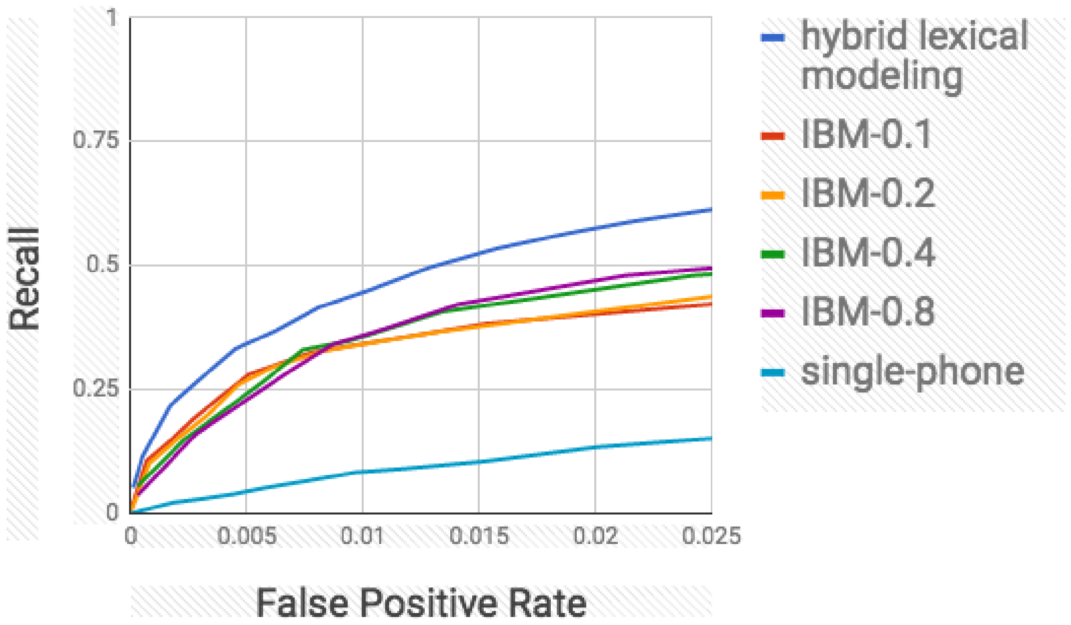


Figure 3.7: OOV detection results on RT03

3.2.6.2.2 First pass OOV recovery Here we report first pass OOV recovery results measured on the setup with the highest $F1$ score for the HLM and baseline IBM approaches. Then we measure the CER between each ref. and hyp. token on all reference OOV tokens with True Positive detection. Because of the nature of acoustic variants, it's usually hard to recover OOVs' spelling exactly. However approximately recovered OOVs still help improve human readability. So, as for performance measurement, we are going to use CER as the major metric. For experiments in the next section where we focus on OOV recovery, we'll also report WER on OOVs.

Table 3.1: OOV recovery performance on Eval2000

	Max. detection F1	# True Pos.	CER (%)
IBM method	0.326	151	49.8
hybrid lexical model (HLM)	0.355	119	32.7

Table 3.2: 1st pass OOV recovery performance on RT03

	Max. detection F1	# True Pos.	CER (%)
IBM method	0.266	128	54.7
hybrid lexical model (HLM)	0.319	151	40.0

From results reported in Table 3.1 and 3.2, it can be seen that overall, the HLM approach performs better than the IBM baseline, on both F1 score and CER on true positive tokens, probably because of the bigger freedom of modeling OOV spellings with a phonemic language model in the HLM approach, than using a fixed set of word fragments in the IBM approach. Remember that, the IBM approach is based on hybrid language models, with millions of word fragments sampled from a phonemic language model inserted to the word LM. Therefore, it can be seen that both HLM and the IBM approaches model OOVs with sampled paths from a phonemic language model. The only major difference is that, HLM generates samples in an online fashion during decoding, and the IBM approach generate a large pool of samples (fragments) before decoding, and look for the "closest match" for OOVs encountered during decoding. Therefore it's understandable that the former gives better recovery performance.

Note that the CERs here are measured on different sets of tokens (with true positive detection) making the comparison not fair. The reason why we did this is because OOV detection is the focus of this section. In the next section where we focus on OOV recovery, we'll use a different metric called OOV CER, measuring CER on the same set of reference OOV tokens.

3.2.6.2.3 Overall decoding performance Here we investigate the impact of different OOV detection approaches on the overall decoding performance. This is important since we never want to sacrifice recognition performance on IV words while trying to detect/recognize OOVs. Results are shown in Table 3.3 and Table 3.4. Again we also include the single-phone baselines. "single-phone-default" means we use the default OOV LM unigram probability estimated from the LM training text ($2.3 \times e^{-6}$), which is exactly the default swbd recipe, which has zero OOV recall. "single-phone-tuned" means the OOV unigram probability is tuned to optimize $F1$ score for OOV detection.

Table 3.3: Word error analysis on Eval2000

	Sub.	Del.	Ins.	Overall Err.
single-phone-default	10.0	3.5	2.2	15.6
single-phone-tuned	9.6	4.2	1.9	15.7
IBM method	10.0	4.0	1.7	15.7
hybrid lexical model (HLM)	10.0	3.6	2.0	15.6

Table 3.4: Word error analysis on RT03

	Sub.	Del.	Ins.	Overall Err.
single-phone-default	11.6	4.9	2.2	18.7
single-phone-tuned	11.2	5.7	2.0	18.9
IBM method	11.5	5.4	1.7	18.7
hybrid lexical model (HLM)	11.6	5.1	2.1	18.7

It can be seen that, compared with the baseline (single-phone-default), neither approach degrades the overall performance much, and the HLM approach matches the baseline performance on both test sets. But both IBM and HLM methods consistently give more deletions errors. To figure out the reason, we show some examples here:

in hickory $\rightarrow$ inhekery

so good to hug her $\rightarrow$ soakatoger

over heals $\rightarrow$ overhealed

randy is $\rightarrow$ brandy's

We can see that the reason is that many IV words were eaten up by OOV false alarms. This motivates us to put more efforts on better OOV recovery, improving recovery accuracy and discouraging false alarms like these.

3.3 OOV recovery with grammar decoding and open vocabulary RNNLM re-scoring

As described in the last section, we adopted a Hybrid Lexical Model (HLM) + Phoneme-to-Grapheme (P2G) OOV detection pipeline to detect potential OOVs, and generate their candidate forms. Also we demonstrated initial results of OOV recovery from first pass decoding. In this section, we'll focus on improving OOV recovery accuracy and discouraging false alarms, by better estimating OOV LM probabilities, efficient second pass decoding and open-vocabulary RNNLM re-scoring.

3.3.1 Efficient second pass OOV recovery with dynamic vocabulary

There are mainly two factors that motivate us for a second pass OOV recovery, based on observation from the first pass decoding results:

- **Calibrating OOVs' weight**

  Before the first pass decoding, we adjusted the unigram probability of

<unk> (C_{oov}, for which we have shown impact on overall decoding and OOV detection results.) in the language model to be close to or a bit larger than the OOV rate estimated on development data. However, it's quite possible that the empirical OOV rate (see Eq. (3.7)) estimated from first pass decoding results, could disagree with the estimated OOV rate which helped us to decide <unk>'s unigram probability in first pass decoding. Second pass decoding enables us to calibrate the OOV preference by taking the "evidential" test OOV rate into consideration, e.g. if we find the empirical OOV rate in the test utterances is higher than expected, we can boost weight of the OOV grammar during 2nd pass decoding, and thereby re-generate lattices better accounting for OOVs.

- **Calibrating OOVs' LM score estimation**

  As shown in the end of last section's experiments, lots of recognition errors on IV words are caused by OOV false alarms. Also, if there's an OOV in the reference, the OOV candidate in hypothesis could get stuck with nearby tokens to cause errors like the following example (the red word is OOV):

  - *Reference:* has trabajado en una oficina

    1st pass recovery: hastrabajado en una oficina

  So the goal here is to design a proper unigram distribution of OOV candidates, to discourage OOV false alarms or bad OOV candidates like "hastrabajado".

The Grammar decoder frameworks enables us to use dynamically created grammar (LM)s and graphs that we want to compile quickly in test time, i.e. at run-time we may want to add extra words like a contact list to the lexicon. The framework is specifically designed for applications with a compelling need to pre-compile the HCLG.fst for various sub-parts and have them dynamically stitched together (typically to avoid recompiling large graphs at run-time). OOV recovery is a perfectly suitable application here. For the OOV recovery scenario, when we build the arpa LM used in 1st pass decoding into G.fst, we replace <unk> with a nonterminal symbol, e.g. #nonterm:oov, as a "place-holder" for the OOV grammar, i.e. OOV candidates obtained from 1st-pass decoding with HLM and P2G. In 2nd pass decoding, rather than inserting the OOV grammar into the original LM and building the big decoding graph again, which takes much time and could potentially blow-up in memory, we only need to build the OOV grammar into FST and then into HCLG (which is fast): $H \circ C \circ L_{oov} \circ G_{oov}$, with H representing the AM, C representing the context dependency, L_{oov} representing the lexicon of recovered OOVs, and G_{oov} presenting the OOV grammar. At decoding time, the grammar decoder will dynamically stitch the small HCLG together with the original HCLG (with a place-holder reserved for the small one).

Sometimes we need to re-score lattices from the first pass decoding with a bigger N-gram LM. In this case, we need to recover the composite LM used in grammar decoding[7], i.e. by inserting the OOV grammar FST into the original

[7]This didn't exist since we avoided building the composite LM and decoding graph by grammar decoding. However for re-scoring purpose, we need to build the single composite LM equivalent to the two LMs used in grammar decoding.

LM FST[8]. Then we can subtract old LM scores from the lattices using this composite LM FST, and re-score the lattices using the second pass LM (also augmented with the extra OOV grammar, by extending the ARPA file or extending the LM FST again with `fstreplace`.)

Assigning a proper unigram distribution over the OOV grammar is super important, and is the main goal of second pass decoding. On one hand, we have a second chance to calibrate the overall OOV "preference", by adjusting the amount of probability mass allocated to OOV grammar (called $P(OOV)$), which contains all recovered OOV candidates. The way we formulate $P(OOV)$ is, we first estimate the empirical OOV rate R_{oov} from the lattices statistics of 1st pass decoding (over the whole test set). Basically, by collecting word IDs from all lattice arcs in test data, R_{oov} is estimated as:

$$\boldsymbol{R}_{oov} = \frac{\#.\ <unk>\ \ arcs\ from\ lattices}{\#.\ all\ arcs\ from\ lattice} \tag{3.7}$$

Then we scale the OOV rate by a scalar α to get the desired probability mass of OOV grammar for second pass grammar decoding:

$$P(OOV) = \alpha R_{oov} = \alpha \times \frac{\#.\ <unk>\ \ arcs\ from\ lattices}{\#.\ all\ arcs\ from\ lattice} \tag{3.8}$$

By setting $\alpha = 1$, it means we want the unigram probability mass of the OOV grammar to equal the empirical OOV rate from 1st pass decoding. Adjusting α will directly affect the OOV recovery results and overall decoding results. We'll explore this experimentally in the next section.

[8]Here we use OpenFST's `fstreplace` operation to replace the <unk> arc with the OOV grammar FST, and use `fstdeterminizestar` to determinize it.

On the other hand, given a fixed probability mass assigned to OOV grammar, we could use different approaches to estimate the unigram distribution. The overall goal is to boost OOV candidates which "make sense", and discourage OOVs which are likely garbage, e.g. a 20-character-long word for English. It's tricky to achieve this goal, since we don't know the ground truth in advance. However we have some prior knowledge, e.g. in first pass decoding, the pronunciation scores of each OOV candidate in lattices come from a phonemic language model (PLM). So PLM score is a natural choice. Suppose the formula we use to assign unigram distribution for OOV grammar is $\mathcal{F}(w)$ (where w is an OOV candidate), we consider the following options:

- **Phonemic Language Model scores**: This is the most natural option as we mentioned, used in the 1st pass decoding. When generating the lattices during 1st pass decoding, the PLM score of each <unk> arc was implicitly encoded in the AM score as the pronunciation probability.

- **Character Language Model scores**: Instead of looking at the PLM scores, we can train a character LM on IV words. For some irregularly spelled languages, or languages whose grapheme set is much larger than its phoneme set, this is more informative than PLM. Furthermore, as we are not directly using the character LM to generate character lattices, we can use a stronger neural LM. In our experiments, an character level RNNLM (CharRnnlm) is used.

- **Uniform distribution**: We just assign a constant probability to all OOV candidates. This is potentially a bad option, which will be investigated as a baseline.

- **Empirical frequency**: For each OOV candidate, we can use the empirical frequency of its pronunciation, meaning the number of <unk> arcs with a particular pronunciation found in lattices, normalized by the total number of <unk> arcs.

- **P2G scores**: When applying P2G to the phone sequence to get the recovered OOV word, we usually take the top P2G candidate, and thereby we can explore using the P2G score given by the joint-sequence G2P model on this candidate as the unigram LM score also.

Note that we have chosen not to estimate unigram probabilities with LM probabilities of similar IVs, or even estimating bi-gram probabilities like what was pursued in [52, 53, 54], since we don't have the ground-truth spelling of the OOV candidates, and the OOVs are infrequent, so that it's hard to collect contextual statistics to robustly estimate higher order LM probabilities for OOVs. Some evidence will be provided in our experiments. Basically, among all candidate options of $\mathcal{F}(w)$, only "empirical frequency" is based on "global" statistics of OOVs. All the others are based on features from a particular OOV candidate's spelling or pronunciation. And in our experiments, we'll show that "empirical frequency" itself works not well, though it helps sometimes, when composed (by interpolation or multiplication) with phone/character LM scores. We'll investigate this in detail in the next section.

Suppose the set of all recovered OOV candidates is $\mathcal{L}$, for each OOV candidate w, we have determined to use function $\mathcal{F}(w)$ to assign the unigram probability. Assuming we have normalized $\mathcal{F}(w)$ over $\mathcal{L}$ to satisfy the summing-up-to-one constraint, we need to further scale them by a scalar β

to get the unigram probabilities we'll assign to these OOV candidates in the OOV grammar:

$$U(w) = \beta \mathcal{F}(w). \tag{3.9}$$

The reason is that we don't want to re-scale the unigram probabilities of the in-vocab words in the LM (since they are consolidated into G.fst), and we want to make sure the "relative" probability mass taken by the OOV grammar is $P(OOV)$ as specified by the user. It's not an problem that after adding OOV candidates, all unigram probabilities don't sum up to one, since this can be compensated during scoring time by adjusting the LM weight. In short, we need β to satisfy:

$$P(OOV) = \frac{\sum_{w' \in \mathcal{L}} U(w)}{1 + \sum_{w' \in \mathcal{L}} U(w)} = \frac{\beta}{1 + \beta} \tag{3.10}$$

where the 1 in the denominator means the original unigram probability mass of all in-vocabulary words is 1 [9]. This gives:

$$\beta = \frac{P(OOV)}{1 - P(OOV)} \tag{3.11}$$

[9] To be more precise, this should be 1 minus the unigram probability of <unk> in LM. But in practice the unigram probability of <unk> is set to be close to zero. Even if it's not close to zero, this effect can be compensated by adjusting α. So we ignore this to keep the equation simple.

3.3.2 Word-level RNNLM rescoring with dynamic vocabulary expansion

The lattices generated with grammar-decoder contains words outside the vocabulary used when training the RNNLM. In order to re-score lattices with RNNLM, we need to expand the vocab of RNNLM first, so that it could assign reasonable scores to lattice paths/nbest-lists containing OOV candidates. However word RNNLM usually operates on a closed vocabulary. Here we specify two factors in Kaldi-RNNLM's design that enables inference on OOVs, resulting in open-vocabulary word RNNLM rescoring.

- **Embedding factorization and sparse feature representation with sub-word features** Suppose the training vocabulary size is N, and the embedding dimension is M, in terms of the word embedding, there are generally two options for word feature representation. One option is to simply use 1-of-N (one-hot) encoding of the words [65], and this results in an $N \times M$ dense matrix $W_{N \times M}$ as the word embedding, which is also a trainable weight matrix as part of the input layer in the neural network. The other option is to use a sparse feature representation of each word. Kaldi-RNNLM follows the second option. Two types of features are used. On word level, three features are used: word-identity as one-hot presentation (only for frequent words); unigram probability of the word (in log space); word length. In order to better represent rare or OOV words, sub-word level features are used. Basically we extract character n-grams of a word, e.g. given a word (e.g., nice), we break it into a combination of character n-grams (e.g </s> n, </s> ni, nic, ice, ce <s>, e <s>), and use bag-of-word (BOW) representation of the character n-grams as sub-word features. Suppose we have K features, (we always have $K > N$ since sub-word features are used), then feature representation of all N words is encoded into a sparse matrix $F_{N \times K}$ (each row is a sparse vector representing features for a word). This matrix is fixed and not trainable. And the word embedding $W_{N \times M}$ can be factorized as

$$W_{N \times M} = F_{N \times K} Y_{K \times M} \tag{3.12}$$

where $Y_{K \times M}$ is a dense matrix representing the feature level (or say, sub-word level) embedding matrix, jointly trainable with the neural network. We call this "feature embedding", or sub-word embedding. We can see in the word embedding $W_{N \times M}$, each row is a word-level embedding vector, obtained by summing over embedding vectors corresponding to each feature.

- **Input and output embedding typing** In a conventional RNNLM setting, the vocabulary is usually pre-defined during training, and the vocabulary cannot be changed once determined, since the output embedding matrix is fixed. In Kaldi-RNNLM [66], input and output embedding (weight matrices) are shared in the neural network. This weight tying architecture was proposed in [67], which has shown that tying embedding significantly reduces the number of parameters, and also reduces perplexity on both the validation set and the test set, indicating less over-fitting as a result of reduction in the number of parameters. Though not proposed in the original paper, we realized that this architecture is a good solution for the OOV problem in RNNLM rescoring, in a sense that it'll be very flexible to extend the vocabulary at test time. To be more specific:

Because of incorporating sub-word information in the feature presentation, and the embedding-tying network architecture, Kaldi-RNNLM framework is capable of dealing with OOVs. Given L OOV candidates recovered from first pass decoding, we go through the following procedures:

- **Extract sub-word features** of recovered OOVs, and augment the original feature matrix: $F_{N\times K} \rightarrow F'_{(N+L)\times K}$.
- Replace both the input and output word embeddings in the RNNLM by the augmented word embedding matrix[10]:

$$W'_{(N+L)\times M} = F'_{(N+L)\times K} Y_{K\times M} \tag{3.13}$$

  Note that this will make the output probabilities of the RNNLM not sum up to one, and Kaldi-RNNLM supports optionally adding a normalization step to the RNNLM outputs during inference, so that they sum up to one exactly.

- Augment N-gram LM FST (G.fst): During Grammar decoding, we built the OOV grammar into HCLG level and then dynamically stitched with the original HCLG when decoding. During RNNLM lattice re-scoring, we need to provide an LM FST composed of the original N-gram LM

[10](Note that the trained feature embedding $Y_{K\times M}$ is not changed)

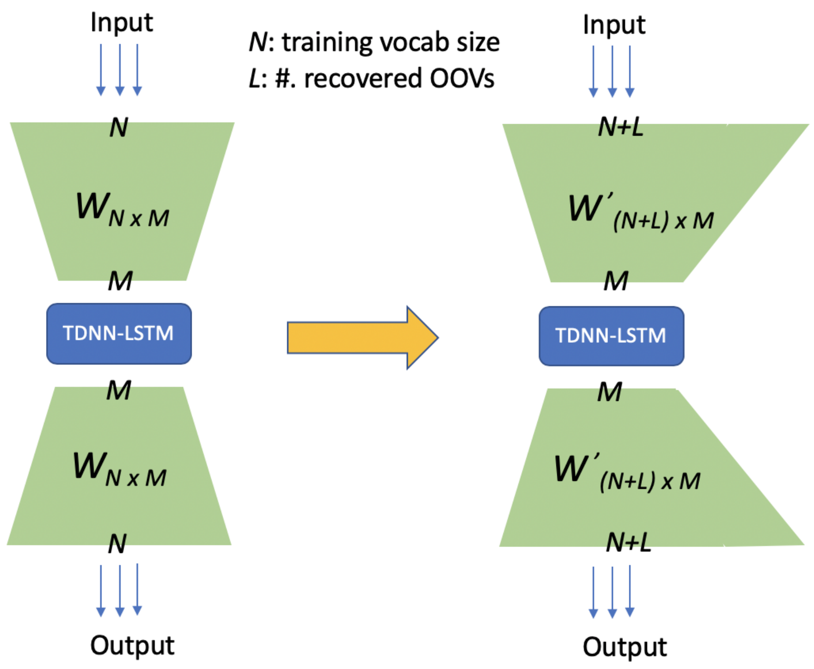


Figure 3.8: Dynamic vocabulary expansion for a word-level RNNLM

with OOV grammar embedded [11], in order to subtract N-gram LM scores from lattices. It's done by using OpenFST's `fstreplace` to replace all <unk> arcs in the original N-gram LM FST [12] by the OOV grammar FST, followed by `fstdeterminizestar` to determinize the final FST.

The whole process is also presented in Figrue. 3.8. Then we can apply the RNNLM with expanded-vocabulary and the augmented G.fst, on n-best lists/lattice paths containing those L OOV candidates for rescoring, as we normally do.

[11]Compiling HCLG with this LM and then decode, would give the same results as Grammar decoding in theory, though much slower.

[12]There won't be many, since we have limited <unk> histories when compiling the original N-gram LM into FST

3.3.3 Experiments

3.3.3.1 Evaluation metrics

As for performance metric, since we aim at both improving OOV recovery accuracy and discouraging false alarms, i.e. we care a lot about the overall decoding performance besides OOV recovery performance, we will comprehensively measure WER and CER on both OOVs and whole decoding hypothesis. When measuring OOV WER and CER, we first align all reference tokens with max-overlapped hypothesis tokens as we did for OOV detection experiments, and then compute WER and CER (Eq.(3.6)) of all ref/hyp token pairs. The OOV WER is defined simply as one minus OOV recognition accuracy, which is the number of exactly recovered OOV tokens divided by the total number of OOV tokens. This metric should look pretty bad in most places, since recovering OOVs exactly is a very hard task. For most experiments, the baseline is obtained by decoding with the same limited vocabulary, N-gram LM and RNNLM, but without any OOV recovery/vocabulary expansion procedure applied.

3.3.3.2 Experiments on Spanish

We first evaluate the proposed OOV recovery pipeline on Spanish. The reason is that Spanish is a regularly spelled language, with almost rule-based G2P. It creates a easier test condition for evaluating our OOV recovery method, ruling out the potential errors from the P2G process. In the next session we'll evaluate on more challenging conditions: read/conversational English.

We use the Heroico (LDC2006S37) dataset for evaluation[13]. The Heroico training set (11hrs) was recorded at Heroico Colegio Militar (HEROICO), containing read speech and free-response answer speech. Devtest is of the same condition with training set, but with read speech only. Test sets (native and nonnative) are recorded at USMA (US military academy), by native & nonnative speakers non-seen in training/devtest datasets. Also the test sets from USMA were collected with different microphones (head-mounted and throat microphones). Most of the content of read speech of the Devtest and Test sets all come from the same pool of 724 distinct sentences, all of which are short, simple sentences used in typical language learning scenarios. That's why the dataset statistics in Table 3.5 show that three tests have identical statistics of OOV rate, number of word types, and OOV type rate. Basically, the three test sets only differ in speaker identities / dialects and acoustic conditions. This is a good test-bed for OOV recovery algorithm to see how speaker identities / dialects and acoustic conditions affect its performance.

The acoustic model is a 11-layer TDNN-F [68] model trained with LF-MMI [27] on context-dependent phoneme units, trained on 11 hours of acoustic training data. The language models are a 3-gram model trained on transcripts, and a 3 layer TDNN-LSTM RNNLM trained on the same data[14]. For the lexicon, We randomly sampled 45K infrequent words (with count <= 10 in training data) to remove from the 91K reference lexicon from the Heroico recipe, so that the resulted lexicon size is about 46K (one pronunciation for each word). Then we trained a G2P model using this lexicon. The number

[13]Kaldi recipe used can be found at
`https://github.com/kaldi-asr/kaldi/tree/master/egs/heroico/s5`

[14]AM and LM recipes can be found at `https://github.com/kaldi-asr/kaldi/tree/master/egs/heroico/s5`

of tokens and OOV rate w.r.t the sampled lexicon of three test sets (Devtest, Native, Nonnative) are as the following:

Table 3.5: Basic statistics of Heroico test sets, w.r.t the sampled vocabulary.

Test set	#. Tokens (K)	OOV rate (%)	#. Word Types	OOV Type Rate (%)
Devtest (1.04h)	7650	20.6	473	33
Native (1.05h)	7498	20.6	473	33
Nonnative (1.21h)	9215	20.6	473	33

3.3.3.2.1 Overview of 2nd pass recovery and open-vocab RNNLM rescoring's impact on OOV recovery performance First, we want to conduct some initial investigation on how much gain we have from the proposed 2nd pass Grammar-decoding v.s. 1st pass OOV recovery already investigated in the last section. For both cases (with 1st pass recovery only or with 2nd pass recovery), we conduct open-vocab RNNLM re-scoring (both lattice and n-best re-scoring), in order to independently verify whether it helps. For 2nd pass recovery we use PLM and CharRnnlm (character RNNLM) to estimate unigram probabilities of OOV candidates. They are the two most basic options, and we'll explore more options later on. We evaluate OOV/Overall WER/CER on three test sets("Devtest", "Native", "Nonnative"). So there are in total 12 tables:

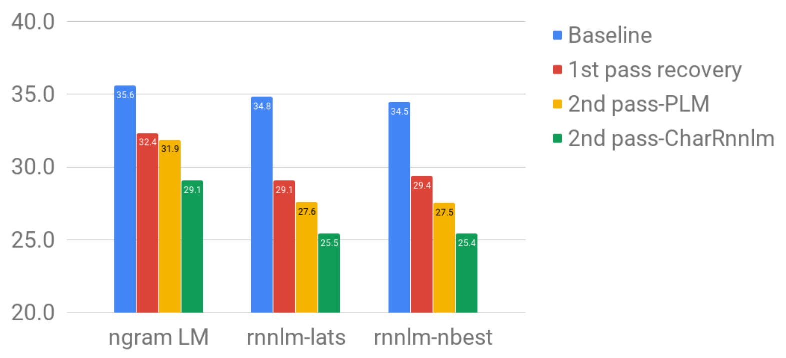


Figure 3.9: Overview of 1st & 2nd pass OOV recovery's impact on overall WER, on "Devtest" subset

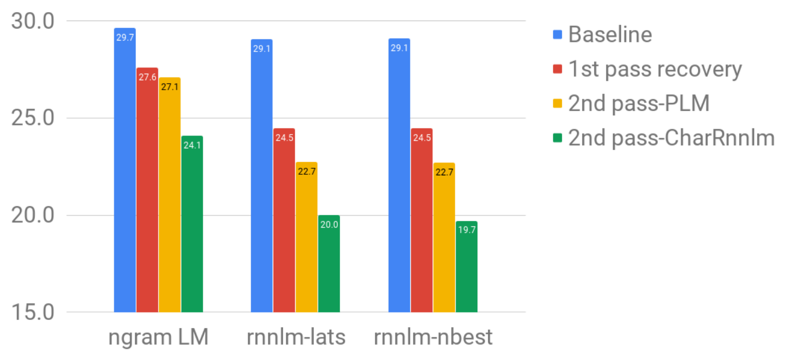


Figure 3.10: Overview of 1st & 2nd pass OOV recovery's impact on overall WER, on "Native" subset

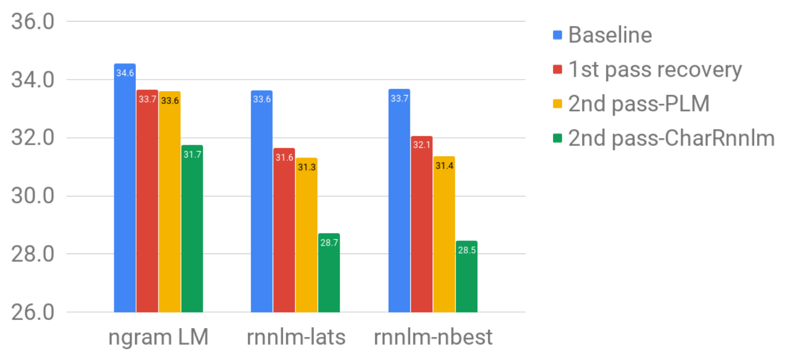


Figure 3.11: Overview of 1st & 2nd pass OOV recovery's impact on overall WER, on "Nonnative" subset

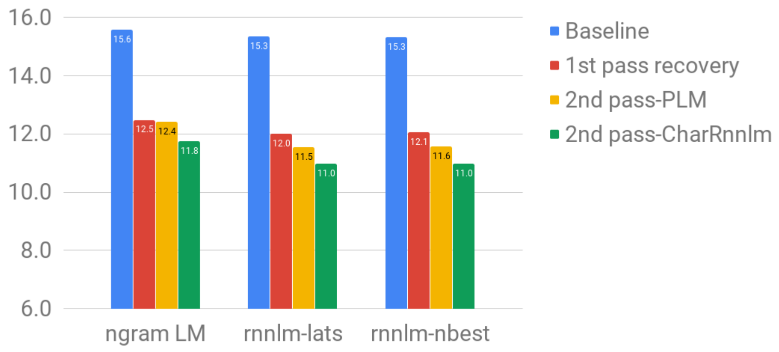


Figure 3.12: Overview of 1st & 2nd pass OOV recovery's impact on overall CER, on "Devtest" subset

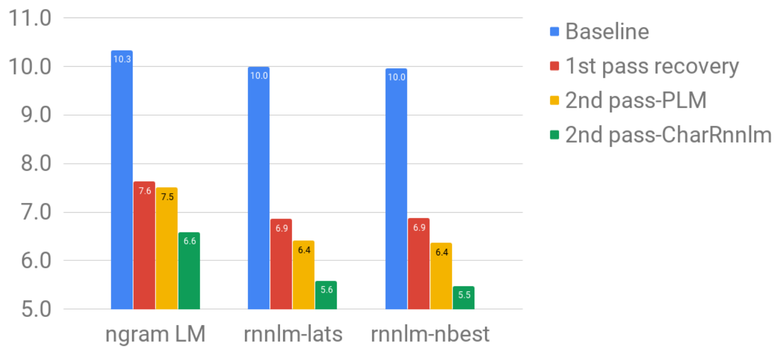


Figure 3.13: Overview of 1st & 2nd pass OOV recovery's impact on overall CER, on "Native" subset

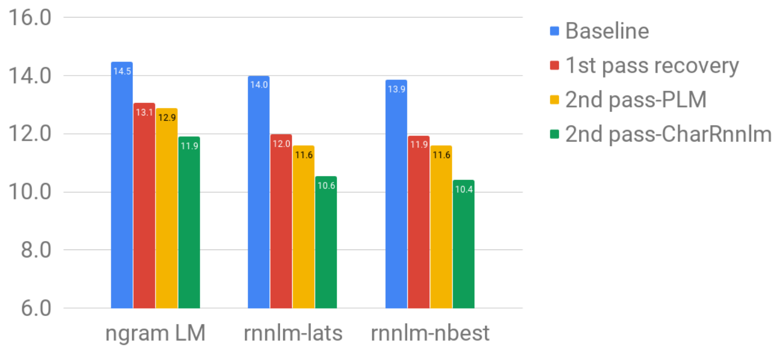


Figure 3.14: Overview of 1st & 2nd pass OOV recovery's impact on overall CER, on "Nonnative" subset

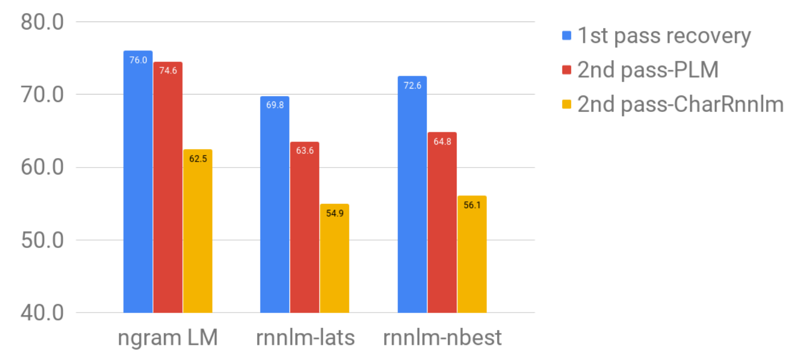


Figure 3.15: Overview of 1st & 2nd pass OOV recovery's impact on OOV WER, on "Devtest" subset

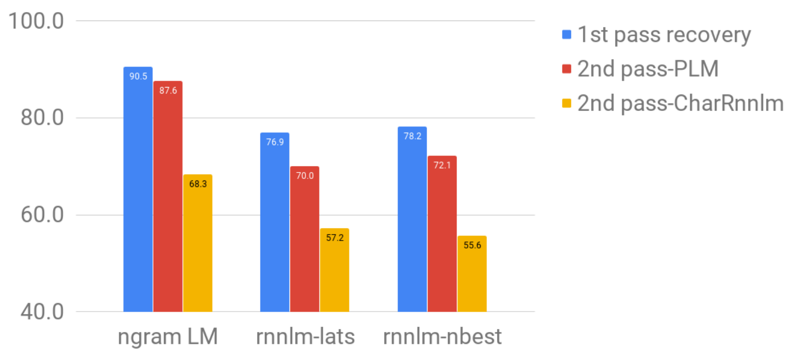


Figure 3.16: Overview of 1st & 2nd pass OOV recovery's impact on OOV WER, on "Native" subset

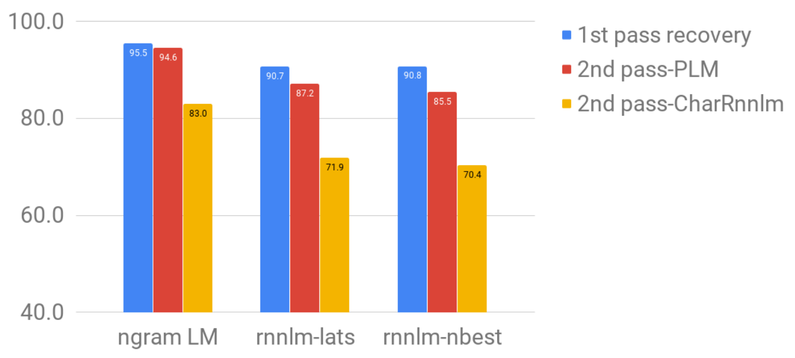


Figure 3.17: Overview of 1st & 2nd pass OOV recovery's impact on OOV WER, on "Nonnative" subset

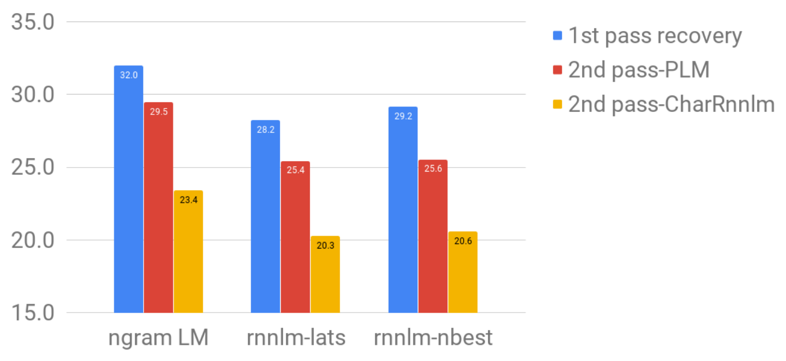


Figure 3.18: Overview of 1st & 2nd pass OOV recovery's impact on OOV CER, on "Devtest" subset

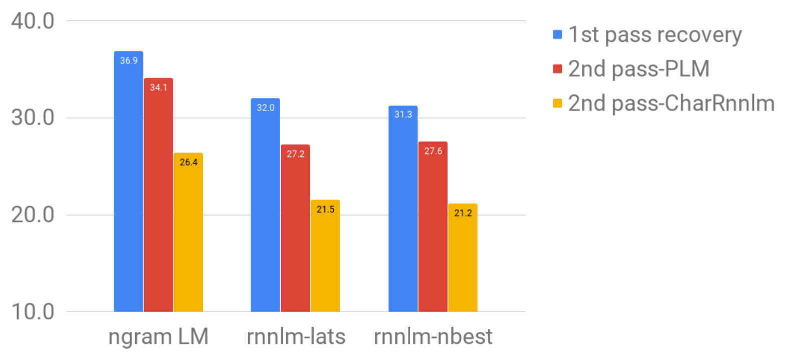


Figure 3.19: Overview of 1st & 2nd pass OOV recovery's impact on OOV CER, on "Native" subset

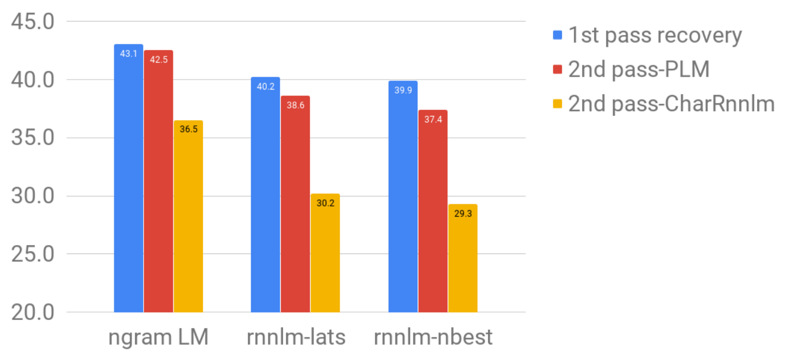


Figure 3.20: Overview of 1st & 2nd pass OOV recovery's impact on OOV CER, on "Nonnative" subset

First, looking at the Figure 3.9 to Figure 3.14, we can see in all cases 1st pass OOV recovery improves both overall WER and CER considerably over the baseline (just decoding & re-scoring with the original small vocab) without surprise, and 2nd pass OOV recovery bring nice improvements further. In detail, we have the following observations (with emphasis on 2nd pass recovery):

- For 2nd pass recovery, a proper strategy of estimating unigram probabilities of OOV candidates seem to be important. It can be seen that CharRnnlm consistently out-performs PLM, regardless of datasets, 1st or 2nd pass decoding, overall or OOV WER or CER.

- Regarding overall WER improvements brought by 2nd pass recovery (with CharRnnlm) over 1st pass recovery, on the easier domains ("Devtest" which contains all read speech, and "Native"), 2nd pass recovery brings around $11 - 13\%$ (N-gram LM) and $13 - 18\%$ (RNNLM) relative improvements. On the harder domain ("Nonnative"), the improvements are 6% (N-gram LM) and 9% (RNNLM). The trend on overall CERs is similar. This basically implies that 2nd pass decoding and open-vocab RNNLM re-scoring both independently help. And the gains on easier domains are consistently larger. The main reason should be, with an easier acoustic condition, the phonemic decoding results are generally better, which is to be confirmed from OOV WER/CER results.

- Regarding OOV WER improvements brought by 2nd pass recovery (with CharRnnlm) over 1st pass recovery, on the easier domains, 2nd pass recovery brings around $18 - 24\%$ (N-gram LM) and $21 - 26\%$ (RNNLM) rel. improvements. On the harder domain, the rel. improvements are 13% (N-gram LM) and 20% (RNNLM). The trend on overall CERs is similar, though the relative gains are larger overall: On easier domains, 2nd pass recovery brings around $27 - 28\%$ (N-gram LM) and $28 - 34\%$ (RNNLM) relative improvements. On the harder domain, the relative improvements are 16% (N-gram LM) and 25% (RNNLM).

This basically implies that 2nd pass grammar decoding and open-vocab RNNLM re-scoring both independently help with both OOV and overall recognition performance. And the gains on easier domains are consistently larger. The main reason should be, with a easier acoustic condition, the phonemic decoding results are generally better, which can be confirmed from the consistent trend in OOV WER/CER results.

To help better understand how does 2nd pass decoding helps, we show some example utterances where we have lower error rates after 2nd pass decoding. The red words in reference texts are OOVs:

- *Reference:* has trabajado en una oficina

  1st pass: hastrabajado en una oficina

  2nd pass: hasta trabajado en una oficina

- *Reference:* te gustaría trabajar en un banco

  1st pass: tegustaría trabajar en un banco

  2nd pass: te gustaría trabajar en un banco

- *Reference:* el tren llegará retrasado

  1st pass: el tren llegar a retrasado

  2nd pass: el tren llegará retrasado

It can be seen that, in the first example, OOV "trabajado" is stuck with its neighbor "has" in the 1st pass decoding hypothesis, and in the 2nd example, two OOVs "te" and "gustaría" are stuck together in the 1st pass decoding hypothesis. In the third example, OOV "llegará" is mis-recognized as two words in the 1st pass decoding hypothesis. These cases all got improved in the 2nd pass decoding hypothesis (though "has" in the first example is still mis-recognized). These examples more intuitively illustrates that 2nd pass decoding with calibrated OOV unigram distribution improves recognition of OOVs and surrounding words.

3.3.3.2.2 Improving 2nd pass OOV recovery performance Earlier we mentioned that the reason why we need a 2nd pass decoding for OOV recovery is to get more freedom to calibrate the OOV cost and OOV distribution, by adopting different estimation schemes of OOV candidates' LM probabilities. Here we explore them in experiments.

Effect of different strategies of OOV unigram probability estimation We explore different options of formula $\mathcal{F}(w)$ for OOV unigram probability estimation in 2nd pass decoding. Results (only showing OOV CERs, which is the most relevant metrics here) are shown in Table 3.21-Table 3.23. "PLM" stands for using phonemic LM scores; "CharRnnlm" means using character RNNLM scores; "Const" means using a uniform distribution; "Empr" means

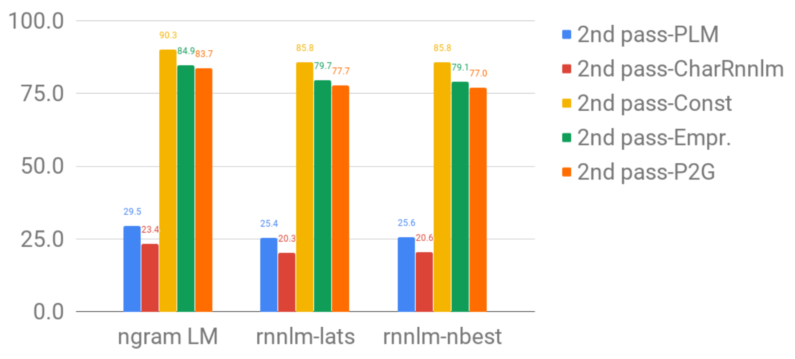


Figure 3.21: Impact of different estimation schemes of OOV unigram probabilities (in 2nd pass OOV recovery) on OOV CER, on "Devtest" subset

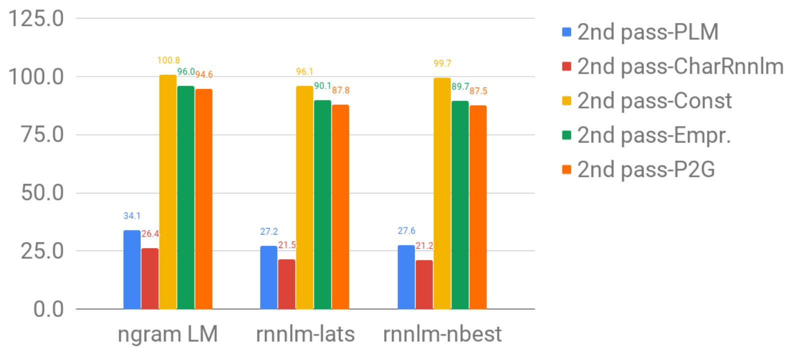


Figure 3.22: Impact of different estimation schemes of OOV unigram probabilities (in 2nd pass OOV recovery) on OOV CER, on "Native" subset

using empirical frequency of OOV candidates based on first pass lattice-statistics. "P2G" means using log-probability of the one-best sequence of grapheme-phonemes when converting OOV candidates from their phonemic form to graphemic form.

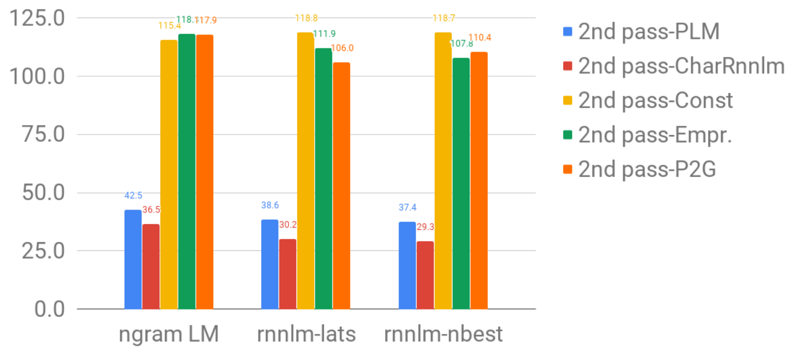


Figure 3.23: Impact of different estimation schemes of OOV unigram probabilities (in 2nd pass OOV recovery) on OOV CER, on "Nonnative" subset

From the above results, we can see that, the "Const", "Empr." and "P2G" options are considerably worse than "PLM" and "CharRnnlm". This agrees with both our expectation and also related literature using character RNNLM to estimate LM probabilities for OOVs in the end-to-end ASR framework [56]. The reason why using uniform probabilities ("Const") is bad is easily understandable (not differentiating good and bad OOV candidates at all). The reason why using empirical frequency ("Empr.") is also bad is harder to understand. Our theory is that, since we only sampled low frequency words as ground-truth OOVs, the empirical frequency is too noisy to be a robust estimator of OOV unigram probabilities. Also, most importantly, for the three options "Const", "Empr." and "P2G", we have observed that neither of them is not able to assign low scores to long and incorrect OOV candidates, resulting in lots of insertion errors (This can be seen from the fact that, on "Nonnative" subset, OOV CER for those three estimators are all above 100%.).

Among the two best options we care the most, the reason why "CharRnnlm" is consistently better than "PLM" is that, since Spanish is a regularly spelled language, the information provided by phone sequences and character sequences are basically the same, so that the only advantage of "CharRnnlm" should be the stronger modeling power of RNNLM over an N-gram LM. Note that in all cases, open-vocab RNNLM re-scoring preserves the advantages of "PLM" and "CharRnnlm" options.

Effect of incorporating empirical frequency into OOV unigram probability estimation

Although using empirical frequency alone cannot robustly estimate OOV unigram probabilities, we find it useful when we couple it with another estimator. In this part, we explore ways of combining the empirical frequency with the CharRnnlm probabilities, again showing results on OOV CERs only (Table 3.24 - Table 3.26). "CharLM-intp-0.1-Empr." means interpolating the CharLM probabilities (with weight 1-0.1=0.9) with the empirical frequency (with weight 0.1) in the log domain. "CharLM-mul.-Empr" means we simply multiply the two terms and then re-normalize all probabilities to form a distribution. From the results it seems interpolation sometimes helps, compared with using CharLM probs alone. However multiplying them is always the best strategy. The possible reason is that simply multiplication results in a sharper distribution with higher variance, and therefore better discrimination between good and bad candidates. In all following experiments, whenever we try this type of composite estimator, we'll adopt the multiplication strategy.

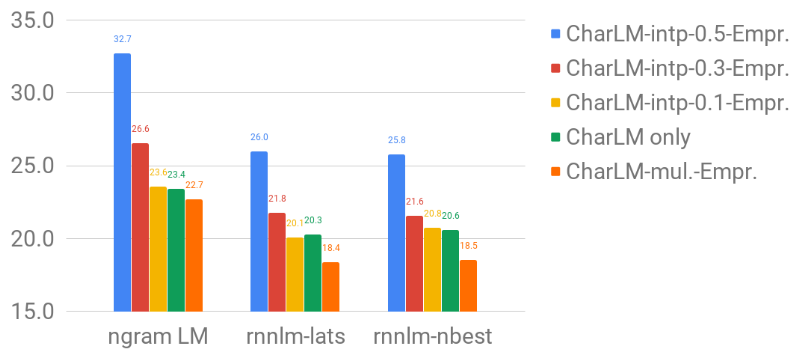


Figure 3.24: Impact of different strategies of incorporating empirical frequency into unigram distributions of OOV candidates (in 2nd pass OOV recovery) on OOV CER, on "Devtest" subset

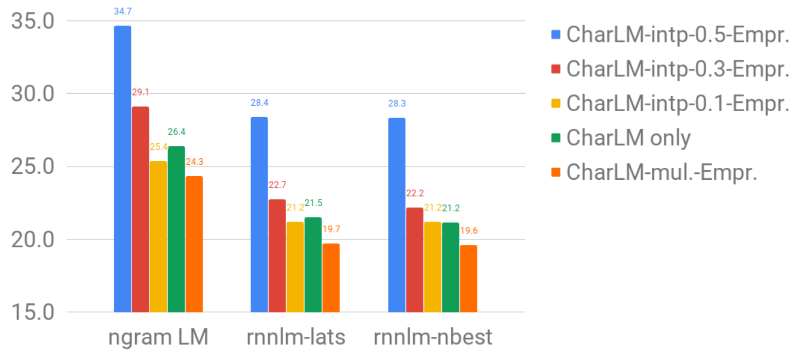


Figure 3.25: Impact of different strategies of incorporating empirical frequency into unigram distributions of OOV candidates (in 2nd pass OOV recovery) on OOV CER, on "Native" subset

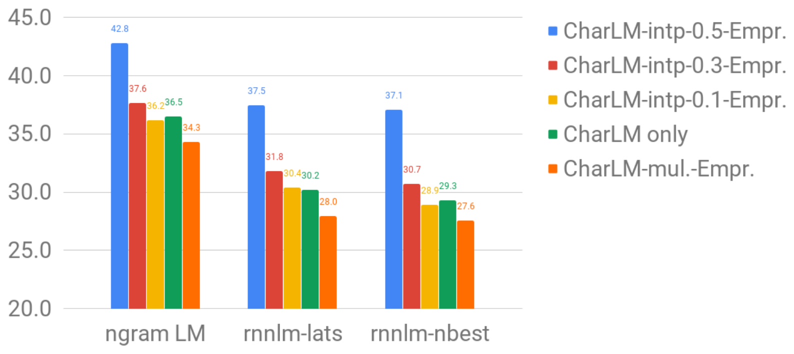


Figure 3.26: Impact of different strategies of incorporating empirical frequency into unigram distributions of OOV candidates (in 2nd pass OOV recovery) on OOV CER, on "Nonnative" subset

Effect of adjusting α, the boosting parameter of OOV grammar in 2nd pass decoding

Next we explore the impact of α (See Eq. (3.8)) which determines the desired probability mass of OOV grammar, on both overall/OOV WER/CERs. We fix the OOV unigram probability estimator at the best one ("CharLM-mul.-Empr"), and vary α from 1 to 5:

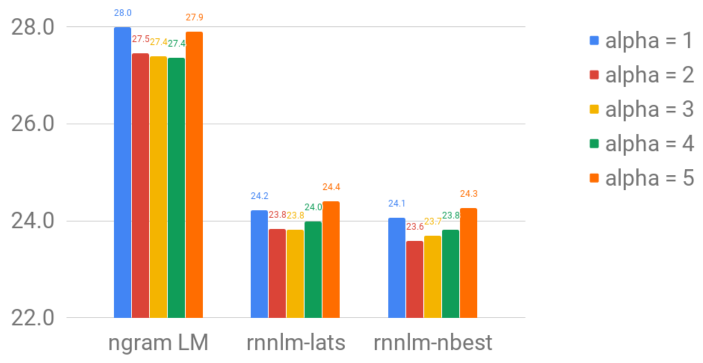


Figure 3.27: Impact of OOV boosting factor (alpha) on overall WER, on "Devtest" subset

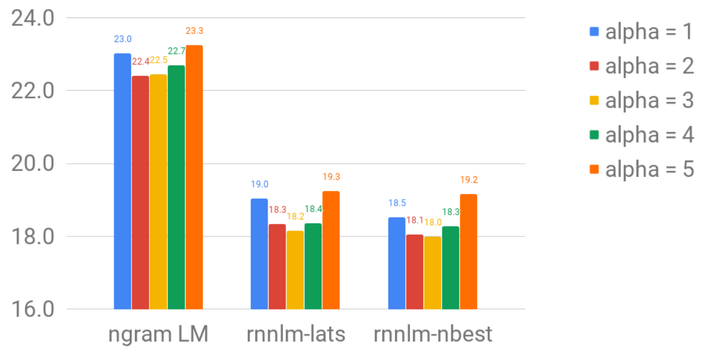


Figure 3.28: Impact of OOV boosting factor (alpha) on overall WER, on "Native" subset

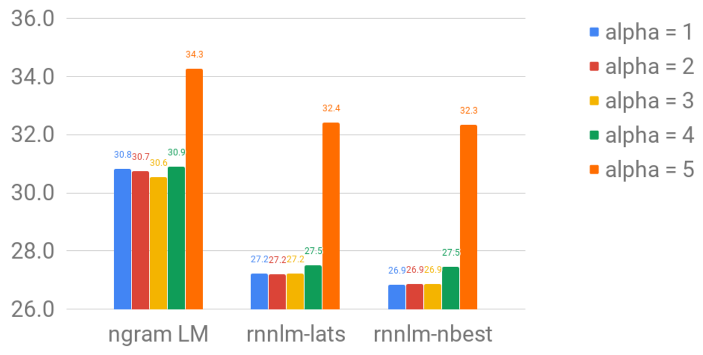


Figure 3.29: Impact of OOV boosting factor (alpha) on overall WER, on "Nonnative" subset

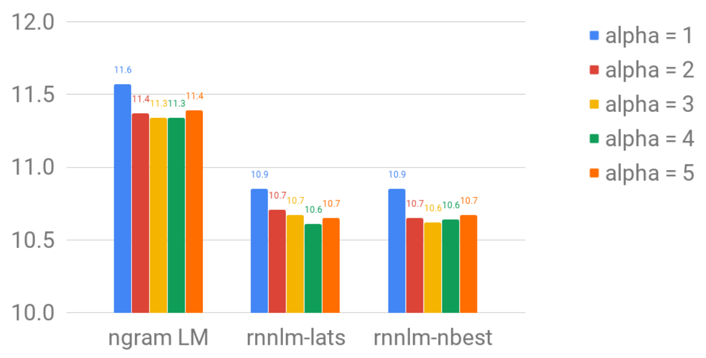


Figure 3.30: Impact of OOV boosting factor (alpha) on overall CER, on "Devtest" subset

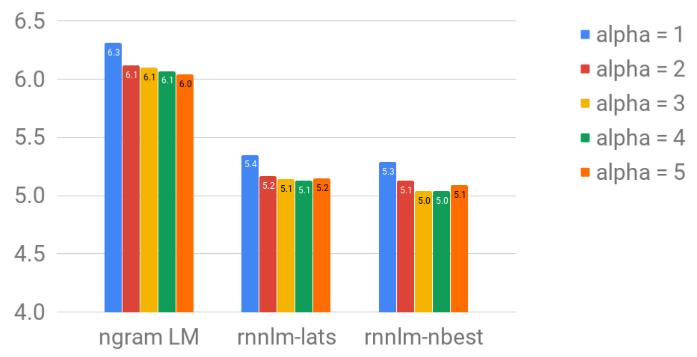


Figure 3.31: Impact of OOV boosting factor (alpha) on overall CER, on "Native" subset

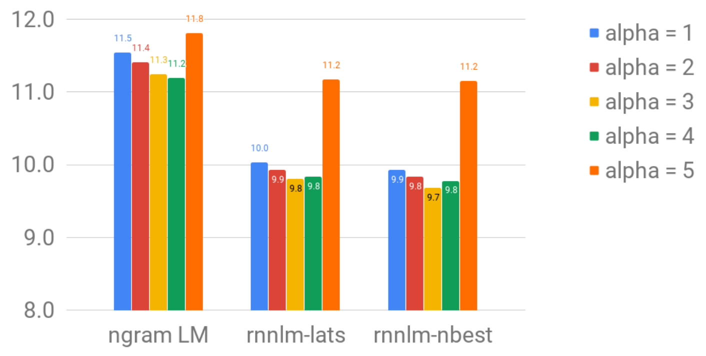


Figure 3.32: Impact of OOV boosting factor (alpha) on overall CER, on "Nonnative" subset

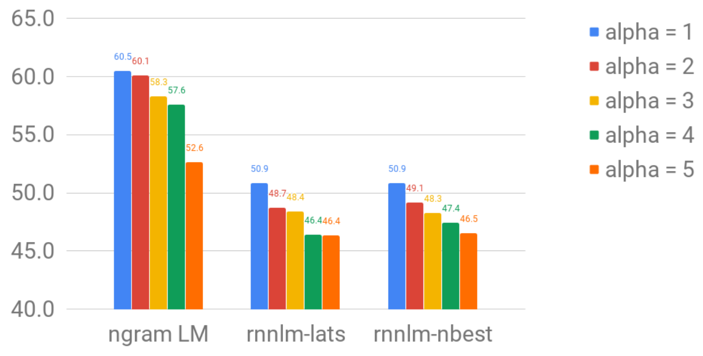


Figure 3.33: Impact of OOV boosting factor (alpha) on OOV WER, on "Devtest" subset

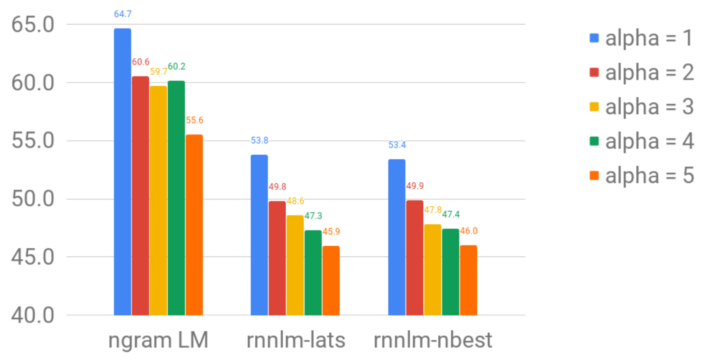


Figure 3.34: Impact of OOV boosting factor (alpha) on OOV WER, on "Native" subset

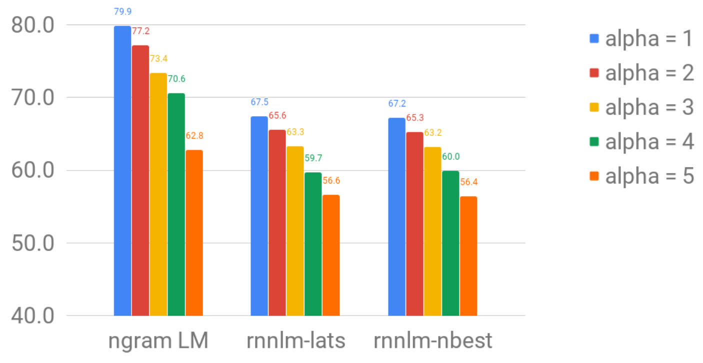


Figure 3.35: Impact of OOV boosting factor (alpha) on OOV WER, on "Nonnative" subset

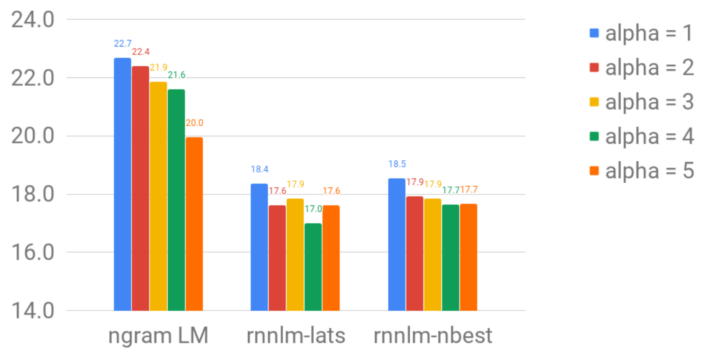


Figure 3.36: Impact of OOV boosting factor (alpha) on OOV CER, on "Devtest" subset

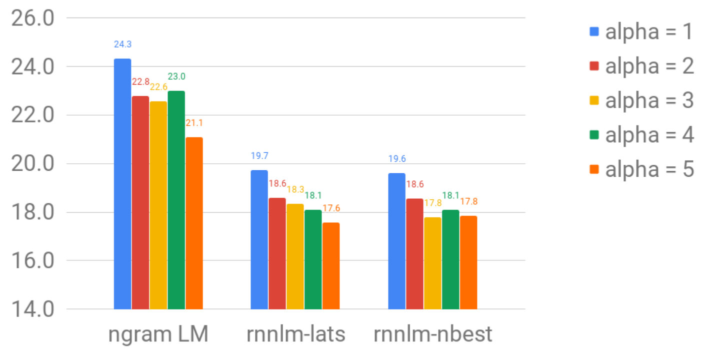


Figure 3.37: Impact of OOV boosting factor (alpha) on OOV CER, on "Native" subset

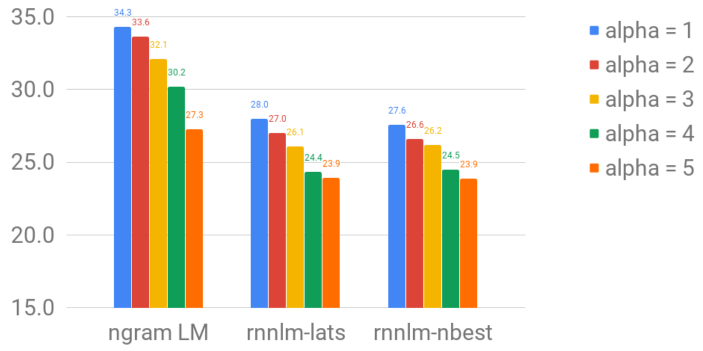


Figure 3.38: Impact of OOV boosting factor (alpha) on OOV CER, on "Nonnative" subset

We can see that, as we increase α, as expected, both OOV WER and OOV CER keep improving, over all test sets. However, it's not the case for overall WER and CER: For overall WER, increasing α from 1 to 3 helps everywhere, but further increasing it causes degradation. Especially, $\alpha = 5$ causes significantly bad overall WER and CER on the harder "Nonnative" test set. We found the reason is that, for "Nonnative" subset, when increasing the OOV prior, substitution errors cause by mis-recognizing an IV word as an OOV word increase much more than "Native" subset. This can be intuitively understood by the fact that, if a nonnative speaker talks to you, because of the accent, you will have more trouble to tell whether a spoken word is OOV or not. This means we need to adjust α very carefully on a validation set with matched acoustic condition in order to balance OOV recovery and overall decoding performance.

3.3.3.3 Experiments on English read speech

From Spanish experiments above, we have verified the effectiveness of 2nd pass decoding and explored various strategies for performance improvement for it. Therefore we'll do 2nd pass decoding by default in following experiments. For English experiments in this section, we'll pick several variables we care the most about to experiment with: using PLM or CharRnnlm to estimate OOV unigram probabilities (incorporating empirical frequency or not); open-vocab RNNLM rescoring. Then we'll compare the OOV recovery pipeline (of the best 2nd pass recovery setup), with the baseline of decoding with limited vocabulary with OOV recovery.

The acoustic model is a 10-layer TDNN-LSTM [28] model trained with LF-MMI [27] on context-dependent phoneme units, trained on all 960 hours of acoustic training data. The language models are a 4-gram model trained on audio books [25], and a 3-layer TDNN-LSTM RNNLM trained on the same data[15]. For the lexicon, we randomly removed 95% infrequent words (with count <= 100 in all training data) words from the 200K reference lexicon, so that the resulted lexicon size is about 10K (11.4K entries). Then we train a G2P model using this lexicon. The number of tokens and OOV rate of four test sets (dev-clean, dev-other, test-clean, test-other) w.r.t the sampled lexicon are as the following:

Table 3.6: Basic statistics of Librispeech test sets, w.r.t the sampled vocabulary.

Test set	#. Tokens (K)	OOV rate (%)	#. Word Types (K)	OOV Type Rate (%)
dev-clean (5.4h)	54.4	9	8.3	45.2
dev-other (5.1h)	51	8.2	7.4	42
test-clean (5.4)	52.6	9	8.1	44.5
test-other (5.3)	52.3	8.5	7.6	43.1

3.3.3.3.1 Effect of different strategies of OOV unigram probability estimation

The following figures are OOV recovery (with 2nd pass decoding) results (Overall/OOV WER/CERs) with four different estimation schemes of OOV unigram probabilities (PLM/CharRnnlm with/without multiplying with empirical frequency). We show results on test-clean and test-other test sets.

[15] AM and LM recipes can be found at:
`https://github.com/kaldi-asr/kaldi/tree/master/egs/librispeech/s5`

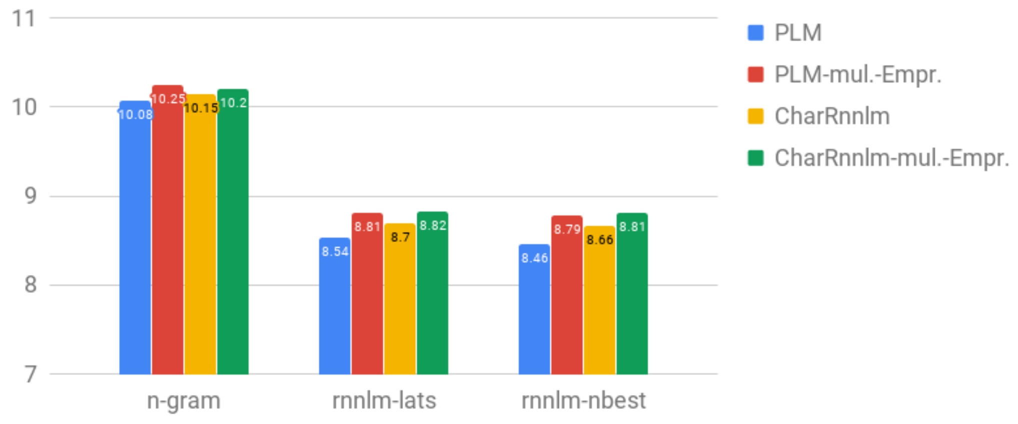


Figure 3.39: Impact of different estimation schemes of OOV unigram probabilities on overall WER, on "Test-clean" subset

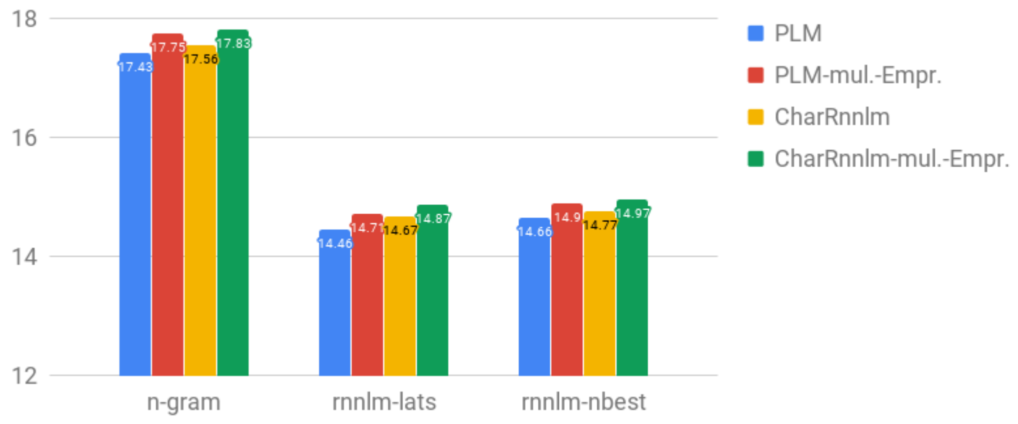


Figure 3.40: Impact of different estimation schemes of OOV unigram probabilities on overall WER, on "Test-other" subset

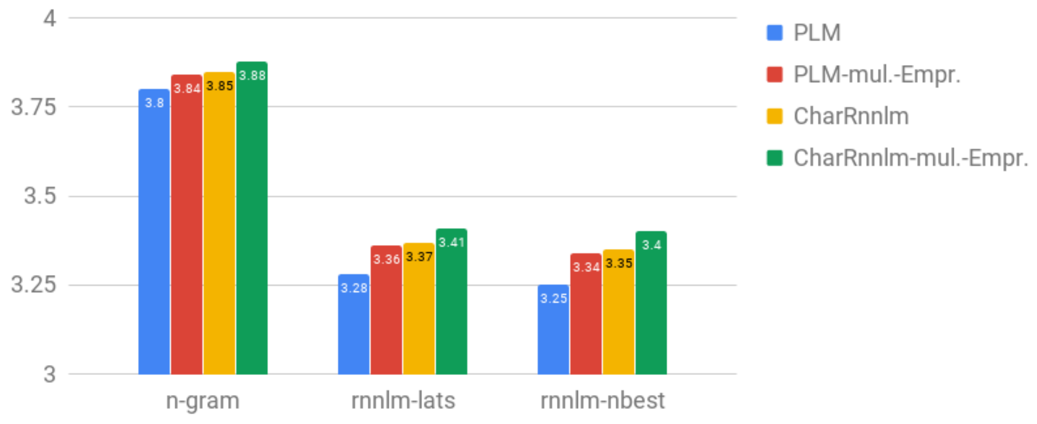


Figure 3.41: Impact of different estimation schemes of OOV unigram probabilities on overall CER, on "Test-clean" subset

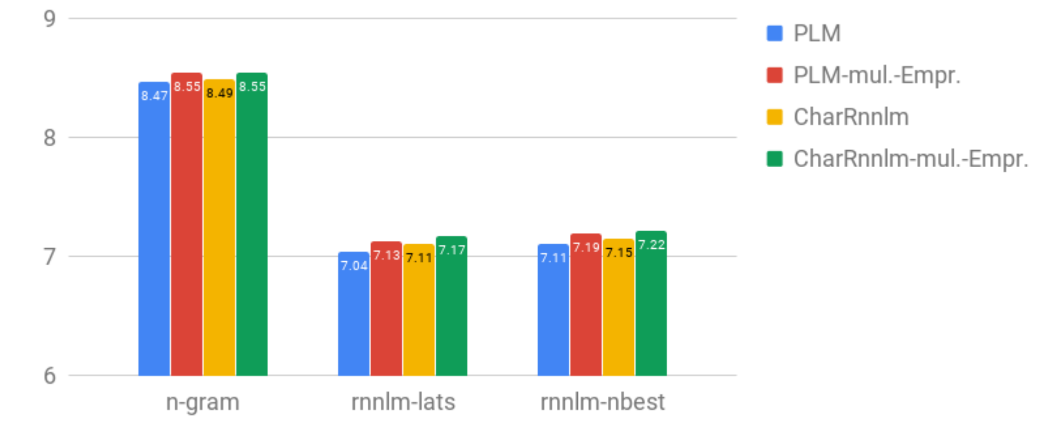


Figure 3.42: Impact of different estimation schemes of OOV unigram probabilities on overall CER, on "Test-other" subset

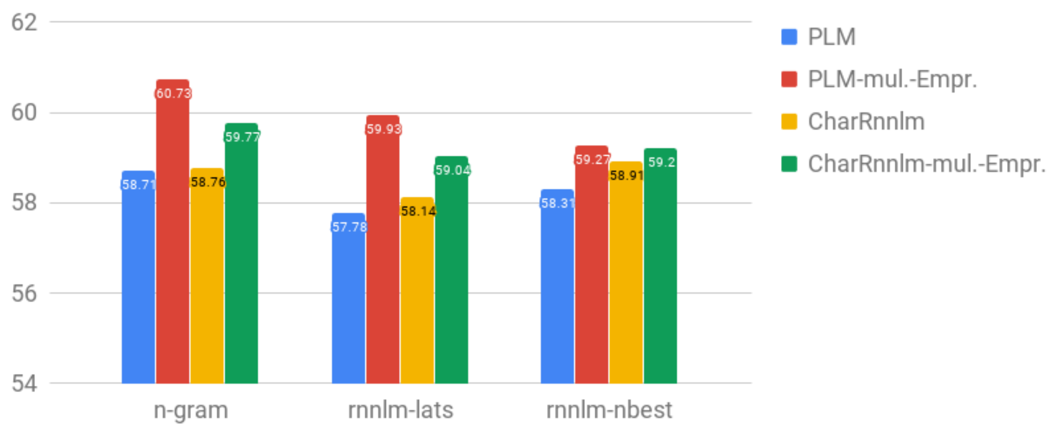


Figure 3.43: Impact of different estimation schemes of OOV unigram probabilities on OOV WER, on "Test-clean" subset

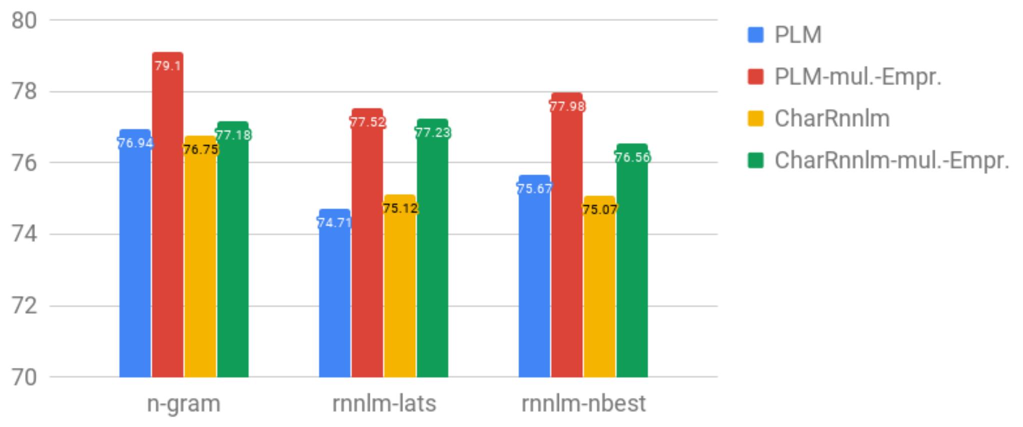


Figure 3.44: Impact of different estimation schemes of OOV unigram probabilities on OOV WER, on "Test-other" subset

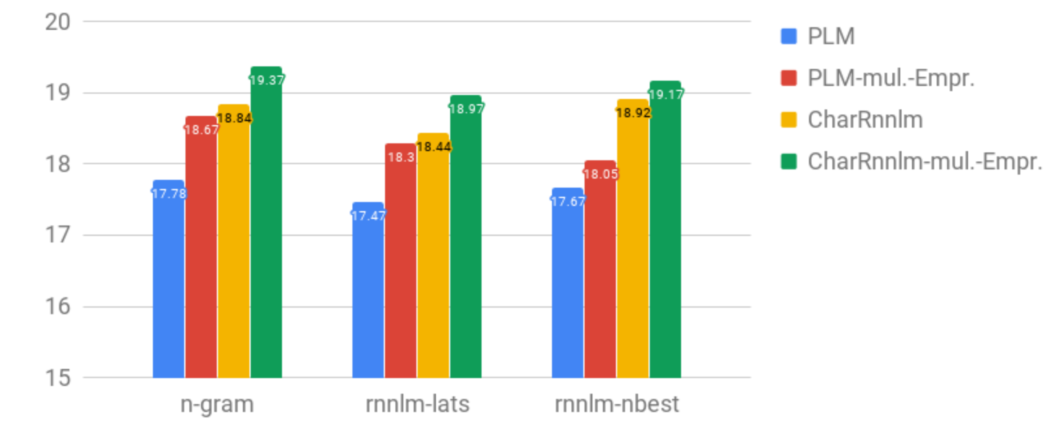


Figure 3.45: Impact of different estimation schemes of OOV unigram probabilities on OOV CER, on "Test-clean" subset

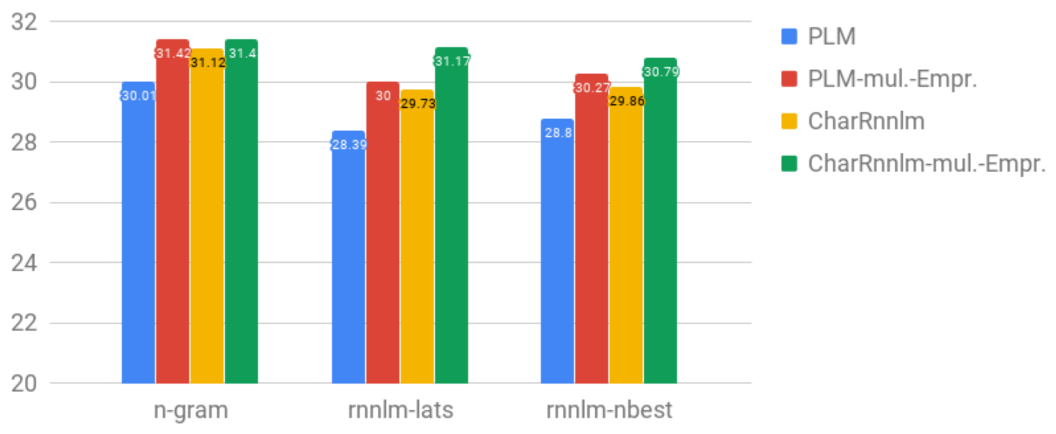


Figure 3.46: Impact of different estimation schemes of OOV unigram probabilities on OOV CER, on "Test-other" subset

From results above, we have the following observations:

- Different from Spanish, the strategy of estimating unigram probabilities of OOV candidates seems not to make a big difference. It can be seen that, over three metrics: overall WER/CER and OOV WER on two test sets, PLM mostly rivals CharRNNLM. On OOV CER, PLM noticeably outperforms CharRnnlm, but only by around relative 5%, which is smaller than the gap 20% in Spanish. The reason should be, as English is a irregularly spelled language, there are generally more errors after recovering OOV spellings with P2G from their pronunciations. Therefore, estimating OOV unigram probabilities with character LM probabilities in the spelling domain has less advantage over using N-gram PLM scores in the pronunciation domain, even with a character RNNLM. This is actually not bad, since in practice using N-gram PLM scores is much more convenient than training a character RNNLM, as the PLM is just obtained from the preparation process of the first pass hybrid decoding.

- It can be seen that over all four metrics and two test sets, incorporating empirical frequency into the OOV unigram probability estimation scheme is a bad idea. Though not much, it consistently degrades results for both PLM and CharRnnlm, not like consistently helping in the Spanish case. This can be understood from the fact that the OOV rates on Librispeech experiments are below 10 (see Table 3.5), much lower than 20% in the Spanish experiments. Therefore the empirical frequency of OOV candidates might be too noisy to be helpful. In practice, we can determine using this strategy or not by looking at empirical OOV rates on a development set.

3.3.3.3.2 Comparing the best OOV recovery setup with baseline Then, we compare the best OOV recovery setup (with PLM, without incorporating empirical frequency) with the baseline of decoding with the limited vocabulary without OOV recovery. We show results on overall WER/CER and OOV CER. OOV WER is not meaningful since the baseline cannot recognize any OOV. Results are shown in Table 3.7 - Table 3.9.

Table 3.7: Overall WERs on four test sets of Librispeech, with the best 2nd pass recovery setup.

dev-clean	Baseline	OOV recovery
n-gram	14.75	9.86
rnnlm-lats	13.6	8.42
rnnlm-nbest	13.48	8.37
dev-other	**Baseline**	**OOV recovery**
n-gram	19.53	17.41
rnnlm-lats	17.12	14.28
rnnlm-nbest	17.23	14.46
test-clean	**Baseline**	**OOV recovery**
n-gram	15.16	10.08
rnnlm-lats	13.97	8.54
rnnlm-nbest	13.92	8.46
test-other	**Baseline**	**OOV recovery**
n-gram	19.9	17.43
rnnlm-lats	17.35	14.46
rnnlm-nbest	17.56	14.66

Table 3.8: Overall CERs on four test sets of Librispeech, with the best 2nd pass recovery setup.

dev-clean	Baseline	OOV recovery
n-gram	7.58	3.89
rnnlm-lats	7.41	3.34
rnnlm-nbest	7.35	3.31
dev-other	**Baseline**	**OOV recovery**
n-gram	11.24	8.51
rnnlm-lats	10.5	7.08
rnnlm-nbest	10.45	7.17
test-clean	**Baseline**	**OOV recovery**
n-gram	7.48	3.8
rnnlm-lats	7.3	3.28
rnnlm-nbest	7.26	3.25
test-other	**Baseline**	**OOV recovery**
n-gram	11.02	8.47
rnnlm-lats	10.13	7.04
rnnlm-nbest	10.13	7.11

Table 3.9: OOV CERs on four test sets of Librispeech, with the best 2nd pass recovery setup.

dev-clean	Baseline	OOV recovery
n-gram	50.88	18.89
rnnlm-lats	51.81	18.8
rnnlm-nbest	52.62	18.63
dev-other	**Baseline**	**OOV recovery**
n-gram	54.93	28.65
rnnlm-lats	54.5	27.78
rnnlm-nbest	54.38	27.66
test-clean	**Baseline**	**OOV recovery**
n-gram	50.35	17.78
rnnlm-lats	49.49	17.47
rnnlm-nbest	49.4	17.67
test-other	**Baseline**	**OOV recovery**
n-gram	53.62	30.01
rnnlm-lats	54.35	28.39
rnnlm-nbest	54.66	28.8

From the above results, we can see that, with N-gram LM, on the easier domains (dev-clean and test-clean), we have around 30% relative improvment in overall WER, around 50% relative improvement in overall CER, and around 60% relative improvement in OOV CER. On the harder domains (dev-other and test-other[16]), we have around 20% relative improvment in overall WER, around 30% relative improvement in overall CER, and around 50% relative improvement in OOV CER, which are generally less than the improvement we have in the easier domains.

In most cases, RNNLM rescoring gives more improvements for the "OOV recovery" column than the "Baseline" column, meaning the relative improvements brought by OOV recovery becomes larger after RNNLM rescoring, on all metrics. This further reaffirms that the proposed open-vocab RNNLM rescoring is capable of helping improve both OOV and overall recognition performance.

3.3.3.3.3 Experiments with the full-vocabulary condition Previously we've shown the effectiveness of the proposed OOV recovery pipeline under the sampled-vocabulary condition (in order to get enough simulated OOVs). Although this simulated "high OOV rate" condition is common in lots of low-resource languages, the "low OOV rate" condition is the most practical scenario for high resource languages like English, which is also harder since the OOVs are mostly name entities. Therefore it's important to make sure our pipeline performs reasonably well in this case. The official Librispeech evaluation condition (200K Librispeech lexicon) is a good test-bed since the

[16]they contain higher WER speakers by construction

vocabulary is large enough so that OOV rates in test sets are relatively low (Table 3.10), while the OOVs are mostly foreign words/name entities (e.g. `RUGGEDO'S`).

Besides, in practice, during test time, we sometimes have access to extra text containing OOVs we want recognize in test audio. The oracle knowledge of spelling and context information (if a particular OOV is frequent enough) of OOVs could potentially help OOV recovery performance. Here we'll explore how much gain we can have by utilizing oracle spelling of OOV words. We'll add one evaluation condition of "+ Oracle spelling", meaning during the 2nd pass (grammar) decoding and RNNLM rescoring, we only add oracle OOV candidates from each test set to lexicon, with their pronunciations given by G2P, and unigram probabilities given by our chosen strategy (which is PLM here). One would wonder here, given oracle spellings, then what's the value of 1st pass hybrid decoding? Our theory is that the first pass hybrid decoding provides "acoustic evidence" of OOV pronunciations from PLM, which may help improve the pronunciation quality of OOVs, and thereby improve recovery performance. To verify this, we collect all phonetic OOV candidates whose spelling recovered by P2G[17] match oracle OOVs' spelling, and use them as extra acoustic-evidence-based pronunciation candidates for oracle OOVs, besides G2P generated candidates (similar as the pronunciation candidates used in lexicon-learning). In our following experiments, we'll investigate how much could these extra pronunciation candidates help with OOV recovery performance.

[17]In order to ensure we cover the oracle spelling of OOVs, we use a large *n* (100) when applying P2G.

The OOV rates w.r.t. the full 200K lexicon on four test sets are summarized in Table 3.10, which are pretty low as expected.

Table 3.10: Basic statistics of Librispeech test sets, w.r.t the official 200K vocabulary.

Test set	#. Tokens (K)	OOV rate (%)	#. Word Types (K)	OOV Type Rate (%)
dev-clean (5.4h)	54.4	0.3	8.3	1.4
dev-other (5.1h)	51	0.6	7.4	1.9
test-clean (5.4h)	52.6	0.4	8.1	1.6
test-other (5.3h)	52.3	0.5	7.6	2.3

The acoustic model is a 17-layer factorized TDNN (TDNN-F) [68] model trained with LF-MMI [27] on context-dependent phoneme units, on all 960 hours of acoustic training data. The language models is the same official 4-gram model trained on audio books [25], and a 5-layer TDNN-LSTM RNNLM trained on the same data[18]. The lexicon is the official 200K lexicon.

Before getting into details on OOV recognition performance, this time we first show the whole picture of overall WERs. In Table 3.11, we present results of our TDNN-F acoustic model (trained with LFMMI) + RNNLM rescoring together with several recent state-of-the art baselines. We listed the acoustic model architecture, training criterion being used in each model. Note that all models uses neural LM (RNNLM or Transformer LM). We categorize the models into two classes: sequence-to-sequence (S2S) and non-S2S. The reason is that regardless of the training procedures, they made a big difference at inference time: Although attention-based sequence-to-sequence models,

[18]recipe can be found at: `kaldi/egs/librispeech/s5/local/chain/tuning/run_tdnn_1d.sh` from Kaldi's GitHub repository

e.g. Listen-Attend-and-Spell (LAS)[69], have achieved state-of-the-art performance in many ASR tasks [70], they have limited potential of online decoding, since usually an entire utterance must be seen by the encoder, before any token can be decoded. Other concerns are the large model size [70] and integration with a language model. From table 3.11 it can be seen that, though the AM size is pretty small (22.6M), our baseline TDNN-F system achieves best results on dev-clean and dev-other among all reported results. On test-clean and test-other, we achieved the best results among non-S2S models, though still lag behind the LAS+SpecAugment[19] model.

Table 3.11: Librispeech WERs with the official vocabulary and neural LMs

Model	AM Params	dev-clean	dev-other	test-clean	test-other
non-S2S					
TDNN-F (LFMMI)	22.6M	2.4	6.97	2.84	7.38
TDNN-F (LFMMI) + **OOV recovery**	22.6M	**2.39**	**6.94**	2.75	7.41
FSMN (LFMMI) [72]	-	2.56	7.47	2.97	7.5
TDNN (CTC) [73]	333M	2.68	8.62	2.95	8.79
S2S					
BLSTM (LAS) [74]	-	3.54	11.52	3.82	12.76
CNN + BLSTM (LAS) [71]	-	-	-	3.2	9.8
CNN + BLSTM (LAS) + SpecAugment [71]	-	-	-	**2.5**	**5.8**

For OOV recovery, it can be seen that it degrades WER by 0.03 on "test-other" and helps a bit on the other three sets. We then present full set of OOV recovery analysis in Table 3.12. The "Baseline" is the same system as "TDNN-F (LFMMI)" in Table 3.11, and RNNLM rescoring is always used in both baseline

[19]SpecAugment[71] stands for a recently proposed data augmentation technique based on time-warping + frequency masking + time masking. We are currently trying that for non-S2S models.

and OOV recovery results[20]. From the results we can see that, on average, OOV recovery helps a tiny on overall WER (by 1%), more noticeable on OOV WER (4%), and much more on OOV CER (%). Particularly, for each metric, the relative improvements on two easier test sets, dev-clean and test-clean (5-8% on OOV WER and 17-20% on OOV CER) are consistently larger than the other two test sets (0-1.5% on OOV WER and 4-6% on OOV CER). Again this agrees with our earlier finding that OOV recovery performance is sensitive to the audio quality. Also we notice that these "real" OOVs here are much harder to recognize (e.g. OOV WERs are all above 90%) that than the simulated OOVs in our earlier limited-vocab experiments, where the OOV WERs were always in the range of 50%-80% (e.g. Table 3.43 and 3.44).

Then, if we use oracle spelling of OOVs, we can see it achieves much more gains over all metrics, enlarging the relative improvement of overall WER from 1% to 7%, OOV WER from 4% to 43%, and OOV CER to 11% to 40%. Furthermore, by adding pronunciations learned from acoustic evidence of 1st pass hybrid decoding to the lexicon, we are able to achieve more gains on all metrics, more notably on OOV WER/CERs. This implies that, even given oracle spelling of OOVs, the proposed HLM framework could provide useful acoustic evidence for OOV pronunciations to improve OOV recovery performance. Note that, to be consistent, for all experiments involving oracle OOV spellings, we've been relying on the same method (phonemic LM scores) as we did before to estimate the unigram LM probabilities of OOVs, rather than using the empirical frequency estimates (which are very low for all

[20]For the OOV recovery pipeline, we have a N-gram rescoring stage following the standard baseline recipe, before RNNLM rescoring. We only present the final results after RNNLM rescoring here for simplicity.

OOVs).

Table 3.12: Librispeech OOV recovery performance (after RNNLM rescoring) with the official 200K-vocab condition (all average relative impr. in red are measured over baseline numbers)

	dev-clean	dev-other	test-clean	test-other	Rel. $\bar{\Delta}$
Overall WER (%)					
Baseline	2.4	6.97	2.84	7.38	
OOV recovery	2.39	6.94	2.75	7.41	1%
+ Oracle spelling	2.15	6.73	2.56	7.18	7%
+ Acoustic evidence	**2.12**	**6.7**	**2.49**	**7.14**	8%
OOV WER(%)					
Baseline	100	100	100	100	
OOV recovery	92.63	100	94.55	98.58	4%
+ Oracle spelling	38.95	68.47	50.91	68.44	43%
+ Acoustic evidence	**30**	**65.76**	**42.27**	**64.18**	50%
OOV CER(%)					
Baseline	47.26	48.32	46.01	53.11	
OOV recovery	39.29	46.19	37.44	50.13	11%
+ Oracle spelling	18.74	35.93	24.08	39.12	40%
+ Acoustic evidence	**15.46**	**35.16**	**19.52**	**37.23**	45%

Here we show some examples utterances of reference and hypothesis (OOV recovery without oracle information) from the "test-other" set, with OOVs in red. We found that many of the OOVs are foreign words or rare name entities. From the examples we can see that the name entities "shahrazad" and "coningsburgh" are not perfectly recognized, but the hypothesis tokens are already acoustically very similar to reference. So the error should be attributed to G2P. And it can be understood that, as we only take the top-1 P2G candidate, the chance of getting the spelling recovered perfectly is small, unless similar examples were seen in G2P training data. Besides, the last example is showing that we still make the "splitting" error on OOVs sometimes.

- *Ref:* shahrazad perceived the dawn of day

  Hyp: shehrazade perceived that the dawn of day

- *Ref:* but the castrato fetched a round

  Hyp: but the castrato fetch the around

- *Ref:* the lord of coningsburgh and he and his followers had scarce departed

  Hyp: the lord of konigsberg and he and his followers had scarce departed

- *Ref:* to display posters announcing platterbaff is out before the poll opens

  Hyp: to display posters announcing platter bath is out before the ball opens

Then we show some example OOVs which were mis-recognized at the "+Oracle spelling" condition, but were correctly recognized at the "+Acoustic evidence" condition:

word	pronunciations
STUTELEY	S T UW1 T L IY0
	S T **AH1** T L IY0
ISLAMISED	IH1 S L AA2 M AH0 S T
	IH1 S L AA2 M AY2 Z D
	IH1 S L AA2 M IH2 Z D
	IH1 **Z** L **AH0** M AY2 Z D
YAUSKY	Y AH1 AH2 S K IY0
	Y AH1 S K IY0
	Y AO1 S K IY0
	Y **OW1** S K IY0

From the above examples, we can see that in most places the acoustic-evidence-based pronunciations differ from G2P-generated pronunciations in vowels. By listening to utterances containing those words, we verified the acoustic-evidence-based pronunciations are closer to the ground-truth in all cases. This illustrates, in some cases 1st pass hybrid decoding can indeed produce pronunciations for OOVs better than G2P.

3.3.3.4 Experiments on English conversational speech

At last, we'll step into a more challenging scenario: OOV recovery for conversational speech. Compared with read speech, because the speaking style is more spontaneous, and the speed also varies more, both word level and phone level recognition will be much harder. We investigate our method on the popular Switchboard conversational English corpus, simulate OOVs by limiting the decoding vocabulary, and repeat the same experiments: use PLM/CharRnnlm to estimate OOV unigram probabilities (either composed with empirical frequency or not); open-vocab RNNLM rescoring, and compare OOV/overall WER/CERs with the baseline of decoding with the the limited vocabulary.

The acoustic model is a 15-layer TDNN-F [68] model trained with LF-MMI [27] on context-dependent phoneme units, trained on all 280 hours of acoustic training data. The language models are a 3-gram model trained on switchboard data, and a 3-layer TDNN-LSTM RNNLM trained on Switchboard+Fisher data [21]. For the lexicon, we randomly removed 78% infrequent

[21] AM and LM recipes can be found at:
`https://github.com/kaldi-asr/kaldi/tree/master/egs/swbd/s5c`

words (with count <= 10 in acoustic training data) words from the 31K reference lexicon, so that the resulted lexicon size is about 7.4K. Then we train a G2P model using this lexicon. The OOV statistics of two test sets (Eval2000, RT03) w.r.t the sampled lexicon, and all experimental results on overall/OOV WER/CERs are as the following:

Table 3.13: Basic statistics of Switchboard test sets, w.r.t the sampled vocabulary.

Test set	#. Tokens (K)	OOV rate (%)	#. Word Types	OOV Type Rate (%)
Eval2000 (3.8h)	42.7	3.3	3.4	25.5
RT03 (6.3h)	75.5	2.9	4.5	28.9

Table 3.14: Overall WERs on Eval2000 and RT03 of Switchboard, with different estimation schemes of OOV unigram probabilities

Eval2000	Baseline	OOV recovery			
		PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	16.2	15.5	15.6	15.5	15.6
rnnlm-lats	14.0	13.3	13.4	13.3	13.4
rnnlm-nbest	14.1	13.3	13.5	13.5	13.6
RT03	**Baseline**	**OOV recovery**			
		PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	18.8	18.2	18.2	18.2	18.3
rnnlm-lats	16.2	15.6	15.7	15.7	15.7
rnnlm-nbest	16.3	15.7	15.8	15.7	15.8

From the results (Table 3.14-Table 3.17), we can see that, similar as Librispeech results, strategies of estimating unigram probabilities of OOV candidates make little difference, with PLM/CharRnnlm without incorporating empirical frequency being slightly better than the other two options over all metrics/test sets. It makes sense since the scenario here (irregular G2P and low OOV rate) is the same as Librispeech.

Table 3.15: Overall CERs on Eval2000 and RT03 of Switchboard, with different estimation schemes of OOV unigram probabilities

Eval2000	Baseline	OOV recovery			
		PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	11.2	10.6	10.7	10.7	10.7
rnnlm-lats	10.1	9.5	9.6	9.6	9.6
rnnlm-nbest	10.2	9.6	9.6	9.7	9.7
RT03	**Baseline**	**OOV recovery**			
		PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	13.5	13.1	13.1	13.1	13.1
rnnlm-lats	12.2	11.7	11.7	11.7	11.8
rnnlm-nbest	12.2	11.8	11.8	11.8	11.8

Table 3.16: OOV WERs on Eval2000 and RT03 of Switchboard, with different estimation schemes of OOV unigram probabilities

Eval2000	OOV recovery			
	PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	84.76	85.87	84.42	85.02
rnnlm-lats	80.65	82.88	80.48	81.68
rnnlm-nbest	80.22	82.11	80.31	83.05
RT03	**OOV recovery**			
	PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	90.46	92.1	89.97	91.19
rnnlm-lats	87.54	90.03	86.93	88.69
rnnlm-nbest	87.23	89.73	86.5	88.75

For OOV recognition performance (Table 3.16-Table 3.17), it can be seen that, the absolute values of OOV WER (ranging around $80-90\%$) and CER (ranging around $40-60\%$) are both much worse than the read speech case (OOV WER ranging around $50-75\%$ and OOV CER ranging around $10-30\%$), meaning recognizing OOVs in conversational speech is much harder than in read speech, because of the speaking style. On OOV CERs, the relative improvement over the baseline is around 20%, much less than 50% in the harder domain of Librispeech experiments. One good finding is that the

Table 3.17: OOV CERs on Eval2000 and RT03 of Switchboard, with different estimation schemes of OOV unigram probabilities

Eval2000	Baseline	OOV recovery			
		PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	56.1	46.29	47.8	46.96	48.01
rnnlm-lats	55.5	44.11	45.43	44.09	44.09
rnnlm-nbest	55.5	43.5	45.51	43.76	45.84

RT03	Baseline	OOV recovery			
		PLM	PLM mul. Empr.	CharRnnlm	CharRnnlm mul. Empr.
n-gram	55.4	46.58	48.01	46.28	47.23
rnnlm-lats	55.1	43.67	45.59	43.87	45.07
rnnlm-nbest	55.2	43.87	45.74	43.86	45.42

relative improvements of OOV recovery again becomes larger after RNNLM rescoring, meaning the proposed open-vocab RNNLM rescoring helps recognize OOVs even in this hard case.

For overall recognition performance (Table 3.14-Table 3.15), compared with the baseline, it can be seen that, as expected, the gain from OOV recovery on overall WER/CER are much less than Librispeech experiments. With either N-gram LM or RNNLM, improvements over the baseline on overall WER/CER are just around 10%, much less than 30% in the harder domain of Librispeech. This is not surprising since the gain in OOV recovery, which directly affects overall recognition performance, also becomes much less. Overall, this shows that, harder acoustic conditions result in poorer performance of the proposed OOV recovery pipeline, reflected in OOV/overall WER/CERs, though we still see consistent improvements over the baseline.

3.4 Summary

In this chapter, firstly, we revisited and introduced our implementation of a hybrid lexical model (HLM) approach for OOV detection, which is efficiently integrated into a standard ASR decoding pipeline. We experimentally showed its advantage over a state-of-the-art hybrid LM + classification based approach, and a single-phone OOV modeling approach, in both OOV detection and recovery performance on a LVCSR task. Secondly, we introduced our grammar decoding framework which enables us to efficiently do second pass decoding to improve OOV recovery performance, avoiding re-compiling the whole decoding graph. We also compared different schemes of estimating OOV unigram probabilities in second pass decoding. Our finding is that phonemic/character language model scores give the best performance, and incorporating empirical frequency into the estimation scheme helps when the OOV rate is high. Thirdly, we introduced a novel open-vocabulary word RNNLM re-scoring framework, enabling us to re-score lattices containing recovered OOVs with a word RNNLM trained without these OOVs. Besides, we also investigated the impact of languages and acoustic condition/speaking styles to the performance of the proposed framework, with the conclusion that the framework performs better with higher phonemicity of the language, less noisy acoustic condition and read (rather than conversational) style speech. Last, we've shown that even if we are given oracle spelling of OOVs, the proposed framework can still achieve performance gain by utilizing acoustic-evidence-based pronunciations obtained from 1st pass hybrid decoding.

Above all, various ASR experiments (with OOV/overall WER/CER as metrics) on Spanish and read/conversational English ASR tasks have shown that the proposed HLM + grammar decoding + open-vocabulary word RNNLM rescoring frameworks is capable of efficiently recovering OOVs in ASR decoding given a closed vocabulary, and achieves further performance gain even if oracle OOV spellings are given.

Chapter 4

Conclusion and Future Direction

4.1 Conclusion

In this dissertation, we explored various strategies for dealing with OOV words in ASR. We have been focused on two lines of research, dealing with OOV words in ASR training and in test data. We first investigated dealing with OOV words in ASR training data, by introducing an acoustic-data driven lexicon learning framework using a likelihood-based criterion for selecting pronunciation candidates from multiple sources, i.e. G2P and phonetic decoding, in a greedy fashion. The conclusion drawn from experiments on various ASR lexicon expansion tasks is that, with the proposed framework, starting with a small expert lexicon, we are able to learn a lexicon for OOV words which performs closer to a full expert lexicon in terms of WER performance on test data, than lexicons built using G2P alone, using a pruning criterion based on pronunciation probabilities or BIC. This work has also been extended to learning pronunciations for IV words, i.e. to improving the hand-crafted lexicon itself. Experiments on lexicon adaptation task shows that for adapting the pronunciations of IV words in a reference lexicon with acoustic evidence helps improve both recognition performance on individual words just by re-decoding with the adapted lexicon, and overall AM performance by re-training the AM with the adapted lexicon.

Secondly, we investigated dealing with OOV words in ASR test data, i.e. OOV detection and recovery. We first re-visited hybrid lexical model (HLM) approach for OOV detection, and experimentally has shown out implementation of HLM approach outperforms a state-of-the-art hybrid LM + classification based approach, and a single-phone OOV modeling approach, in both OOV detection and recovery performance on Switchboard LVCSR task. Next we introduced the grammar-decoding framework for efficient second pass decoding to improve OOV recovery performance. By comparing different schemes of estimating OOV unigram probabilities in second pass decoding, we concluded that phonemic/character language model scores give the best performance. Then we introduced a novel open-vocabulary word RNNLM re-scoring framework, to address the problem of re-scoring lattices containing recovered OOVs, with a word RNNLM that was ignorant of OOVs when it was trained. The conclusion drawn from various ASR experiments on Spanish and read/conversational English ASR tasks is that, the proposed HLM + grammar decoding + open-vocabulary word RNNLM rescoring framework both improve OOV recovery and overall decoding performance over baselines significantly, and the amount of improvements depend on phonemicity of the language, audio quality and speaking styles. Above all, we have shown the potential of an open-vocabulary word-level ASR system which efficiently recovers OOV words while also improves the overall decoding performance, and effectively utilizes oracle spellings (if given) of OOVs to achieve better performance. Additional computational overhead is considerably minimized by avoiding re-compiling decoding graph or re-training RNNLM after vocabulary expansion in test time, making it a practical framework for word-based ASR pipeline with state-of-the-art performance.

4.2 Future directions

Recently there's a trend of adopting graphemic lexicons for ASR, eliminating the need of a human-made lexicon. It has been shown that for lots of non-logographic languages, the graphemic approach is quite successful, rivaling or even outperforming phonemic approaches in many cases with enough training data. We think the underlying reason is that, though the representation power of graphemes is much weaker than phonemes for most languages, context-dependency can make up for it by a large margin. And the lexicon learning method proposed in this dissertation would still be quite applicable: instead of aiming at learning a pronunciation "`eh f d iy ey`" for word **FDA**, we aim at learning something like "`E F D I E Y`". In other words, we just switch the "basis" of representation for pronunciation learning from phonemes to graphemes, where the pronunciation candidates are just naive graphemic pronunciations and graphemic-decoding candidates. We suspect this has the same potential as the current phonemic approach for lexicon learning, in terms of improving acoustic modeling performance, while still totally eliminating the need of an expert lexicon and a G2P model. Similarly, for OOV recovery, more interesting work can be done if we switch to graphemic systems. For example, we'll be using a character LM rather than a phonemic LM to model OOV pronunciations. While the hybrid decoding framework will be the same, we don't need the "P2G" step of Section 3.2.5 in OOV recovery, and therefore a potential source of errors could be avoided. Furthermore, along the open-vocabulary RNNLM rescoring direction, another future avenue is to better understand how an RNNLM models word contexts, and what kind of features (now we only use letter N-gram features) can be used to better model OOV's contexts. For example, if we have sentences like "I visited New York" in RNNLM training text, and we have a sentence "I visited New Orleans" (suppose "Orleans" is an OOV) in test data. How could we design informative and context-aware features, so that the RNNLM will guarantee to rank hypothesis like "I visited New Orleans" above hypothesis like "I visited new or leans"? This should be an interesting and challenging problem.

Bibliography

[1] A. Rousseau, P. Deléglise, and Y. Estève, “Ted-lium: an automatic speech recognition dedicated corpus.” in *LREC*, 2012, pp. 125–129.

[2] H.-K. Kuo, E. E. Kislal, L. Mangu, H. Soltau, and T. Beran, “Out-of-vocabulary word detection in a speech-to-speech translation system,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7108–7112.

[3] T. N. Sainath, R. Prabhavalkar, S. Kumar, S. Lee, A. Kannan, D. Rybach, V. Schogol, P. Nguyen, B. Li, Y. Wu *et al.*, “No need for a lexicon? evaluating the value of the pronunciation lexica in end-to-end models,” *arXiv preprint arXiv:1712.01864*, 2017.

[4] T. Likhomanenko, G. Synnaeve, and R. Collobert, “Who needs words? lexicon-free speech recognition,” 2019.

[5] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, “Flat-start single-stage discriminatively trained hmm-based models for asr,” *IEEE/ACM TASLP*, 2018.

[6] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention - w/o data augmentation," in *Proceedings of INTERSPEECH*, 2019.

[7] X. Li, A. Gunawardana, and A. Acero, "Adapting grapheme-to-phoneme conversion for name recognition," in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007, pp. 130–135.

[8] F. Beaufays, A. Sankar, S. Williams, and M. Weintraub, "Learning name pronunciations in automatic speech recognition systems," in *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*. IEEE, 2003, pp. 233–240.

[9] M. J. Gales, K. M. Knill, and A. Ragni, "Unicode-based graphemic systems for limited resource languages," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5186–5190.

[10] D. F. Harwath and J. R. Glass, "Speech recognition without a lexicon-bridging the gap between graphemic and phonetic systems." in *INTERSPEECH*, 2014, pp. 2655–2659.

[11] C.-y. Lee, T. J. O'Donnell, and J. Glass, "Unsupervised lexicon discovery from acoustic input," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 389–403, 2015.

[12] C.-y. Lee, Y. Zhang, and J. R. Glass, "Joint learning of phonetic units and word pronunciations for asr." in *EMNLp*, 2013, pp. 182–192.

[13] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.

[14] L. Lu, A. Ghoshal, and S. Renals, "Acoustic data-driven pronunciation lexicon for large vocabulary speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 374–379.

[15] N. Goel, S. Thomas, M. Agarwal, P. Akyazi, L. Burget, K. Feng, A. Ghoshal, O. Glembek, M. Karafiát, D. Povey *et al.*, "Approaches to automatic lexicon learning with limited training examples," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5094–5097.

[16] I. McGraw, I. Badr, and J. R. Glass, "Learning lexicons from speech using a pronunciation mixture model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 357–366, 2013.

[17] R. Rasipuram *et al.*, "Combining acoustic data driven g2p and letter-to-sound rules for under resource lexicon generation," in *Proceedings of INTERSPEECH*, no. EPFL-CONF-192596, 2012.

[18] G. Chen, D. Povey, and S. Khudanpur, "Acoustic data-driven pronunciation lexicon generation for logographic languages," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5350–5354.

[19] S. Tsujioka, S. Sakti, K. Yoshino, G. Neubig, and S. Nakamura, "Unsupervised joint estimation of grapheme-to-phoneme conversion systems and

acoustic model adaptation for non-native speech recognition," *Interspeech 2016*, pp. 3091–3095, 2016.

[20] A. Laurent, S. Meignier, T. Merlin, P. Deléglise, and F. Spécinov-Trélazé, "Acoustics-based phonetic transcription method for proper nouns." in *INTERSPEECH*, 2010, pp. 2286–2289.

[21] T. Hain, "Implicit pronunciation modelling in asr," in *ISCA Tutorial and Research Workshop (ITRW) on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology*, 2002.

[22] G. Chen, H. Xu, M. Wu, D. Povey, and S. Khudanpur, "Pronunciation and silence probability modeling for asr." in *INTERSPEECH*, 2015, pp. 533–537.

[23] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian *et al.*, "Generating exact lattices in the wfst framework," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4213–4216.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," *Proc. ASRU*, 2011.

[25] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5206–5210.

[26] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts." in *INTERSPEECH*, 2015, pp. 3214–3218.

[27] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," *INTERSPEECH*, 2016.

[28] V. Peddinti, Y. Wang, D. Povey, and S. Khudanpur, "Low latency acoustic modeling using temporal convolution and lstms," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 373–377, 2018.

[29] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Estève, "Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation," in *International Conference on Speech and Computer*. Springer, 2018, pp. 198–208.

[30] C. White, G. Zweig, L. Burget, P. Schwarz, and H. Hermansky, "Confidence estimation, oov detection and language id using phone-to-word transduction and phone-level alignments," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 4085–4088.

[31] H. Lin, J. Bilmes, D. Vergyri, and K. Kirchhoff, "Oov detection by joint word/phone lattice alignment," in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007, pp. 478–483.

[32] S. Kombrink, L. Burget, P. Matějka, M. Karafiát, and H. Hermansky, "Posterior-based out of vocabulary word detection in telephone speech,"

in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[33] D. Klakow, G. Rose, and X. Aubert, “Oov-detection in large vocabulary system using automatically defined word-fragments as fillers,” in *Sixth European Conference on Speech Communication and Technology*, 1999.

[34] A. Yazgan and M. Saraclar, “Hybrid language models for out of vocabulary word detection in large vocabulary conversational speech recognition,” in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP’04). IEEE International Conference on*, vol. 1. IEEE, 2004, pp. I–745.

[35] A. Rastrow, A. Sethy, and B. Ramabhadran, “A new method for oov detection using hybrid word/fragment system,” in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 3953–3956.

[36] A. Rastrow, A. Sethy, B. Ramabhadran, and F. Jelinek, “Towards using hybrid word and fragment units for vocabulary independent lvcsr systems.” in *Interspeech*, 2009, pp. 1931–1934.

[37] C. Parada, M. Dredze, D. Filimonov, and F. Jelinek, “Contextual information improves oov detection in speech,” in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 216–224.

[38] M. Bisani and H. Ney, "Open vocabulary speech recognition with flat hybrid models," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[39] L. Qin and A. I. Rudnicky, "Finding recurrent out-of-vocabulary words." in *INTERSPEECH*, 2013, pp. 2242–2246.

[40] S. Kombrink, M. Hannemann, and L. Burget, "Out-of-vocabulary word detection and beyond," in *Detection and Identification of Rare Audiovisual Cues*. Springer, 2012, pp. 57–65.

[41] T. Asami, R. Masumura, Y. Aono, and K. Shinoda, "Recurrent out-of-vocabulary word detection using distribution of features." in *INTERSPEECH*, 2016, pp. 1320–1324.

[42] I. Bazzi and J. Glass, "Modeling out-of-vocabulary words for robust speech recognition," in *The Proceedings of the 6 (th) International Conference on Spoken Language Processing*, vol. 1, 2000.

[43] I. Szöke, "Hybrid word-subword spoken term detection," *BRNO University of Technology Department of Computer Graphics and Multimedia*, 2010.

[44] E. Egorova and L. Burget, "Out-of-vocabulary word recovery using fst-based subword unit clustering in a hybrid asr system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5919–5923.

[45] M. A. B. Shaik, A. E.-D. Mousa, S. Hahn, R. Schlüter, and H. Ney, "Improved strategies for a zero oov rate lvcsr system," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5048–5052.

[46] S. Kombrink, M. Hannemann, L. Burget, and H. Heřmanský, "Recovery of rare words in lecture speech," in *International Conference on Text, Speech and Dialogue*. Springer, 2010, pp. 330–337.

[47] L. Qin and A. Rudnicky, "Oov word detection using hybrid models with mixed types of fragments," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[48] ——, "Learning better lexical properties for recurrent oov words," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 19–24.

[49] M. A. Basha Shaik, D. Rybach, S. Hahn, R. Schlüter, and H. Ney, "Hierarchical hybrid language models for open vocabulary continuous speech recognition using wfst," in *SAPA-SCALE Conference*, 2012.

[50] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[51] M. Hannemann, S. Kombrink, M. Karafiát, and L. Burget, "Similarity scoring for recognizing repeated out-of-vocabulary words," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[52] I. Illina and D. Fohr, "Out-of-vocabulary word probability estimation using rnn language model," in *8th Language & Technology Conference*, 2017.

[53] A. Currey, I. Illina, and D. Fohr, "Dynamic adjustment of language models for automatic speech recognition using word similarity," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 426–432.

[54] L. Orosanu and D. Jouvet, "Adding new words into a language model using parameters of known words with similar behavior," *Procedia Computer Science*, vol. 128, pp. 18–24, 2018.

[55] K. Li, H. Xu, Y. Wang, D. Povey, and S. Khudanpur, "Recurrent neural network language model adaptation for conversational speech recognition," *INTERSPEECH, Hyderabad*, pp. 1–5, 2018.

[56] T. Hori, S. Watanabe, and J. R. Hershey, "Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 287–293.

[57] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Springer Handbook of Speech Processing*. Springer, 2008, pp. 559–584.

[58] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays *et al.*, "Personalized speech recognition on mobile devices," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5955–5959.

[59] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," 2009.

[60] P. R. Dixon, C. Hori, and H. Kashioka, "A specialized wfst approach for class models and dynamic vocabulary," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[61] J. Schalkwyk, L. Hetherington, and E. Story, "Speech recognition with dynamic grammars using finite-state transducers," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[62] P. Aleksic, C. Allauzen, D. Elson, A. Kracun, D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5172–5175.

[63] K. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," 2015.

[64] G. Chen, "Phd dissertation: Low resource keyword spotting," 2016.

[65] X. Chen, X. Liu, Y. Qian, M. J. Gales, and P. C. Woodland, "Cued-rnnlmâĂŤan open-source toolkit for efficient training and evaluation of recurrent neural network language models," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 6000–6004.

[66] H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur, "Neural network language modeling with letter-based features and importance sampling," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 6109–6113.

[67] O. Press and L. Wolf, "Using the output embedding to improve language models," *arXiv preprint arXiv:1608.05859*, 2016.

[68] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohamadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Proceedings of INTERSPEECH*, 2018.

[69] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[70] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[71] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[72] X. Yang, J. Li, and X. Zhou, "A novel pyramidal-fsmn architecture with lattice-free mmi for speech recognition," *arXiv preprint arXiv:1810.11352*, 2018.

[73] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *arXiv preprint arXiv:1904.03288*, 2019.

[74] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," *arXiv preprint arXiv:1805.03294*, 2018.

Vita

Xiaohui Zhang received the B.E. degree in Electronic Engineering with a Minor in Applied Physics from the Beijing University of Technology, Beijing, China in 2012. He enrolled in the Ph.D. program in the Department of Electrical and Computer Engineering at the Johns Hopkins University in August 2012, and has been a member of the Center for Language and Speech Processing since then. He also obtained an M.S.E degree in Applied Mathematics and Statistics at the Johns Hopkins University in 2015. His research interests include acoustic modeling, pronunciation learning, OOV recovery for automatic speech recognition, and optimization for large scale machine learning. He worked as a research intern at Mobvoi. Inc during the summer of 2014, and in the Speech Group at Google Inc. during the summer of 2015. He received ECE Graduate Fellowship in 2012 and Alexa fellowship in 2017, both at JHU. Now he is working at Facebook. Inc as a research scientist on video ASR.