Language model domain adaptation for automatic speech recognition systems

Khassanov, Yerbolat

2020

Khassanov, Y. (2020). Language model domain adaptation for automatic speech recognition systems. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/141323>

<https://doi.org/10.32657/10356/141323>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 17:51:41 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

LANGUAGE MODEL DOMAIN ADAPTATION
FOR AUTOMATIC SPEECH RECOGNITION
SYSTEMS

YERBOLAT KHASSANOV

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

2020

LANGUAGE MODEL DOMAIN ADAPTATION FOR AUTOMATIC SPEECH RECOGNITION SYSTEMS

YERBOLAT KHAASSANOV

School of Computer Science & Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

6 Dec. 2019

.....

Date

Xaif

.....

Yerbolat Khassanov

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

6 Dec. 2019

.....

Date

Handwritten signature of Chng Eng Siong in blue ink.

.....

Assoc. Prof. Chng Eng Siong

Authorship Attribution Statement

This thesis contains material from 5 papers published in the following peer-reviewed conferences in which I am listed as an author.

Chapter 3 is published as [Y. Khassanov, T.Y. Chong, B. Bigot, and E.S. Chng. Unsupervised Language Model Adaptation by Data Selection for Speech Recognition. In Asian Conference on Intelligent Information and Database Systems 2017, pp. 508-517. Springer, Cham. DOI: 10.1007/978-3-319-54472-4_48](#)

The contributions of the co-authors are as follows:

- Prof. Chng provided the initial project direction and assisted to edit the final manuscript draft.
- I proposed a potential solution, prepared the experimental setup and conducted all experiments. I also prepared the initial manuscript draft.
- Dr. Bigot assisted to set up the experiments and to edit the final manuscript draft.
- Dr. Chong assisted to analyze the obtained results and to edit the final version of the manuscript.

Chapter 4 is published as

- (1) [Y. Khassanov and E.S. Chng. Unsupervised and Efficient Vocabulary Expansion for Recurrent Neural Network Language Models in ASR. In Proceedings of Interspeech 2018, pp. 3343-3347. ISCA. DOI: 10.21437/Interspeech.2018-1021](#)
- (2) [Y. Khassanov, Z. Zeng, V.T. Pham, H. Xu, and E.S. Chng. Enriching Rare Word Representations in Neural Language Models by Embedding Matrix Augmentation. In Proceedings of Interspeech 2019, pp. 3505-3509. ISCA. DOI: 10.21437/Interspeech.2019-1858.](#)
- (3) [Z. Zeng, Y. Khassanov, V.T. Pham, H. Xu, E.S. Chng, and H. Li. On the End-to-End Solution to Mandarin-English Code-Switching Speech Recognition. In Proceedings of Interspeech 2019, pp. 2165-2169. ISCA. DOI: 10.21437/Interspeech.2019-1429.](#)

The contributions of the co-authors are as follows:

- In the first and second papers, Prof. Chng provided the initial project direction and assisted to edit the final manuscript drafts. I and Prof. Chng initialized the idea of proposed embedding matrix augmentation for vocabulary adaptation. I prepared the experimental setup and conducted the experiments. I also prepared the initial manuscript drafts.

- In the second paper, Mr. Zeng assisted in the construction of Singapore English speech recognition system. Dr. Pham and Dr. Xu provided the guidance during the experiment setup, assisted in the analysis and the interpretation of obtained results and revised the manuscript draft.
- In the third paper, Dr. Xu, Prof. Chng and Prof. Li provided the initial project direction and assisted to edit the manuscript drafts. Mr. Zeng prepared the experiment setup, conducted all speech recognition experiments and prepared the initial manuscript draft. Dr. Pham conducted the N-best hypotheses rescoring experiments. I was responsible for the language model vocabulary adaptation using the embedding matrix augmentation technique. All authors helped to revise the final manuscript draft.

Chapter 5 is published as [Y. Khassanov, H. Xu, V.T. Pham, Z. Zeng, E.S. Chng, C. Ni, and B. Ma. Constrained Output Embeddings for End-to-End Code-Switching Speech Recognition with Only Monolingual Data. In Proceedings of Interspeech 2019, pp. 2160-2164. ISCA. DOI: 10.21437/Interspeech.2019-1867.](#)

- I initialized the idea of constrained output embeddings for improving the code-switching capability of speech recognition systems.
- I prepared the experimental setup, conducted all experiments and prepared the initial manuscript draft.
- Dr. Xu and Dr. Pham guided the experiment setup and assisted to design the analyses.
- Mr. Zeng assisted in the preparation of the dataset and construction of the end-to-end speech recognition system.
- Prof. Chng, Prof Ma and Dr. Ni assisted in interpreting the obtained results and helped to revise the final manuscript draft.

6 Dec. 2019

.....

Date

.....

Yerbolat Khassanov

Acknowledgements

I would like to express my sincere thanks and appreciation to my supervisor Dr. Chng Eng Siong for his invaluable guidance, encouragement and support in both professional and personal matters. His wide knowledge and logical way of thinking led me to the success in the field of speech recognition and language modeling. I especially appreciate the time we spent discussing various topics that help me to broaden my horizons and to grow as a research scientist.

I also thank my colleagues at TL@NTU and Rolls-Royce@NTU corporate labs for their generous help. Especially, I would like to thank my colleagues and friends Dr. Xu Haihua, Dr. Pham Van Tung, Dr. Benjamin Bigot, Dr. Chong Tze Yuang, Mr. Zhiping Zeng, Mr. Kyaw Zin Tun and Mr. San Lin for their assistance and support. I thank all people in the school of computer science and engineering that help me to complete my Ph.D. program.

A special thanks to my parents and sister, for their consistent love and support. Finally, words cannot express how grateful I am to my beloved wife and son for all of the sacrifices, encouragement and support throughout the past five years.

Abstract

This research addresses the language model (LM) domain mismatch problem in automatic speech recognition (ASR) systems. The ASR systems rely on LMs to constrain its recognition output to linguistically correct hypotheses. While LMs significantly improve the linguistic competence of ASR, they are highly sensitive to the domain mismatch between training (source) and test (target) data. Even a slight difference between source and target domains might severely degrade LM’s effectiveness, and consequently harm the recognition performance of ASR.

Although LM domain mismatch is caused by the combination of various factors, in this work, we focus only on the following three factors: topic, vocabulary coverage and code-switching practice. In particular, we will first thoroughly describe each of these factors and their impact on the performance of LM and ASR, and then will propose three novel LM adaptation methods addressing each of them.

The first proposed method addresses the topic domain mismatch in count-based N-gram LMs employed at the decoding stage of the deep neural network (DNN) hidden Markov model (HMM) based hybrid ASR. The proposed method is based on the two-pass adaptation approach with the data selection technique. Different from the conventional two-pass adaptation methods that directly modify the parameters of LM using the recognition output from the first pass, our method instead modifies the training data by filtering out the irrelevant text segments. Consequently, our method avoids the error propagation caused by the re-usage of incorrect recognition output hypotheses and prevents the distortion of the captured linguistic knowledge.

The second proposed method addresses the vocabulary coverage mismatch in word-level recurrent neural network (RNN) LMs frequently used at the rescoring stage of both DNN-HMM and end-to-end (E2E) ASR systems. Despite the superior generalization capability of RNN LMs, its vocabulary coverage will be always limited to the words present in the training data. Whereas the remaining words, which

are usually rare domain-specific words, are either discarded or mapped to a special token such as <unk>. Consequently, the important keywords such as rare person and location names will be underrepresented, and thus, removed from the final recognition output. To overcome this problem, we propose an efficient vocabulary adaptation method based on the word embedding matrix augmentation. Our method employs the similar words to expand the vocabulary coverage or to enrich the representations of rare words in pre-trained RNN LM without requiring additional in-domain training data and expensive post-processing.

The third proposed method addresses the code-switching (CS) practice mismatch between training and test data in multilingual E2E ASR systems. Specifically, the training data consists of intersentential CS (inter-CS) type utterances, i.e. a practice of mixing languages between utterances. On the other hand, the test data consists of intrasentential CS (intra-CS) type utterances, i.e. a practice of mixing languages within utterances, which are considered more challenging. While the inter-CS data can be obtained by simply combining several monolingual corpora from different languages, the labeled intra-CS data is extremely difficult to obtain. Therefore, we propose an effective adaptation method to improve the intra-CS speech recognition capability of E2E ASR built using only abundant inter-CS data. Particularly, our method constrains the output token embeddings of monolingual languages, residing within internal LM of E2E ASR, to be close to each other, and hence promotes the E2E ASR to easily switch languages.

We evaluated all proposed methods on standard datasets using the state-of-the-art tools. We also compared the proposed methods against strong baseline systems where significant improvements were achieved.

Contents

Acknowledgements	ix
Abstract	xi
List of Figures	xvii
List of Tables	xix
List of Acronyms	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	4
1.2.1 Topic Domain Adaptation	4
1.2.2 Vocabulary Coverage Adaptation	5
1.2.3 Code-switching Practice Adaptation	6
1.3 Thesis Organization	7
2 Literature Review of LM Adaptation Techniques for ASR	9
2.1 Background	10
2.1.1 Automatic Speech Recognition	10
2.1.1.1 DNN-HMM Hybrid ASR Systems	10
2.1.1.2 End-to-End ASR Systems	13
2.1.1.3 ASR Evaluation	15
2.1.2 Statistical Language Models	16
2.1.2.1 N-gram Language Models	17
2.1.2.2 Recurrent Neural Network Language Models	18
2.1.2.3 Language Model Evaluation	20
2.1.3 Domain Mismatch Problem	21
2.1.3.1 Acoustic Model Domain Mismatch	22
2.1.3.2 Language Model Domain Mismatch	22
2.1.3.3 Domain Mismatch Impact on ASR	22
2.2 LM Adaptation Techniques for ASR	23
2.2.1 General LM Adaptation Framework	24

2.2.2	Information Combination Based Adaptation	26
2.2.2.1	Methodologies	26
2.2.2.2	Cache-based LM	28
2.2.2.3	Topic-mixture LM	29
2.2.2.4	The Pros and Cons	30
2.2.3	Model-based Adaptation	30
2.2.3.1	Fine-tuning	31
2.2.3.2	Linear Hidden Network Cascading	31
2.2.3.3	Linear Hidden Unit Contributions	32
2.2.3.4	The Pros and Cons	32
2.2.4	Feature-based Adaptation	33
2.2.4.1	Context dependent LM	33
2.2.4.2	Factored LM	34
2.2.4.3	The Pros and Cons	34
2.2.5	Hybrid Adaptation	35
2.3	Summary	36
3	Language Model Adaptation by Data Selection	39
3.1	Introduction	40
3.1.1	Data Selection Framework	41
3.1.2	Applications of Data Selection	42
3.1.3	Overview of Data Selection Techniques	45
3.1.3.1	Entropy Based	45
3.1.3.2	Vector Space Model Based	48
3.1.3.3	Others	50
3.2	Two-pass LM Adaptation Framework based on Data Selection . . .	52
3.3	Experiment	55
3.3.1	Datasets	55
3.3.2	ASR and LM Configurations	56
3.3.3	Experiment Setup and Results	57
3.4	Summary	62
4	Neural LM Vocabulary Adaptation by Embedding Matrix Augmentation	63
4.1	Introduction	64
4.1.1	Neural LM Architecture	65
4.1.2	Embedding Matrices	66
4.1.3	Vocabulary Coverage Mismatch	67
4.1.3.1	Out-of-shortlist Words	68
4.1.3.2	Underrepresented Words	68
4.2	Review of Vocabulary Adaptation Methods	69
4.2.1	Adaptation Methods for Out-of-shortlist Words	69
4.2.2	Adaptation Methods for Rare Word Representations	71
4.3	Embedding Matrix Augmentation	72

4.3.1	Expanding Vocabulary Coverage	73
4.3.2	Enriching Rare Word Representations	76
4.4	Experiments	78
4.4.1	Experiments for Vocabulary Coverage Expansion	78
4.4.1.1	Experiment Setup and Data	78
4.4.1.2	Experiment Results and Discussion	80
4.4.2	Experiments for Rare Word Representation Enriching	81
4.4.2.1	Experiment Setup and Data	82
4.4.2.2	Experiment Results and Discussion	83
4.5	Summary	86
5	Code-Switching Practice Adaptation by Constrained Output Embeddings	87
5.1	Introduction	88
5.1.1	Code-switching	89
5.1.2	End-to-end ASR Architecture	91
5.2	Related Works	95
5.2.1	Multi-pass Systems	95
5.2.2	Multilingual Systems	96
5.2.3	Semi-supervised Learning	96
5.2.4	Transfer Learning	96
5.2.5	Data Synthesis	97
5.3	Constraining Output Embedding Matrix	97
5.3.1	Constraint Types	98
5.3.1.1	Jensen-Shannon Divergence	99
5.3.1.2	Cosine Distance	99
5.4	Experiment	100
5.4.1	Experiment Setup	100
5.4.1.1	Dataset	100
5.4.1.2	Baseline E2E-CS-ASR configuration	101
5.4.2	Experiment Results and Analysis	101
5.4.2.1	Results	102
5.4.2.2	Analysis	103
5.5	Summary	106
6	Conclusions and Future Work	107
	List of Author's Awards, Patents, and Publications	111
	Bibliography	113

List of Figures

2.1	A DNN-HMM hybrid ASR architecture.	11
2.2	A word lattice generated for the input test utterance “ <i>smells good too</i> ”. The arcs are labeled in the form of ‘ <i>word_id:word</i> ’. The state transition probabilities are not shown for clarity.	12
2.3	A WFST example of the pronunciation model. The arcs are labels in the form of ‘ <i>input_symbol:output_symbol/weight</i> ’. The ‘ <i>input_symbol</i> ’ is a phoneme, the ‘ <i>output_symbol</i> ’ is a word and the ‘ <i>weight</i> ’ is a transition probability. The ‘ <i><eps></i> ’ corresponds to empty output. Observe that the word ‘ <i>good</i> ’ has alternative pronunciations.	13
2.4	A WFST example of a simple language model. The language model constrains the search space to linguistically plausible word sequences, therefore, it’s a mapping from words to words, i.e. both the ‘ <i>input_symbol</i> ’ and the ‘ <i>output_symbol</i> ’ are words.	13
2.5	Joint CTC/Attention based end-to-end ASR architecture.	14
2.6	Recurrent neural network language model architecture.	19
2.7	General LM adaptation framework.	24
2.8	Architectural changes in model-based adaptation techniques: (a) vanilla RNNLM, (b) linear hidden unit cascading, (c) linear hidden unit contributions.	32
2.9	The context dependent RNN LM architecture. The context feature vector f_t can be input into hidden or output or both layers.	34
3.1	Data selection framework.	41
3.2	Proposed two-pass LM adaptation framework based on data selection.	52
3.3	WER results obtained by the proposed LM adaptation framework.	57
3.4	Perplexity results of adapted LMs computed on reference data.	58
3.5	WER results for finer-level domain data selection using CED with 2-gram LM.	59
3.6	WER results for finer-level domain data selection using VSM with TFIDF features.	60
3.7	The comparison of WER results obtained for single speakers (Bill Gates and Dan Barber) using two different data selection techniques (CED and VSM) under two different domain settings (TED and specific).	61

4.1	Recurrent NLM architecture decomposed into three parts.	65
4.2	A word embedding space visualized with t-SNE. Each point corresponds to one word.	67
4.3	NLM architecture after vocabulary expansion.	76
4.4	WER performance of E-LSTM at different frequency threshold values used to split frequent and rare street names.	84
5.1	Hybrid CTC/Attention end-to-end ASR architecture with constrained output token embeddings. The output token embeddings are learned by the parametric matrix of linear output projection layer (Output-Proj).	92
5.2	The impact of CD and JSD constraint interpolation weights on MER performance for test _{eng} (red/top) and test _{man} (blue/bottom) sets. .	104
5.3	PCA visualization of shared output token embedding space without (a) and with (b,c,d) proposed constraints.	105

List of Tables

3.1	TED-LIUM corpus characteristics.	55
3.2	TED-LIUM corpus test set details.	56
3.3	The best WERs (%) achieved by different data selection techniques.	61
4.1	The characteristics of TED-LIUM corpus.	79
4.2	The perplexity and WER results on dev and test sets of TED-LIUM corpus.	80
4.3	The perplexity and WER results on evaluation set. The "+KN4" represent results obtained by interpolating neural LMs with baseline KN4 LM. The E-LSTM is obtained by enriching rare words in baseline LSTM.	84
4.4	Recognition accuracy of 265 rare street names.	85
4.5	Examples of correctly recovered neighbouring words after rescoring with E-LSTM.	85
5.1	SEAME dataset statistics after removing the CS utterances from the train set. 'Man' and 'Eng' refer to Mandarin and English languages, respectively.	101
5.2	The MER (%) performance of different ASR models built using monolingual data. The test sets are further split into monolingual (mono) and code-switching (CS) utterances.	103
5.3	The MER performance after applying the language model during the decoding stages.	105

List of Acronyms

AM	Acoustic Model
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
BLSTM	Bidirectional Long Short Term Memory
BLEU	Bilingual Evaluation Understudy
BOW	Bag of Words
BPE	Byte Pair Encoding
BPTT	Backpropagation Through Time
CD	Cosine Distance
CE	Cross Entropy
CED	Cross Entropy Difference
CNN	Convolutional Neural Network
CS	Code-Switching
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
FNN	Feedforward Neural Networks
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IC	Information Combination
IDF	Inverse Document Frequency
IV	In-vocabulary
JSD	Jenson-Shannon Divergence
KLD	Kullback-Leibler Divergence
KN	Kneser-Ney
LDA	Latent Dirichlet Allocation
LM	Language Model
LSA	Latent Semantic Analysis

LSI	Latent Semantic Indexing
LSTM	Long Short Term Memory
LVCSR	Large-vocabulary Continuous Speech Recognition
MER	Mixed Error Rate
MFCC	Mel Filterbank Cepstral Coefficients
MLE	Maximum Likelihood Estimate
MLP	Multi-layer Perceptron
MTL	Multi-task Learning
NERC	Named Entity Recognition and Classification
NLM	Neural Language Model
NLP	Natural Language Processing
NN	Neural Network
OOS	Out of Shortlist
OOV	Out Of Vocabulary
PCA	Principal Component Analysis
PLP	Perceptual Linear Prediction
PM	Pronunciation Model
PPL	Perplexity
RBM	Restricted Boltzmann Machine
RE	Relative Entropy
RNN	Recurrent Neural Network
RTF	Real Time Factor
SMT	Statistical Machine Translation
SP	Speed Perturbation
TF	Term Frequency
TM	Translation Model
TTS	Text to Speech
VSM	Vector Space Model
WER	Word Error Rate
WFSA	Weighted Finite State Acceptor
WFST	Weighted Finite State Transducer

Chapter 1

Introduction

1.1 Motivation

Designing a machine that can mimic complex human behaviors such as understanding spoken language and responding accordingly has been envisioned long before the advancement of computers. The major step in fulfilling this vision is to develop automatic speech recognition (ASR) systems that transform speech into text. The ASR task has intrigued researchers for centuries and in the past several years its interest has been further boosted by its new applications in the internet of things and mobile devices such as voice command, voice search, message dictation, and virtual assistants (e.g. Google Now, Apple’s Siri, Microsoft’s Cortana and Amazon’s Alexa) [1].

A brief history of early speech recognition systems. Given the complexity of human language, the speech recognition technology evolved gradually. The first speech recognition systems focused on simple tasks such as recognizing numbers. For example, in the 1950s, Bell Laboratories designed Audrey [2] which is a first known and documented speech recognizer. Audrey could recognize ten digits (0-9) spoken isolatedly by a single speaker with an accuracy of 90%; accuracy hovered between 70% and 80% when other people spoke to it. In the early 1960s, IBM demonstrated Shoebox¹, a system which could recognize sixteen words, including ten digits and six arithmetic operations. Over the next decade, speech recognition technology advanced progressively from a simple machine that could recognize a

¹https://en.wikipedia.org/wiki/IBM_Shoebox

few words to a sophisticated system recognizing speech with a large vocabulary. Notably, in the 1970s, DARPA initiated speech understanding research program which was responsible for Carnegie Mellon’s Harpy [3] system. Harpy could recognize speech using a vocabulary of 1,011 words, approximately the vocabulary of an average three-year-old.

With increased vocabulary, however, the complexity of the task considerably grew. Particularly, the confusion attributed to homophones. For example, the words ‘buy’, ‘bye’ and ‘by’ comprise the same phoneme sequence ‘B AY’ according to the ARPabet² phoneme set. Distinguishing such words was an infeasible task for the early speech recognition systems that mainly relied on acoustic observations. Therefore, the recognition capability of early ASR systems was limited.

Introduction of language models in ASR. Relying solely on acoustic observations proved to be insufficient to achieve human-like performance. In search of other sources of knowledge, Jelinek et al. [4] attempted to incorporate linguistic regularities of the natural language into the ASR. He proposed to encode the linguistic regularities into a so-called language model (LM) using the statistical principles. The duty of the LM was to encapsulate syntactic, semantic, and pragmatic properties of the language in order to improve the linguistic competence of ASR. Specifically, in ASR, encapsulated knowledge was used to constrain the decoder’s search space to linguistically correct word sequences. The consequence of LM integration was faster search and higher recognition accuracy. Since then, LMs became an indispensable part of large vocabulary speech recognition systems.

The domain mismatch problem. The LMs represent the encapsulated knowledge in the form of probability distribution of linguistic units (e.g. characters, subwords, words) estimated from some textual training data [5]. The estimated distribution, however, may differ from the distribution present in the test data, i.e. input utterances submitted to ASR system. For example, the training and test data might cover different topics, speaking styles, genres and so on. This will lead to domain mismatch problem between training and test data which has been shown to severely degrade the effectiveness of LM [6]. Subsequently, the domain mismatched LM will deteriorate the recognition performance of ASR. For example, a LM trained on industry domain data, but applied to input utterances from the math domain, may force the ASR to recognize ‘*cofactor is*’ as ‘*cow factories*’ (the

²<https://en.wikipedia.org/wiki/ARPABET>

Hamming distance between phoneme sequence of these phrases is 1). It is therefore desirable to choose the training data from the same domain as the test data.

A brief review of LM adaptation techniques. To alleviate the impact of domain mismatch, several effective LM adaptation techniques have been proposed [7, 8]. In general, these techniques can be classified into the following four groups: information combination based, model-based, feature-based and hybrid. The information combination based techniques combine several LMs, e.g. using the linear interpolation, to reflect the word distribution present in the target domain. The popular information combination techniques include the cache-based LMs [9] and the topic-mixture LMs [10]. On the other hand, the model-based adaptation techniques [11, 12] change the architecture of pre-trained LM to enable it to capture domain-specific knowledge. In the same spirit, the feature-based adaptation techniques [13, 14] augment the input to the model with auxiliary domain-specific feature, e.g. topic and genre features. The hybrid adaptation techniques are the any combination of the previous three techniques.

Although existing LM adaptation techniques were demonstrated to be effective in some scenarios, there are three main concerns remain open:

- 1) Most of the existing techniques rely on the presence of some target seed data or other domain-specific knowledge. However, in real-world applications, such knowledge might be non-existing, leaving all these techniques impractical.
- 2) Some of the techniques use the recognition output from the first-pass as a pseudo in-domain data to adapt a LM. However, these techniques are susceptible to the recognition errors which lead to error propagation problem. In addition, these techniques directly modify the parameters of LM which might lead to the loss of important linguistic knowledge obtained from the human generated text.
- 3) Existing techniques are mainly designed to operate in monolingual space focusing on topic or genre level domains. Unfortunately, they don't address other aspects of a domain such as vocabulary coverage and code-switching practice.

In this thesis, keeping the aforementioned concerns in mind, we propose three novel LM adaptation techniques.

1.2 Contributions

The main contributions of this thesis are centered around three novel LM adaptation techniques. The first technique addresses the topic domain mismatch, the second technique addresses the vocabulary coverage mismatch and the last technique addresses the code-switching practice mismatch. Although, each addressed mismatch can be considered a separate research problem, we view them from the point of general domain mismatch problem between training and testing conditions, and attempt to mitigate them by manipulating language models.

1.2.1 Topic Domain Adaptation

We first focus on topic domain mismatch problem in count-based N-gram LMs employed at the decoding stage of the deep neural network (DNN) hidden Markov model (HMM) based hybrid ASR. Due to the statically-compiled structure of DNN-HMM hybrid ASR, the dynamic adaptation of N-gram LM is infeasible. Therefore, we resort to two-pass adaptation approach where the recognition output from the first-pass is employed to adapt LM. However, unlike the existing two-pass adaptation approaches that directly apply the recognized output to modify the parameters of LM, we instead propose to adapt the training data using the data selection techniques. Specifically, we propose to use the recognized output to filter out irrelevant sentences from the training data. The filtered training data is then used to build a domain-specific LM which is employed at the second-pass decoding. The motivation is that the training corpus consists of the data from different domains and as such, the LM trained from it is generic and not discriminative. Besides, our method partially prevents the error propagation as the recognized output is used indirectly, i.e. to select relevant data. Furthermore, the LM adapted using our approach will encapsulate better linguistic knowledge, since instead of modifying its parameters we modified the training data where the wholeness of the selected sentences is kept unchanged. The comprehensive analysis of the proposed method will be presented in Chapter 3.

The proposed method was documented and published in:

- 1) Khassanov Y., Chong T. Y., Bigot B. and Chng E. S., “Unsupervised Language Model Adaptation by Data Selection for Speech Recognition.” In Asian

Conference on Intelligent Information and Database Systems, 2017, pp. 508-517. Springer, Cham.

1.2.2 Vocabulary Coverage Adaptation

We next explore the vocabulary coverage issue in the word-level recurrent neural network (RNN) LMs. The RNN LMs help ASR systems to achieve the state-of-the-art results by rescoring the output word lattices or N-best hypotheses list. One of the main drawbacks of the word-level RNN is its vocabulary coverage which is limited to the frequent words in the training data. Whereas the remaining rare words and the words not present in the training data are deemed out-of-vocabulary (OOV) and ignored, e.g. replaced by special *<unk>* symbol. However, the OOV words might be important domain-specific words such as person, location and organization names. For example, in voice search task [15], the correct recognition of the street name (location) and the restaurant name (organization) is crucial. To mitigate this problem, we propose an efficient vocabulary adaptation method. The proposed method can be used to expand the vocabulary coverage or to enrich the rare word representations in pre-trained RNN LM. Specifically, our method exploits the representations of similar words to augment the embedding matrices in input and output layers, while keeping other layers of RNN LM intact. Importantly, the proposed method doesn't require additional in-domain data and doesn't involve expensive post-processing procedures. The thorough description and evaluation of the proposed approach will be provided in Chapter 4.

The proposed method was documented and published in:

- 1) Khassanov Y. and Chng, E.S., "Unsupervised and Efficient Vocabulary Expansion for Recurrent Neural Network Language Models in ASR." In Proc. of Interspeech, 2018, pp. 3343-3347.
- 2) Khassanov Y., Zeng Z., Pham V.T., Xu H. and Chng E.S., "Enriching Rare Word Representations in Neural Language Models by Embedding Matrix Augmentation." In Proc. of Interspeech, 2019, pp. 3505-3509.
- 3) Zeng Z., Khassanov Y., Pham V.T., Xu H., Chng E.S. and Li H., "On the End-to-End Solution to Mandarin-English Code-switching Speech Recognition." In Proc. of Interspeech, 2019, pp. 2165-2169.

1.2.3 Code-switching Practice Adaptation

In this study, we focus on improving the code-switching (CS) capability of the attention-based end-to-end (E2E) ASR built using only monolingual data. The CS is a practice of alternating between several languages within single discourse common to multilingual societies such as USA (e.g. Spanish-English), Singapore (e.g. Mandarin-English) and India (e.g. English-Hindi). There are two main types of CS: intersentential (inter-CS) and intrasentential (intra-CS). In the inter-CS, the language switch takes place only on the utterance boundaries. While in the intra-CS, the language switch might take place within an utterance as well, and thus considered more challenging. With the growth of code-switching trend, the existing monolingual ASR systems faced a serious problem. Furthermore, the manually transcribed CS data is extremely rare for building robust CS ASR systems, especially the intra-CS type. To circumvent this problem, we propose to improve the intra-CS speech recognition capability of the E2E ASR trained on a combination of several monolingual data from different languages which is akin to inter-CS type. Our work can be also interpreted as an adaptation from the inter-CS practice to the intra-CS practice. In particular, our method enforces the output token distributions of monolingual languages, residing withing implicit LM of E2E ASR, to be close to each other. Consequently, the closely located output token embeddings will facilitate the E2E ASR to easily switch between languages. Our method doesn't rely on any cross-lingual resources such as labeled CS seed data, word-aligned parallel data or dictionary. The detailed description of the proposed approach will be provided in Chapter 5.

The proposed method was documented and published in:

- 1) Khassanov Y., Xu H., Pham V.T., Zeng Z., Chng E.S., Ni C. and Ma B. "Constrained Output Embeddings for End-to-End Code-Switching Speech Recognition with Only Monolingual Data." In Proc. of Interspeech, 2019, pp. 2160-2164.

1.3 Thesis Organization

The thesis is organized into five chapters. Chapter 1 is devoted to motivation, the main contributions of this work and thesis organization.

In Chapter 2, we first provide background information on ASR systems, statistical LMs, and domain mismatch problem. We then review the state-of-the-art LM domain adaptation techniques employed in speech recognition.

In Chapter 3, we propose a novel topic adaptation method based on data selection for count-based N-gram LM used in the conventional DNN-HMM ASR systems. We first introduce the data selection concept including the general framework, its applications and review of the state-of-the-art data selection techniques. We then present our proposed data selection based adaptation framework. Lastly, we describe the experiments conducted to evaluate the proposed framework and discuss the obtained results.

In Chapter 4, we propose a vocabulary adaptation method for RNN LMs. We start with a description of the vocabulary mismatch problem and its importance for ASR. We then present our proposed solution, including a comprehensive description and mathematical formulations. Next, we explain the experiment setup and data used to evaluate the proposed method, followed by the discussion of the obtained results.

In Chapter 5, we present the code-switching practice adaptation method for attention-based E2E ASR. We first describe different code-switching practices and challenges that they pose to existing speech and language processing applications. We then explain the proposed solution, followed by experiment setup and data. Lastly, we reveal the obtained results and discuss important observations.

Chapter 6 concludes the thesis by highlighting important contributions, observations, and suggestions for future research.

Chapter 2

Literature Review of LM Adaptation Techniques for ASR

This chapter introduces the key components and concepts involved in the language model (LM) adaptation process, followed by the review of the state-of-the-art LM adaptation techniques. We split this chapter into the two main sections as follows.

In section 2.1, we first expose the readers to the background information on the automatic speech recognition (ASR) systems, the statistical LMs and the domain mismatch problem. Particularly, in subsection 2.1.1, we briefly describe the state-of-the-art ASR architectures, including the hybrid deep neural network (DNN) hidden Markov model (HMM) and purely neural end-to-end (E2E) architectures. Since our main concern is a LM adaptation, in subsection 2.1.2, we delve deeper into the LMs by providing mathematical formulations and describing the state-of-the-art LM approaches. Lastly, in subsection 2.1.3, we describe the domain mismatch problem and its impact on the performance of LM, and subsequently, on the entire ASR system.

In section 2.2, we review the state-of-the-art LM adaptation techniques proposed to deal with the domain mismatch problem in ASR. Although these techniques may have different schemes and mechanisms, they share a common general framework described in subsection 2.2.1. We then describe the LM adaptation techniques categorized into the four main sub-groups: 1) Information combination (subsection 2.2.2), 2) Model-based (subsection 2.2.3), 3) Feature-based (subsection 2.2.4), and 4) Hybrid (subsection 2.2.5).

2.1 Background

This section provides supplementary background knowledge required to understand the following chapters. We first describe the state-of-the-art ASR architectures (subsection 2.1.1) including both conventional hybrid ASR and recently proposed end-to-end ASR. The hybrid ASR will be employed in Chapter 3 and Chapter 4, while the end-to-end ASR is employed in Chapter 5. Next, we will review statistical LMs (subsection 2.1.2) including both traditional N-gram LMs and neural LMs. The N-gram LMs will be employed in Chapter 3 and the neural LMs will be employed in Chapter 4 and Chapter 5. Lastly, we will describe the domain mismatch problem (subsection 2.1.3).

2.1.1 Automatic Speech Recognition

The speech recognition research has a long history where it has progressed from simple systems that could recognize a few words to more complicated systems that can recognize thousands of words. The latest major progress in speech recognition took place at the beginning of the 2010s when the DNN has taken over the dominating Gaussian Mixture Model (GMM) as the output distribution in the HMM [16–20]. More recently, purely neural E2E architectures have emerged as a potential successor of DNN-HMM [21–25]. Although E2E models greatly simplify the ASR building pipeline and address some of the important limitations of DNN-HMM, they are still considered immature, achieving worse or comparable results to DNN-HMM. In this subsection, we briefly describe and compare these two competing the state-of-the-art ASR architectures.

2.1.1.1 DNN-HMM Hybrid ASR Systems

The principal modules of the DNN-HMM hybrid ASR are depicted in Figure 2.1. These modules collaborate to perform a speech recognition task as follows.

Feature extraction. First, the feature extraction module converts a raw input audio signal into a sequence of acoustic feature vectors, typically mel-frequency cepstral coefficients (MFCCs) [26] or perceptual linear predictive coefficients (PLPs) [27]

FIGURE 2.1: A DNN-HMM hybrid ASR architecture.

concatenated with their first- and second-order temporal differences [28]. The acoustic features are usually computed every 10ms over 25ms window.

Decoder. Produced acoustic feature vectors are then deployed by the decoder module. Specifically, the decoder searches through all possible word sequences in a search space to find the sequence which best conforms with the input acoustic features. The search is usually performed using dynamic programming such as the Viterbi algorithm [29]. The output of the decoder is a word lattice.

Word lattice. The word lattice is a graph comprising the most likely word sequences corresponding to the input features (see Figure 2.2). The graph is a weighted finite-state acceptor (WFSA) consisting of states and directed arcs [30]. By convention, the states are represented by a circle with a unique number. The bold circles are called initial states and the double circles are called final states. The directed arcs connect states and are labeled using the transition cost (e.g. probability) and word (or word id). The legal word sequence is a path along arcs from initial to final state, and its probability is given by the product of the corresponding transition probabilities. The most likely path is presented as a final transcription. The N most likely paths are called the N -best hypotheses list. The size of the word lattice is regulated using the beam size.

Search space. While the word lattice contains the most likely word sequences, the search space comprises all possible word sequences (or paths) that can be recognized by ASR. Therefore, it can be of astronomical size, especially in the large vocabulary systems. Additionally, the search space is represented using a weighted finite-state transducer (WFST) [31] which generalizes WFSA. In particular, the search space WFST maps the input acoustic features to the word sequence.

FIGURE 2.2: A word lattice generated for the input test utterance “*smells good too*”. The arcs are labeled in the form of ‘*word_id:word*’. The state transition probabilities are not shown for clarity.

Acoustic, pronunciation and language models. The search space WFST is obtained by composing other three transducers¹ which represent: acoustic model (AM), pronunciation model (PM), and LM. These transducers are typically mappings at different levels of linguistic units. For example, the AM transducer maps the acoustic features to the corresponding phoneme sequence. The PM transducer (see Figure 2.3) maps the phoneme sequence to the word sequence. Lastly, the LM transducer (see Figure 2.4) maps the unregulated word sequences to grammatically correct word sequences. Thence, by composing these three transducers, we obtain a single transducer which directly maps the input acoustic features to the grammatically plausible sequence of words.

It is important to mention that the core component of hybrid ASR systems is AM which is based on DNN-HMM model (the name used to generally call such ASR systems). Particularly, HMM is used to model temporal dependencies between acoustic features, while DNN models the phoneme emission probability in HMM states. For an in-depth description of hybrid ASR systems, refer to [1, 32, 34]. For the existing toolkits to implement hybrid ASR systems, check Kaldi [35], RWTH Aachen [36], HTK [37], Sphinx [38] and Julius [39]. In the following section, we will describe purely DNN based E2E ASR architectures.

¹In fact, the main WFST is obtained by composing four sub-module WFSTs. However, for the sake of simplicity, we omitted the context-dependency WFST. For more details about transducers please refer to [31–33].

FIGURE 2.3: A WFST example of the pronunciation model. The arcs are labels in the form of ‘*input_symbol:output_symbol/weight*’. The ‘*input_symbol*’ is a phoneme, the ‘*output_symbol*’ is a word and the ‘*weight*’ is a transition probability. The ‘<eps>’ corresponds to empty output. Observe that the word ‘*good*’ has alternative pronunciations.

FIGURE 2.4: A WFST example of a simple language model. The language model constrains the search space to linguistically plausible word sequences, therefore, it’s a mapping from words to words, i.e. both the ‘*input_symbol*’ and the ‘*output_symbol*’ are words.

2.1.1.2 End-to-End ASR Systems

Different from the hybrid ASR, the E2E systems are purely neural network that directly map the input acoustic features into the graphemes (e.g. characters, sub-words or words). Over the past few years, several varieties of E2E ASR have been proposed such as connectionist temporal classification (CTC) [21], recurrent neural network (RNN) transducer [40, 41], RNN aligner [42], neural transducer [43, 44] and attention-based encoder-decoder [25, 45, 46]. Recently, Kim et al. [47] proposed joint CTC/Attention E2E ASR architecture, incorporating the advantages of both CTC and attention-based encode-decoder models. The joint E2E architectures have been shown to achieve the state-of-the-art results on various speech recognition tasks for different languages [48–50]. In this subsection, we will describe only the joint CTC/Attention architecture, since it was used in our experiments.

Joint CTC/Attention model. The overall architecture of the joint CTC/Attention model is depicted in Figure 2.5 where the main modules are the feature extractor, the shared encoder, and the joint decoder. These modules interact to perform a speech recognition task as follows.

FIGURE 2.5: Joint CTC/Attention based end-to-end ASR architecture.

Feature extraction. The feature extraction module plays a similar role as in conventional hybrid ASR, i.e. converts a raw audio signal into the acoustic features (e.g. log-mel filterbank coefficients).

Shared encoder. The shared encoder module transforms the extracted low-level acoustic features into the high-level features. To reduce the computational complexity, the low-level features are often subsampled. The encoder is usually modeled using the stacked bidirectional long short-term memory (BLSTM) networks, e.g. 4-8 layers. Note that encoder plays a similar role as the acoustic model in hybrid ASR systems.

Joint decoder. The joint decoder module is responsible for generating output symbols corresponding to the input speech signal. It consists of two main parts: attention-based decoder and CTC networks.

1) *Attention-based decoder.* The attention-based decoder is comprised of two main components: attention and decoder. The attention component performs alignment between high-level acoustic features and output symbols. Specifically, at every time step, it points (or attends) to the most relevant high-level features required to generate the current symbol. The output of the attention is a so-called context

vector produced by taking a weighted average of high-level acoustic features. The produced context vector is then processed by the decoder component which is modeled as the stacked unidirectional LSTM networks, e.g. 1-2 layers. Note that the decoder component is akin to LM in hybrid ASR, except it is conditioned on the context vector.

2) *CTC*. On the other hand, the CTC network directly maps the high-level acoustic features into the output symbols². It is modeled using fully connected feed-forward networks. The reason for incorporating the CTC network is a too flexible attention-based decoder that allows non-sequential alignments. The non-sequential alignment is suitable for applications such as machine translation where the word order in target and source sentences might be different. However, in speech recognition, the input features and the output symbols generally follow the same order. Therefore, the CTC network is employed to enforce a monotonic alignment between input and output sequences.

E2E ASR vs. hybrid ASR. While the hybrid ASR models its core modules separately using different objectives, the E2E ASR models them under a single network using the same objective function. Therefore, the E2E ASR imposes the optimization coherence and greatly simplifies the ASR building pipeline [48]. Moreover, the E2E ASR can be used to recognize character or subword units, as such, it is immune to the out-of-vocabulary problem. Additionally, the E2E ASR doesn't require a handcrafted pronunciation model which is prone to human errors, instead, it fully relies on the data-driven learning approaches. Consequently, the E2E ASR systems require more training data and are considered 'data-hungry'; in low and moderate resource scenarios they are often inferior to hybrid ASR. To implement the E2E ASR, check the following toolkits: ESPNET [51], RETURNN [52], wav2letter [53] and EESSEN [54].

2.1.1.3 ASR Evaluation

Word error rate. The most popular metric of performance of an ASR system is word error rate (WER). The WER is derived from the Levenshtein distance working at the word level between a reference and a hypothesis. It is computed as

²Due to the sequence length difference between high-level acoustic features and output symbols, the CTC network employs additional special symbol called 'blank'.

follows:

$$\text{WER} = \frac{S + D + I}{N} \quad (2.1)$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions and N is the number of words in the reference. The WER is 0 when the hypothesis is the same as the reference. Note that the WER can get arbitrarily large, because the ASR can insert a large amount of words.

Other metrics. Another important metric that should be taken into consideration during the ASR system comparison is the recognition time. The recognition time is usually measured in terms of real time factor (RTF), i.e. how long does the processing takes compared to the length of the input recording. A good system should reach RTF close to 1, i.e. the processing is done in real-time. Additionally, the size of the ASR model and the required computational resources are also important, especially for the small scale computers with the limited memory and the processing power, e.g. internet of things and mobile devices.

2.1.2 Statistical Language Models

The goal of the statistical LM is to encapsulate linguistic knowledge such as syntactic, semantic and pragmatic from the textual training data [5, 6]. The encapsulated knowledge can then be used to quantify ‘correctness’ of the word sequences, e.g. sentence, paragraph or document. The LMs are especially useful in applications that produce word sequences such as speech recognition, machine translation, handwriting recognition, image captioning, question answering and so on.

In general, statistical LMs evaluate the ‘correctness’ of a word sequence $W = (w_1 \ w_2 \ \dots \ w_T) = w_1^T$ by assigning it some score such as joint probability. The joint probability of the entire word sequence is usually computed as a product of the conditional probabilities of individual words using the Chain rule as follows:

$$P(W) = P(w_1^T) = P(w_1) \prod_{t=1}^{T-1} P(w_{t+1}|w_1^t) \quad (2.2)$$

However, estimating the conditional probability of a word w_{t+1} given its full context w_1^t is problematic due to the data sparsity problem [55]. An effective solution to deal with the unseen contexts is to truncate them, since shorter contexts are more

likely to be seen. The family of statistical LMs that employ truncated contexts can be generally classified under N-gram models which are described next.

2.1.2.1 N-gram Language Models

The N-gram models are based on Markov assumption where instead of conditioning a word probability on its full context, only the preceding $N-1$ words are used³. For example, in 3-gram (or trigram) model, the context is limited to two previous words. Thus, the probability of a word sequence is approximated as follows:

$$P(W) = P(w_1^T) \approx P(w_1)P(w_2|w_1) \prod_{t=2}^{T-1} P(w_{t+1}|w_{t-1}^t) \quad (2.3)$$

Instead of conditioning only on the surface form of $N-1$ words, other linguistic features (or factors) can be used such as part-of-speech, stems, lemma and morphemes. Such models are generally referred to as *factored language models* [56, 57] which aim to exploit auxiliary linguistic information beyond the simple surface form of the words. For example, 2-factor LM generalizes the *class-based LM* [58, 59] by assigning one factor to word and the second factor to its class. Other popular N-gram LM varieties include *skip-gram* model [5] which skips an arbitrary word in the context, e.g. $P(w_{t+1}|w_{t-2} \dots w_t)$, and *distant-d* model [60, 61] which skips immediately preceding $d-1$ words, e.g. $P(w_{t+1}|w_{t-2} \dots w_{t-1})$.

N-gram LM realization. The N-gram LM can be realized using various machine learning models such as feed-forward neural networks (FNN) [62–64], maximum entropy [65, 66], decision tree [67, 68] and count-based approaches [5]. Among them, count-based N-gram models are probably the most dominant LM type owing to its simplicity and effectiveness.

Count-based N-gram LM. The parameters of count-based N-gram LM are estimated by simply counting N-gram sequences in the training data. This approach is derived using the maximum likelihood estimation. Subsequently, a conditional probability of a word in the trigram model is estimated as follows:

$$P(w_{t+1}|w_{t-1}^t) = \frac{C(w_{t-1} \ w_t \ w_{t+1})}{\sum_{w \in V} C(w_{t-1} \ w_t \ w)} \quad (2.4)$$

³ N is usually set to 3-5.

where $C(w_{t-1} w_t w_{t+1})$ is the count of 3-gram word sequence $w_{t-1} w_t w_{t+1}$ in the training data and V is a vocabulary. Such estimation, however, is prone to the data sparsity problem, especially for the rare word sequences, resulting in unreliable probability estimates. To alleviate the data sparsity problem, the smoothing techniques [5, 69] are employed which borrow the probability mass from the frequent sequences and re-distribute to the rare sequences.

One of the main advantages of the count-based N-gram LM is that its parameters can be stored in a list form as a mapping from the word sequences to the probability scores. Subsequently, this list can be transformed into the WFST representation that can be embedded in the search space of DNN-HMM ASR. Therefore, the count-based N-gram models remain one of the dominant LM types used in ASR. The main drawback of the count-based N-gram is obviously its short-range context dependency where words outside of $N-1$ sized context are ignored. The short-range context dependency has been addressed by full context models such as latent semantic analysis LM [70, 71] and recurrent neural network (RNN) LM [72]. Notably, RNN LMs have been shown to possess strong generalization capability and achieve the state-of-the-art results in many speech and language processing applications.

2.1.2.2 Recurrent Neural Network Language Models

The main idea behind RNN LM is to replace a full context history by its equivalent representation, i.e. $w_1^t \approx h_t$, and compute the conditional probability $P(w_{t+1}|h_t)$ [72, 73]. The equivalent representation is learned during the training process by mapping the words into the continuous space vector representations [63], also known as word embeddings. The word embeddings have been shown to capture important linguistic information [74, 75], and therefore, can be used to represent any sequence of words including the ones unseen during the training.

The conventional RNN LM architecture is depicted in Figure 2.6. The input to the network is a word represented by the one-hot encoding vector $w_t \in \mathbb{R}^{|V|}$ where t is a time index and $|V|$ is a vocabulary size. The one-hot vector is then mapped into the continuous vector representation s_t using the embedding matrix $\mathbf{S}$:

$$s_t = \mathbf{S}w_t \quad (2.5)$$

FIGURE 2.6: Recurrent neural network language model architecture.

The embedding vector s_t and a high-level context feature vector from the previous time step⁴ h_{t-1} are then combined by a non-linear hidden layer, which can be represented as function $f(\cdot)$, to produce a new full context feature vector h'_t :

$$h'_t = f(s_t, h_{t-1}) \quad (2.6)$$

The high-level context feature vector h_{t-1} is obtained by the linear projection of the previous hidden layer output h'_{t-1} using the parametric matrix $\mathbf{W}$:

$$h_{t-1} = \mathbf{W}h'_{t-1} \quad (2.7)$$

The non-linear function $f(\cdot)$ can employ simple activation units such as ReLU and hyperbolic tangent or more complex memory units such as LSTM and GRU. The non-linear hidden layers can also be formed by stacking several such functions.

Lastly, the context vector h'_t is mapped to the high-dimensional vector $y_t \in \mathbb{R}^{|V|}$ using the output embedding matrix $\mathbf{U}$:

$$y_t = \mathbf{U}^T h'_t \quad (2.8)$$

The entries of the output vector y_t represent the scores of words (or logits) to follow the context h_t . These scores are then normalized by the softmax function to form a probability distribution $P(w_{t+1}|h_t)$.

Although RNN LMs have been shown to achieve the state-of-the-art results in language modeling task [72], they are not suitable to be embedded in the search space of DNN-HMM ASR [76, 77]. The reason lies in RNN LM's ability to model any

⁴At time 0, the h_{-1} is usually randomly initialized or set to 0 vector.

sequence of words. Given the number of all possible word sequences, representing them in a list form as a mapping from the word sequences to the scores becomes impractical. Therefore, in DNN-HMM ASR, RNN LMs are usually employed at the later stages, e.g. to rescore generated word lattices or N-best hypotheses list. On the other hand, RNN LMs are implicitly modeled by E2E ASR systems. Additionally, in E2E ASR, they can also be used as an external LM both at the decoding and the rescoring stages [78–80]. The aforementioned utilities of RNN LM makes it indispensable part of ASR systems.

2.1.2.3 Language Model Evaluation

There are two effective ways to measure the performance of a LM depending on the task at hand and other constraints such as available computational resources and time.

Task-dependent. The best way to evaluate the performance of a LM is to deploy it in a target task and measure the performance of the task. This is the only way to certainly tell which LM better suits the considered task. For example, in the ASR task, to compare two LMs we have to run two recognition processes, one for each, and then measure the obtained WERs.

However, in some cases, the task-dependent evaluation might be computationally expensive. For instance, in the ASR task with the very large training and evaluation sets, we might have to wait from several hours up to several days to get the performance results. To quickly compare several LMs independently from the task, other performance metrics can be used.

Task-independent. The most commonly used task-independent evaluation metric for LM is perplexity (PPL) [81]. The intuitive explanation of perplexity is that it approximates the number of alternatives the LM believes there are following each word, i.e. average branching factor. Therefore, the LM with a lower perplexity is considered a better model. The perplexity is computed by taking the geometric mean of the inverse probabilities of the words in the evaluation set $E = (w_1 \ w_t \ \dots \ w_T)$, that is:

$$\text{PPL} = \sqrt[T]{\prod_{t=0}^{T-1} \frac{1}{P(w_{t+1}|h_t)}} \quad (2.9)$$

where h_t is a context history and $P(\cdot)$ is a probability function represented by a LM.

Another popular task-independent metric is cross-entropy (CE)⁵ which is simply a $\log_2$ of the perplexity, i.e.:

$$\text{CE} = \log_2 \text{PPL} = -\frac{1}{T} \sum_{t=0}^{T-1} P(w_{t+1}|h_t) \quad (2.10)$$

Note that the task-independent and the task-dependent evaluations are not always correlated. Klakow and Peters [82] have found that the task-independent evaluations such as PPL should be employed only when the training and the evaluation data belong to similar domains.

2.1.3 Domain Mismatch Problem

In this subsection, we will first introduce the definitions and the mathematical formulations of some important concepts required to define the domain mismatch problem, followed by the impact of the domain mismatch problem on AM, LM, and ASR tasks.

Domain. A domain $\mathcal{D}$ can be represented as a 2-tuple $(\mathcal{X}, P(X))$: a feature space $\mathcal{X}$ and a probability distribution $P(X)$, where $X = \{x_1, \dots, x_N\} \in \mathcal{X}$. For example, if our learning task is language modeling, then $\mathcal{X}$ is the space of all word sequences, X is a particular word sequence and x_i is the i^{th} word.

Domain mismatch. In general, if two domains $\mathcal{D}_S = (\mathcal{X}_S, P_S(X))$ and $\mathcal{D}_T = (\mathcal{X}_T, P_T(X))$ are different, i.e. $\mathcal{D}_S \neq \mathcal{D}_T$, then they may have different feature spaces $\mathcal{X}_S \neq \mathcal{X}_T$ or different word sequence probability distributions $P_S(X) \neq P_T(X)$ or both [83]. In this work, we will only focus on the domain mismatch caused by the probability distribution difference between the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$. In language modeling example, that would correspond to the word sequence distribution difference between train and test data. Note that the domain mismatch problem is applicable to both acoustic and language models as will be described in the following sections.

⁵In fact, it is an estimate of the cross-entropy, since a true distribution of words in natural language is unknown.

2.1.3.1 Acoustic Model Domain Mismatch

In the acoustic modelling task, the domain mismatch between train and test data is mostly caused by a variability in acoustic signal such as speakers characteristics (e.g. gender, age, mood, accent and dialect), environment noise (e.g. street, transport and nature), recording settings (e.g. microphone difference, close vs far talk, small vs large room) and speech rate. The AM domain mismatch problem has been exhaustively studied and many effective domain adaptation techniques have been proposed. For the detailed review of AM domain adaptation techniques, please refer to [84–91]. In this thesis, we will focus on the LM domain mismatch problem which is explained next.

2.1.3.2 Language Model Domain Mismatch

The statistical LM is a probability distribution over word sequences. As this distribution is learned from the textual training data, its shape will be formed according to the domain of the training data, e.g. topic and style [6]. For example, a training data from the medical domain will enforce LM to allocate more probability masses to the terms such as ‘bronchitis’ or ‘aorta’ rather than to the terms such as ‘grater’ or ‘mincer’ more commonly used in food-related domains. Thus, the characteristics of training data is a major factor that determines the effectiveness of a LM.

The LMs have been shown to be extremely sensitive to the differences, i.e. domain mismatch, between training and test data [6, 7]. The effect of domain mismatch may still be strong even when the differences seem insignificant to human. For instance, Rosenfeld [66] reports that the perplexity of a LM trained on Wall Street Journal newswire text is doubled when applied to the very similar Associated Press newswire text from the same time period. Moreover, the LM domain mismatch may also affect the performance of the downstream applications such as ASR.

2.1.3.3 Domain Mismatch Impact on ASR

The domain mismatch in ASR is mainly caused either by AM or LM or both. While the AM domain mismatch leads to the acoustic level errors, the LM domain mismatch is responsible for the linguistic level errors. Specifically, in ASR, LM

imposes a collection of constraints on the search space to govern that the selected paths (or hypotheses) conform with the regularities of a language. Inappropriate LM will confuse the ASR system during recognition of acoustically similar words (or phrases). For example, a LM trained on agriculture domain data, but applied to math domain might force decoder to choose a path passing through ‘COW FACTORIES’ instead of ‘CO-FACTOR IS’ which are phonetically similar.

To evaluate the impact of AM and LM on ASR performance, Lefevre et al. [92] conducted several experiments employing a combination of in-domain and out-of-domain LMs and AMs. The experiment results showed that around 90% of WER increase was due to the domain mismatched LM. In [93], similar trends were observed for different datasets and languages. These results suggest that the impact of LM domain mismatch is more devastating than the AM domain mismatch. Therefore, for reliable performance of ASR, it is crucial to ensure that the domain of a LM matches the target domain of the test data.

2.2 LM Adaptation Techniques for ASR

This section reviews the state-of-the-art LM adaptation techniques designed to alleviate the impact of domain mismatch problem caused by linguistic dissimilarities between training and test data. First, in subsection 2.2.1, we describe the general LM adaptation framework including its core modules such as background corpus, adaptation corpus, information extraction, LM estimation, and LM adaptation. Next, we describe popular LM adaptation techniques categorized into four main groups: information combination (subsection 2.2.2), model-based (subsection 2.2.3), feature-based (subsection 2.2.4) and hybrid (subsection 2.2.5). Some of these techniques share common methodologies and have subtle differences which will be pointed out in great details. As it will be shown later, these techniques are solely designed to adapt topic mismatches between training and testing sets, and thus applicable to Chapter 3 of this thesis. To address vocabulary and language mixing practice mismatches, we will present novel LM adaptation techniques in Chapter 4 and Chapter 5, respectively. Therefore, the similar works to those proposed LM adaptation techniques from other fields will be mentioned in the related works section of corresponding chapter.

2.2.1 General LM Adaptation Framework

The general LM adaptation framework is depicted in Figure 2.7 (adopted from [7]). This framework consists of five main modules: background corpus, adaptation corpus, information extraction, LM estimation, and LM adaptation. The purposes of these modules are described below.

FIGURE 2.7: General LM adaptation framework.

Background corpus. The background corpus is a collection of textual data used as a training data to estimate the parameters of initial LM, also known as background LM. Thus, the characteristics of the background corpus (i.e. size, topic, genre) will define the capability of the background LM, and subsequently, affect the performance of a downstream application.

The background corpus is usually constructed by collecting data from the domains similar to the target domain. However, in some cases, the target domain might be unknown beforehand, or it might cover several domains changing over time. For example, as in the application of ASR to the broadcast news or talk shows. In such cases, the common practice is to construct a generic background corpus by assembling data from various domains.

LM estimation. The LM estimation module is responsible for the construction of a background LM using the background corpus. In general, any of the language modeling techniques described in section 2.1.2 can be applied as long as it suits the requirements of the corresponding downstream application. For example, for the decoding stage of conventional DNN-HMM hybrid ASR, the count-based N-gram LMs are preferred. Whereas, implicit LM of E2E ASR is based on RNN framework.

Adaptation corpus. The adaptation corpus is a collection of textual data from the target domain. This corpus incorporates essential information about the target domain such as topic, genre, and style which can be used to adapt background LM. The adaptation corpus, however, is rarely available in prior. Therefore, it is usually collected manually by the domain experts which is time-consuming and expensive.

An alternative way to obtain adaptation corpus is to exploit the initial output of the system. For instance, in ASR, the generic background LM can be used to produce initial hypotheses which can be used as an adaptation corpus or to obtain tips about the target domain. This is the best adaptation data that we can acquire as generated hypotheses directly reflect the input test data that needs to be recognized. However, the initial hypotheses are erroneous and using them might hurt the performance of a LM. Nevertheless, they still might contain some useful domain-specific clues as has been shown in [94, 95].

Information extraction. The function of the information extraction module is to obtain target domain-specific information from the adaptation corpus. The extracted information can take the form of N-gram counts, bag-of-words, topic information, keywords and so on. A good example of an information extraction module is a topic models [96]. The extracted target domain information is then used to modify the background LM during the adaptation stage.

LM adaptation. Lastly, the LM adaptation module employs the extracted target domain information to adapt the background LM so that it better fits the target domain. Over the past few decades, a variety of LM adaptation techniques have been proposed. The adaptation techniques mostly depend on the available target domain information and employed LM type. We will thoroughly review these techniques in the following sections.

Adaptation modes. It is important to mention that in ASR the LM adaptation can be performed in two ways: online and offline. The online mode implies that the parameters of background LM are modified during the decoding stage. For example, the background LM is updated incrementally after each decoded word so that it exhibits better performance on the remaining material from the target domain. On the other hand, offline adaptation is performed at a later time. The offline adaptation is suitable for video streaming websites such as ‘YouTube’ where

the uploaded videos can be transcribed belatedly, e.g. when computational resource is available.

In addition, as we mentioned previously, the LMs can be used implicitly as an internal component of ASR system or explicitly as an external model, e.g. to rescore output hypotheses. Therefore, adaptation methods can also differ depending on the stage where a LM is used. The internal LM adaptation is considered more difficult than the external one, especially for the DNN-HMM hybrid ASR systems where the LM WFST is composed with other models. The following sections will describe various LM adaptation techniques.

2.2.2 Information Combination Based Adaptation

The idea behind information combination (IC) is to combine two or more information sources or probabilistic models into one. In language modeling, it can be applied for two different purposes: ensemble learning and adaptation. For ensemble learning, IC methods are used to combine several LM types, presumably trained on the same data and complementary to each other, to improve overall performance [97]. On the other hand, for adaptation, an in-domain LM is combined with the out-of-domain LMs to improve the performance on an in-domain task [7]. Here, we are interested in its application for LM adaptation purposes. Therefore, we first describe different IC methodologies followed by the popular IC-based adaptation techniques such as cache LM and topic-mixture LM.

2.2.2.1 Methodologies

In literature, several effective LM combination methodologies have been proposed such as linear interpolation [98], log-linear interpolation [99], back-off [100], fast marginal adaptation [101, 102] and maximum entropy [66]. Among them, the most popular ones are linear interpolation, log-linear interpolation and back-off methods which will be described next.

Linear interpolation. The goal of the linear interpolation based adaptation is to boost the probability of in-domain words (or phrases). This is achieved by first training an in-domain LM $P_I(\cdot)$ using the adaptation corpus and then linearly

interpolating it with a background LM $P_B(\cdot)$ as follows:

$$P_A(w_{t+1}|h_t) = \lambda P_B(w_{t+1}|h_t) + (1 - \lambda) P_I(w_{t+1}|h_t) \quad (2.11)$$

where $\lambda \in [0, 1]$ is an interpolation weight used to control the contribution of the models, and $P_A(\cdot)$ is an adapted model.

The justification for linear interpolation based adaptation is that the in-domain LM is trained on small adaptation corpus, and thus, its probability mass will be mostly distributed among in-domain words. While having good performance on the specific in-domain data, its performance on slightly different data will significantly drop. On the other hand, the background LM is too generic, leading to the suboptimal performance. To compensate each other's drawbacks, those two LMs can be combined using the linear interpolation. Subsequently, after the interpolation process, the posterior probability of in-domain words within adapted LM will increase, shifting its domain towards in-domain data.

Log-linear interpolation. The log-linear interpolation [99] maintains much of the simplicity of the standard linear interpolation. However, instead of combining LMs as in Eq. (2.11), the following scheme is used:

$$P_A(w_{t+1}|h_t) = \frac{P_B(w_{t+1}|h_t)^\lambda P_I(w_{t+1}|h_t)^{(1-\lambda)}}{Z} \quad (2.12)$$

where Z is a normalization constant used to ensure that $P_A(\cdot)$ is a true probability. Although log-linear interpolation method has been shown to perform slightly better than the standard linear interpolation, it is rarely used in practice.

Back-off. In back-off based model combination, we mainly employ the in-domain LM $P_I(\cdot)$ and turn (or back-off) to the estimates of background LM $P_B(\cdot)$ in cases when no sufficient evidence is observed in the adaptation corpus:

$$P_A(w_{t+1}|h_t) = \begin{cases} P_I(w_{t+1}|h_t), & C(w_{t+1}|h_t) > T \\ \alpha(h_t) P_B(w_{t+1}|h_t), & \text{otherwise} \end{cases} \quad (2.13)$$

where $\alpha(\cdot)$ is a back-off weight used to ensure that $P_A(\cdot)$ is a valid probability distribution, $C(\cdot)$ is a count of word w_{t+1} following the context h_t in the adaptation corpus and T is some threshold value.

The main difference between linear interpolation and back-off methods is that linear interpolation always combines estimates from both models, whereas the back-off method turns to background LM only when an estimate from the in-domain LM is unreliable.

2.2.2.2 Cache-based LM

The information combination methods serve as a basis for many popular LM adaptation techniques including the cache-based LM [9, 95, 103–105]. The cache-based adaptation techniques are based on the hypothesis that a word used in recent past is more likely to be used soon than background LM would predict. This can be better explained by the following example. Suppose the word sequence ‘he bought’ is given, and that the word ‘flower’ has followed this sequence 5% of the time in the training data, while the word ‘flour’ has followed 1% of the time. Thus, a standard 3-gram background LM would assign a higher probability of observing ‘flower’ comparing to the ‘flour’. Now suppose that several recognized sentences, preceding the word sequence ‘he bought’, contained the word ‘flour’, while none contained the word ‘flower’. In this case, assigning a higher probability to the word ‘flour’ seems to be more reasonable.

This can be achieved by boosting the probability of previously recognized words within background LM. In particular, the recognized words are first accumulated in the cache (or buffer). The buffer is then used to build cache LM $P_{cache}(\cdot)$ which is interpolated with the background LM $P_B(\cdot)$ as follows:

$$P_A(w_{t+1}|h_t) = \lambda P_B(w_{t+1}|h_t) + (1 - \lambda) P_{cache}(w_{t+1}|h_t) \quad (2.14)$$

To reduce the recognition latency, the adaptation can be performed at fixed periods, e.g. every 30 seconds. Note that the cache-based LM adaptation can be also implemented using the back-off method [5].

The concept of cache in language modeling has gained immense popularity in the 1990s. It was motivated by the fact that the cache contains the most relevant data that directly reflects the input test data. However, the WER improvement from cache-based adaptation was mediocre and inconsistent [106]. This is due to the ‘locked-in’ phenomenon, also referred to as error propagation, where increasing

the probabilities of misrecognized words imposes the ASR system to repeat the same mistakes. To mitigate the ‘locked-in’ phenomenon, Khassanov et al. [107] proposed to use cache data to select relevant sentences from the generic background corpus, and then to use selected data to update the background LM. Although the proposed method avoids the direct usage of cache data, it will increase the latency and introduce additional complexities such as reliable data selection process.

2.2.2.3 Topic-mixture LM

The topic-mixture LM is another popular adaptation approach based on information combination methods. This approach was first proposed for count-based N-gram LMs [10, 108–110] and later expanded to RNNLMs [111]. In topic-mixture LM, the background corpus is first split into several topic clusters (hard or soft clustering). These clusters are then used to build several topic LMs which are combined using linear interpolation:

$$P(w_{t+1}|h_t) = \sum_{j=1}^K \lambda_j P_j(w_{t+1}|h_t) \quad (2.15)$$

where K is a number of topics, λ_j is a weight assigned to topic LM $P_j(\cdot)$ such that $\sum_{j=1}^K \lambda_j = 1$. During the adaptation, the interpolation weights λ_j of topic LMs are tuned to match the topic of current input test data.

The topic-mixture LMs have two main challenges: data splitting and interpolation weight estimation. The data splitting is usually performed using a simple clustering method such as k-means algorithm. However, splitting data into smaller pieces introduces the data sparsity problem. To alleviate this problem, a ‘full’ LM $P_{full}(\cdot)$ trained on the entire data is recommended to be used as an additional mixture component:

$$P(w_{t+1}|h_t) = \sum_{j=1}^{K-1} \lambda_j P_j(w_{t+1}|h_t) + \lambda_K P_{full}(w_{t+1}|h_t) \quad (2.16)$$

To estimate the interpolation weights some target domain data is required, however, such data is usually not available in prior. Alternatively, it can be estimated using

the previously recognized words, i.e. cache data [95]. Different from the cache-based LM, in topic-mixture LM, the cache is used indirectly to determine the relevance of different topics. Ideally, the interpolation weights should be updated after every word (or utterance), however, this approach is impractical. Instead, they can be updated, for example, after every 10% of the test set.

2.2.2.4 The Pros and Cons

The information combination is widely used adaptation technique, especially for count-based N-gram LMs. The main drawback of these techniques is that they require some prior knowledge about the target domain. In addition, they directly modify the parameters of LM which might hurt linguistic regularities captured from the natural textual data. Nevertheless, they have been shown extremely effective and simple to implement. Moreover, the information combination is a core component for many other popular count-based N-gram LM adaptation techniques such as cache-based LMs and topic-mixture LMs.

2.2.3 Model-based Adaptation

The model-based adaptation techniques aim to embed a target domain-specific knowledge within a pre-trained LM by changing its architecture and parameters [12, 112, 113]. These techniques are only applicable to neural LMs where network architecture can be easily altered. For instance, the architecture can be changed by inserting an additional layer or scaling the hidden unit activations. It is important to mention that the model-based adaptation techniques were first employed in DNN acoustic model adaptation, and then applied for LMs [90, 114]. The model-based adaptation techniques have many varieties, therefore, we categorized them into the three general classes: fine-tuning [113], linear hidden network cascading [112] and linear hidden unit contributions [12]. These three classes are described in the following subsections.

2.2.3.1 Fine-tuning

The fine-tuning is frequently used and well-known practice in the artificial neural network (ANN) field. In fine-tuning based adaptation [11, 12, 113], the architecture of the pre-trained model is unchanged, only the model parameters are updated by further training (or post-training) it with an adaptation data. Commonly, as an adaptation data the recognized output hypotheses are used, similar to the cache-based LMs, which are shown to gain small but consistent improvements [12, 113]. Although the fine-tuning has been shown inferior to other model-based adaptation techniques [11], it is the simplest to implement.

2.2.3.2 Linear Hidden Network Cascading

The linear hidden network cascading method is adopted from the acoustic model adaptation [115, 116] where an additional network layer is inserted into a pre-trained model, see Figure 2.8b. In particular, a baseline LM is first trained using the available background corpus. Next, the pre-trained LM is modified by cascading additional network layer, also referred to as an adaptation layer. The adaptation layer is usually a linear network initialized with the identity matrix, and its goal is to learn latent linear transformation to the target domain space:

$$h'_t = W_a h_t + b_a \quad (2.17)$$

where h_t is a hidden state from the previous layer, h'_t is a modified hidden state, W_a and b_a are learnable matrix and bias parameters of the adaptation layer respectively. There are several locations where the adaptation layer can be inserted. For example, Park et al. [112] proposed to insert it between input projection and non-linear hidden layers. On the other hand, Deena et al. [11] inserted it between non-linear hidden and output projection layers. Lastly, the modified LM is fine-tuned using the adaptation corpus. During the adaptation, a common practice is to update only the parameters of the adaptation layer, while freezing the other parameters.

2.2.3.3 Linear Hidden Unit Contributions

The linear hidden unit contributions based adaptation method was first introduced in acoustic models [90], its application for LM adaptation is proposed in [12, 117]. The idea is to use domain-specific knowledge to re-weight (or scale) the hidden unit activations h_t of some layer in the network (see Figure 3.7d):

$$h'_t = h_t \odot r_t \quad (2.18)$$

where h_t is a hidden state from the previous layer, r_t is scaling vector, $\odot$ is element-wise product symbol and h'_t is a re-scaled hidden unit activations. The motivation is that hidden unit activations represent high-level feature vector where each value of the vector corresponds to some linguistic information. The magnitudes of these values determine the contributions of the corresponding linguistic information. Thus, by learning the contributions of linguistic information for each domain, we can tune them to match any interested domain. The contributions are usually learned using some domain knowledge such as topic or genre.

FIGURE 2.8: Architectural changes in model-based adaptation techniques: (a) vanilla RNNLM, (b) linear hidden unit cascading, (c) linear hidden unit contributions.

2.2.3.4 The Pros and Cons

One of the main drawbacks of model-based adaptation is the catastrophic forgetting phenomenon [116]. The catastrophic forgetting refers to a partial or total loss of the previously learned information caused by the adaptation process. Especially, this effect must be taken into account when adapting a model with a limited amount of data.

Another drawback is that the baseline model is required to be adapted to each domain separately. In tasks, where many domains are present such as in broadcast news, the adaptation process might be computationally expensive and time-consuming. Additionally, it will result in many replicas of the adapted baseline model which might require a lot of memory for storage.

The advantage of model-based adaptation is that no prior domain-specific knowledge is required. The pre-trained baseline LM can be adapted once the adaptation corpus is available. Overall, the model-based LM adaptation techniques have been proven effective and easy to implement.

2.2.4 Feature-based Adaptation

The feature-based LM adaptation techniques are similar to the acoustic model adaptation techniques based on i-vectors [89, 118, 119]. The idea is to exploit additional domain-specific knowledge such as topic and genre. For example, by augmenting the input of LM with auxiliary features that encode domain information. Different from the model-based techniques where a pre-trained LM is adapted, the feature-based techniques are carried out from the beginning of the training stage. The feature-based techniques are mainly known in neural LM adaptation field, however, we will show that they actually cover some of the count-based N-gram LM adaptation techniques as well. We split the feature-based techniques into two main subgroups: context dependent LM and factored LM.

2.2.4.1 Context dependent LM

The context dependent LM adaptation is only applicable to neural LMs [11, 13, 14] where an additional real-valued feature vector is provided as an input to the model⁶, i.e. $P(w_{t+1}|h_t, f_t)$ where f_t is a feature vector, see Figure 2.9. These feature vectors are used to represent the overall contextual information in association with each word. The context vectors are usually estimated using external topic models such as latent Dirichlet allocation [120] and latent semantic analysis [121]. Though, some attempts have been made to estimate them implicitly [117], e.g. by using sequence summary networks [122]. Alternatively, the context vector can be modeled as a

⁶The feature vector can be input to any layer of neural LM.

one-hot vector representation [11]. For example, each value in the one-hot vector may correspond to some genre, thus, vector size will be equal to the number of available genres. On the account of additional contextual information, the context dependent LMs achieve significant improvements.

FIGURE 2.9: The context dependent RNN LM architecture. The context feature vector f_t can be input into hidden or output or both layers.

2.2.4.2 Factored LM

The factored LM is a generalization of context dependent LMs which views words as consisting of K factors, i.e. $w_t \equiv \{f_t^1, \dots, f_t^K\}$ [56, 57]. The factors can be anything including word itself, part-of-speech, stem, class and so on. Generally, the factored models can learn $P(f_{t+1}^{1:K} | f_{1:t}^{1:K})$ which can be reduced to LM form $P(f_{t+1}^1 | f_{1:t}^{1:K})$ where f_{t+1}^1 is word itself. Moreover, the factored LMs can generalize the feature-based adaptation techniques to count-based N-gram LM [56]. In addition, the factored LM can generalize other popular LM types. For example, the 2-factor LM generalizes the class-based LM [58, 59] by assigning one factor to word and the second factor to its class. Similar to context dependent LMs, the factored LMs achieve significant improvements as a result of exploiting additional linguistic information.

2.2.4.3 The Pros and Cons

One of the main disadvantages of the feature-based adaptation is that it requires the auxiliary features to be known at the time of model training. Moreover, these

features must be known for the test data as well. Furthermore, if new feature types become available or if features are slightly altered, e.g. need to add a new genre, then the entire model needs to be re-trained from scratch. The re-training might take from several hours up to several days depending on the model and background corpus sizes. Besides, the effectiveness of these techniques highly depends on the quality of the features.

The main advantage of feature-based adaptation techniques is that the model doesn't need to be adapted to each domain separately, i.e. one model fits all. During the adaptation, no further post-processing is required, only the input features are required to be changed to switch the domain of a LM.

2.2.5 Hybrid Adaptation

The hybrid techniques simply involve a combination of two or more adaptation techniques described above. The main motivation to combine several techniques is to exploit complementary knowledge and to cover each others' limitations. In literature, many different hybrid approaches have been proposed, we will briefly describe the popular ones. The detailed review of combinations of different LM adaptation techniques can be found in [5–8].

Linear hidden network cascading & Context dependent LM. Deena et al. [11] proposed to alleviate the effect of catastrophic forgetting in model-based techniques (i.e. linear hidden network cascading) by leveraging feature-based approach (i.e. context dependent LM). Similar approaches with slight differences have been also explored in [123, 124].

Linear hidden network contributions & Context dependent LM. Hentschel et al. [117] also combined feature-based and model-based techniques. In particular, the implicitly learned context feature vectors were used as a scaling vector to tune the linear hidden unit contributions.

Fine-tuning & Context dependent LM. The fine-tuning of context dependent LM has been explored in [125]. The motivation is to exploit complementary domain information from both context feature and in-domain text.

2.3 Summary

In this chapter, we briefly described the architecture of two state-of-the-art ASR systems, namely hybrid DNN-HMM and E2E, including their core components such as AM, PM, and LM. These components interact to transform given speech signal into corresponding word sequences. Importantly, in hybrid ASR, these components are explicitly modeled using different objectives, while E2E ASR implicitly models them jointly under a single deep neural network.

We also described general LM formulation and reviewed the state-of-the-art language modeling approaches such as count-based N-gram and RNN LMs. The LM aims to incorporate as much as possible linguistic knowledge which is used to assess the correctness of given word sequences. They are especially useful in applications where natural text is generated, e.g. speech recognition, machine translation, question answering, information retrieval, image captioning and many others. For speech recognition task, the count-based N-gram LMs are mainly used in hybrid ASR, whereas the RNN LM is an internal component of the E2E ASR. The RNN LMs can be also used externally on both ASR architectures to rescore the word lattice or N-best output hypotheses list.

However, LMs are highly vulnerable to the linguistic differences (e.g. topic, genre, and style) between training and test data which is known as domain mismatch problem. Therefore, we have discussed the causes and effects of the domain mismatch problem on both AM and LM, and its impact on the recognition performance of ASR.

Lastly, we described the general LM adaptation framework developed to deal with the domain mismatch problem and reviewed the state-of-the-art LM adaptation techniques for both count-based N-gram and neural LMs. We split the LM adaptation techniques into four groups including information combination, model-based, feature-based and hybrid approaches. These techniques mainly aim to modify the parameters and architecture of LM to better match the target domain. We observed that these techniques mainly address the topic or genre level domain mismatch. Additionally, these techniques either require seed in-domain data or rely on some prior domain-specific knowledge. Importantly, these techniques are designed to operate in monolingual space and fall behind the recent trend of mixing several languages within a single discourse. To address the aforementioned drawbacks of

conventional LM adaptation techniques, in the following chapters we will propose novel domain adaptation techniques which don't require any in-domain data or prior domain knowledge.

Chapter 3

Language Model Adaptation by Data Selection¹

In this chapter, we propose a novel language model adaptation framework for conventional DNN-HMM ASR systems. The proposed framework is based on data selection techniques that aim to select important and relevant training samples for the underlying task. First, in Section 3.1, we describe the data selection task including the general framework, its applications and the state-of-the-art data selection techniques. Next, in Section 3.2, we present our proposed data selection based adaptation framework. The proposed framework is a two-pass system designed to adapt the count-based N-gram LM residing within DNN-HMM ASR. In Section 3.3, we describe the experiments conducted to evaluate the proposed framework, and discuss obtained results and important observations. Lastly, Section 3.4 summarizes this chapter.

¹The work in this chapter has been published in [107].

3.1 Introduction

In natural language processing (NLP) system design, the conventional wisdom is that “there is no data like more data” [126, 127], i.e. it was common belief that more data results in better statistical models. The truth, however, is that the performance of the models not only depends on the quantity of the data, but also on the quality. Where quality here means how relevant the data is to the target domain. Thus, the focus is now shifting towards efficient ways to select high-quality training data for statistical models as it has been shown to improve the performance of NLP systems significantly.

Inspired by the success of the application of data selection techniques in other related fields, in this work, we explore its application for LM adaptation in DNN-HMM based hybrid ASR systems. In ASR, it is common practice to employ a LM trained on large textual data collected from different domains in order to increase the domain coverage of the system. However, this leads to suboptimal performance of the generic LM when a test data from a specific target domain is input to the system [6]. To mitigate this problem, the generic LM needs to be adapted to match the statistical properties of the input test data [7].

In the past few decades, many effective LM adaptation techniques have been proposed such as cache LM [9] and topic-mixture LM [10], to mention few. Most of them resort to two-pass adaptation approach where a recognized output from the first-pass is used to adapt a LM which is then employed in the second-pass decoding. Although the adapted LM will be biased in favor of the words or phrases from the target domain, it will have two important drawbacks. The first drawback is an error propagation caused by the direct usage of a recognized output which is subject to recognition errors. In other words, the adapted LM will be encouraged to repeat the same mistakes. The second drawback is a distortion of linguistic regularities encapsulated within LM caused by selectively modifying the parameters of a LM. The problem is that the parameters of a LM are learned from a “natural” text data produced by a human, while directly tweaking those parameters will ruin the linguistic knowledge.

Keeping the aforementioned drawbacks in mind, in this work, we propose data selection based LM adaptation framework for ASR systems. Specifically, instead of modifying the parameters of LM, we propose to modify the training data by

filtering out the irrelevant sentences. Our proposed adaptation method is also based on the two-pass approach, however, we use the recognized output from the first-pass indirectly, i.e. to select in-domain data from the linguistically correct sentences. Consequently, we avoid the usage of semantically and syntactically incorrect data to adapt a LM. Additionally, the linguistic regularities encapsulated within adapted LM will be adequate as it was trained on “natural” text. The proposed method will be thoroughly explained in the following sections.

First, in this section, we will briefly describe the general data selection framework and its applications, followed by the overview of the state-of-the-art data selection techniques including entropy scores, vector space models and others.

3.1.1 Data Selection Framework

The data selection techniques aim to filter a large corpus of data to a ‘good’ data suitable for a particular task [128]. This is usually achieved by employing small set of in-domain data representative of the task, also known as a *seed* set, to select similar data from a much larger corpus. The selected data is then used to train a task specific model which usually performs better than a model trained on the entire corpus. The Fig. 3.1 depicts the main modules of the data selection framework which are described next.

FIGURE 3.1: Data selection framework.

In-domain data. An in-domain data (or seed set) is a collection of data that describes the target task. It incorporates important information about the task such as topic, genre and style. This data is usually limited in size and not sufficient to train a reliable model.

Large corpus. A large corpus is usually a collection of all available data covering various domains. Multiple domains are included to ensure that a data similar to the in-domain data can be found. Using the large corpus to train a model usually results in suboptimal performance.

Data selection. The data selection module is responsible for generation of high quality data matching the in-domain data. This is achieved by employing information present in the in-domain set to select relevant data from the large corpus. As will be discussed in the following sections, the employed information and selection criteria varies across different data selection techniques.

3.1.2 Applications of Data Selection

In this section, we review some NLP applications and use cases where the data selection techniques have been employed.

Translation Model. A translation model $P(s|t)$ is an important component of conventional statistical machine translation (SMT) systems which together with a language model defines the search space of all possible translations. In particular, it models the probability that a source sentence s is the translation of a target sentence t . This probability is learned from the parallel corpus where each sentence is placed alongside its translation.

The translation models (TM), however, are sensitive to the choice of parallel corpus used as a training data. For instance, consider the word ‘BANK’ which has different meanings depending on the topic and context it is used at, e.g. financial institution or geographical entity as in ‘river bank’. In another language, most probably it has different translations for both meanings. As such, using TM trained on one specific domain data to translate a test data from different domain might severely degrade the translation performance of a SMT system. Even if we train a TM on the generic parallel corpus, the performance of the SMT will be suboptimal. In that case, the translation of ‘BANK’ in the sense of financial institution most probably would overwhelm the other translations as it is more frequently used.

To deal with such problems, bilingual data selection techniques are employed to select relevant sentences from the parallel corpus [128–131]. The selected data

results in better TMs that significantly improve the translation performance of the SMT systems.

Named-entity recognition and classification model. The named-entity recognition and classification (NERC) is a subtask of information retrieval which aims to detect and classify named entities in the test set into pre-defined categories such as locations, organizations, person names and so on. Nowadays, NERC systems mostly resort to supervised machine learning techniques to build efficient and accurate models [132]. However, these techniques require large amount of labeled training data which is time consuming and expensive to produce.

To deal with the deficiency of the labeled data, the semi-supervised learning approaches are commonly used [132]. In this approach, unlabeled sentences from the large text corpus are first automatically labeled by the baseline NERC system, and then high-confidence machine-labeled sentences are used as an additional training data. However, in [133], authors observed that using irrelevant machine-labeled sentences to train the model might degrade the performance of the NERC system. Thus, they proposed to employ data selection techniques as preprocessing step to discard irrelevant sentences from the corpus. As a result, the robustness of semi-supervised approaches significantly increased leading to consistent improvement of NERC systems

Acoustic Model. An acoustic model (AM) is an important component of conventional DNN-HMM ASR systems which models the relationship between acoustic features and phonemes. In order to train an AM, a speech corpus consisting of audio files with corresponding transcripts is required. However, speech corpora are highly expensive and time consuming to generate. Consequently, only limited number of good speech corpora are available that covers some of the popular domains. Moreover, AM training is computationally demanding and an extremely lengthy process. To this end, with respect to AMs, data selection techniques are mostly employed in two scenarios [134]:

- 1) To reduce the budget for building speech corpus, e.g. for new languages or domains [135–138].
- 2) To minimize computational resource required to train AM [126, 134, 139].

In the first scenario, data selection techniques are used to identify most informative subset of the untranscribed audio files for human annotation. The annotated data is then used as the AM training data. In addition, the size of the selected subset should fit the available budget as human annotation is not free. For example, Hakkani-Tür et al. [135] proposed to use available in-domain set of transcribed data to build a baseline ASR system. The baseline system is then used to automatically transcribe available audio files. They relied on the confidence score to identify the most informative subset of the audio files. In other words, utterances recognized with the lowest confidence scores are the best candidates to be manually annotated as the ASR system is ineffective at recognizing them. These manually annotated data are then used to retrain the AM. As a result, the capability of the ASR system to handle utterances with the similar acoustic properties will be improved. In their experiment, the proposed method reduced the amount of data to be transcribed by 27% while keeping the performance of AM unchanged.

In the second scenario, data selection techniques are employed to select the most representative subset of the speech corpus to train an AM. The motivation is to reduce the requirements for the computational resources. For example, Wu et al. [126] used in-domain transcripts to select training data from the large speech corpus (840 hours). In particular, they compared in-domain transcripts against each transcript present in the speech corpus. The audio files whose transcripts were identified as the most similar were selected to train AM. In their experiment, the proposed method selected 17% (150 hours) of the speech corpus to achieve the same WER as using the entire speech corpus.

Language Model. In Section 2.1.2, we have described LM structure and the state-of-the-art LM types. The LM is a probability distribution over words and is used to quantify ‘correctness’ of word sequences. Such models are essential for NLP applications that generate sequence of words such as speech recognition, machine translation, spelling correction, handwriting recognition, image captioning and so on. For this reason, in the past few decades, LMs have attracted substantial attention and many text corpora became available.

The LMs, however, like many other statistical models are highly sensitive to the choice of training data [140]. Even slight difference between training and test data might severely degrade the performance of the overlying NLP application. Thus, applying LMs to different domain data is troublesome. Fortunately, the

probability distribution of the LM is learned from the ordinary text corpus which is in abundance, e.g. books, new articles, web and so on. Consequently, many effective data selection techniques have been proposed to enable the collection of relevant LM training data. In the following sections, we will provide comprehensive review of LM training data selection techniques.

3.1.3 Overview of Data Selection Techniques

In this section, we review popular LM training data selection techniques. As was explained in the previous section, data selection techniques employ in-domain set to select similar data from the large corpus. Specifically, data selection techniques extract features (e.g. N-grams, bag-of-words) from the in-domain set and compare it against the similar features extracted from the text segments (e.g. sentence, paragraph, document) in the large corpus. These features are usually compared using some similarity measures such as cosine similarity, cross-entropy, Euclidean distance and others. The text segments with the best similarity scores are then used to train a new task-specific LM.

Two main components of data selection techniques are (1) extracted features used to represent a text segment and (2) evaluation measures used to quantify similarity. We categorized widely used data selection techniques based on these components into three groups as follows: entropy based, vector space model based and others.

3.1.3.1 Entropy Based

Under the entropy based category we placed several related techniques including the cross-entropy, cross-entropy difference and relative-entropy. In general, these techniques aim to select training data that yields a task-specific LM with the lowest cross-entropy value estimated on the in-domain set. The entropy based techniques are considered the state-of-the-art for LM training data selection task.

Cross-entropy. The cross-entropy² is a measure used to evaluate fitness of a LM to a test set. It is computed based on the probability assigned to the test set by the LM. The high probability assignment implies that the LM is a good representative

²Note that cross-entropy is a log 2 of the perplexity measure. Therefore, this technique is sometimes called perplexity based data selection.

of the test set’s domain. Since the LM is learned from the training set, that also implies that the training and test sets have similar statistical properties. Thus, cross-entropy measure can be employed to evaluate similarity between training and test sets.

Likewise, we can employ cross-entropy measure to compare in-domain set against each text segment in the large corpus as follows [141, 142]:

- 1) A language model is first trained on the in-domain set.
- 2) The obtained in-domain LM is then used to assign cross-entropy score to each text segment in the large corpus.
- 3) The text segments with the best cross-entropy scores are selected as a ‘pseudo’ in-domain data. A number of text segments to select can be determined based on some held-out data.
- 4) Lastly, selected ‘pseudo’ in-domain data are used to train task specific LM which is employed in target NLP application.

The cross-entropy data selection technique has been shown to select high quality data. This data produces a LM that significantly improves the performance of various NLP applications.

Cross-entropy difference. Moore and Lewis [143] further modified the cross-entropy data selection technique. Particularly, they proposed to select text segments similar to the in-domain set and at the same time dissimilar to the out-of-domain (or general-domain) set. To identify such text segments, they proposed to use cross-entropy difference (CED) score which is a difference between cross-entropy scores assigned by the in-domain LM (CE_{ID}) and out-of-domain LM (CE_{OD}), that is:

$$CED(s) = CE_{ID}(s) - CE_{OD}(s) \quad (3.1)$$

where s is a text segment from the large corpus. Since defining and labeling the out-of-domain data is difficult, a randomly selected portion of the corpus (or entire large corpus) is used to train out-of-domain LM.

The CED based data selection procedure is performed as follows:

- 1) First, in-domain LM is trained on the in-domain set and out-of-domain LM is trained on randomly selected portion of the large corpus.
- 2) Obtained in-domain and out-of-domain LMs are then used to compute CED score for each text segment in the large corpus (using Eq. (3.1)).
- 3) Next, the text segments with the best CED scores are selected as a ‘pseudo’ in-domain data. A number of text segments to select can be determined based on some held-out data.
- 4) Lastly, selected ‘pseudo’ in-domain data are used to train a task-specific LM which is employed in target NLP application.

The CED based approach was shown to outperform the standard cross-entropy data selection technique [143].

The task-specific LM, produced by CED data selection technique, has been successfully applied in various NLP applications and became one of the most widely used techniques. Therefore, many other techniques have been proposed to improve it further. For example, in [144], authors improved procedure for drawing the out-of-domain data. In [128], authors observed that rare words have negative effect on CED score calculation, therefore, they proposed to replace rare words with their corresponding part-of-speech tags. The effects of various data pre-processing steps such as lemmatization and replacing words with corresponding named-entity tags was examined in [145]. Similarly, the effect of data pre-processing steps for other languages such as Spanish, Czech and Chinese are studied in [146]. In another work [131], authors proposed to use neural networks based LMs to compute cross-entropy scores. They hypothesize that the continuous vector representation of words in the neural network LMs are effective at approximating unknown word contexts, which are prevalent in the general-domain corpus.

Relative-entropy. The cross-entropy and cross-entropy difference data selection techniques are based on the so called rank-and-select approach, where scored text segments are first ranked and then only top-ranked text segments are selected. These techniques might end up with choosing identical and similar sentences. As such, the word distribution in the selected data will be biased towards high probability region of the in-domain LM which mostly consists of function words. In other words, CE and CED data selection techniques will favor text segments consisting

of function words, especially CE. Whereas the text segments consisting of words from the low probability region of the in-domain LM, which may contain important domain-specific words, are more likely to be discarded. Therefore, Sethy et al. [147] proposed data selection method which measures overall word (or N-gram) distribution similarity between in-domain set and selected data. The effective method for measuring distribution similarity is a relative-entropy (RE) measure, that is:

$$\text{RE} = - \sum_{w \in V} P_{\text{ID}}(w) \log_2 \frac{P_{\text{ID}}(w)}{P_{\text{SEL}}(w)} \quad (3.2)$$

where w is an arbitrary word, V is a vocabulary, $P_{\text{ID}}(w)$ and $P_{\text{SEL}}(w)$ are probabilities assigned by LM trained on the in-domain set and LM trained on the selected data, respectively.

The main idea of this data selection approach is to iteratively select text segments whose addition to the already selected set results in relative-entropy reduction. This technique was shown to outperform the standard cross-entropy data selection technique [147], however, it is computationally expensive as we need to re-estimate $P_{\text{SEL}}()$ every time we add a new text segment.

3.1.3.2 Vector Space Model Based

In this category, we group the data selection techniques that represent the text segments in a vector form [148]. The similarity between the text segments is then computed using the corresponding vector representations. These techniques are widely used in the information retrieval field where given a query and a list of documents, and the goal is to retrieve documents relevant to the query. Likewise, in vector space model (VSM) based data selection techniques [149–152], an in-domain set is used as a query to retrieve relevant text segments from the large corpus.

The VSM based data selection procedure is performed as follows:

- 1) The in-domain set and text segments in the large corpus are first transformed into vector form representations.
- 2) The in-domain vector is then compared against each text segment vector using some similarity metric.

- 3) The text segments with the highest similarity scores are selected as a ‘pseudo’ in-domain data. A number of text segments to select can be determined on some held-out data.
- 4) Lastly, selected ‘pseudo’ in-domain data are used to train task-specific LM which is employed in target NLP application.

Interestingly, the VSM based data selection techniques don’t aim to improve the performance metrics used in LMs, e.g. perplexity or cross-entropy scores. Nevertheless, they were shown to produce effective LMs that improve the performance of overlying NLP applications.

Next, we will briefly describe the frequently used vector representation and similarity metrics in VSM based data selection.

Vector representations. The popular vector representations used in VSM based data selection techniques are term frequency (TF), term frequency-inverse document frequency (TFIDF), latent semantic indexing (LSI) [121] and latent Dirichlet allocation (LDA) [120]. These representations typically capture different characteristics of the textual data. For example, TF is a vector of vocabulary size where each entry corresponds to the occurrence frequency of a term (e.g. word) in the text segment. However, in the text segments, the function words usually occur more frequently, as such, they will dominate the content of the TF vector and influence the selection procedure. Thus, in order to emphasize keywords, we can scale TF values of each word by corresponding inverse document frequency (IDF) to obtain TFIDF vector form.

Thus, LSI [121] technique can be used to project these vectors to the semantic space where semantically similar text segments tend to be clustered together. As a result, we can measure semantic similarity between text segments even if they don’t share any common words. On the other hand, LDA [120] representation assumes that each text segment covers several latent topics, hence, each text segment is represented as a distribution of these latent topics. The text segments covering identical latent topics are considered similar. A comprehensive explanation of these and other vector representations can be found in [120, 121, 153]. Please note that in our work we didn’t evaluate data selection techniques based on semantic similarity such as LSI and LDA which were left for future work.

Similarity measures. The most popular similarity measure used to compare two vectors is cosine similarity which typically measures the cosine of the angle between two vectors. Given a candidate vector $\vec{c}$ and target vector $\vec{t}$, the cosine similarity is computed as follows:

$$\cos(\vec{t}, \vec{c}) = \frac{\vec{t} \cdot \vec{c}}{\|\vec{t}\| \|\vec{c}\|} \quad (3.3)$$

where ‘ $\cdot$ ’ is a dot product operation, $\|\vec{t}\|$ and $\|\vec{c}\|$ are the lengths (Euclidean norm) of vectors $\vec{t}$ and $\vec{c}$, respectively.

Other popular similarity measures used to compare vectors include Euclidean distance, Jaccard coefficient and Pearson correlation coefficient. The review on these and other similarity measures can be found in [154].

3.1.3.3 Others

In this category, we discuss the remaining data selection techniques that did not fit the previous two categories.

BLEU. The bilingual evaluation understudy (BLEU) score [155] based data selection was proposed in [156]. The BLEU is the machine translation evaluation metric where a translated candidate sentence is compared to a set of reference sentences. It is a fully automatic evaluation metric that forms a viable alternative to expensive and time consuming human judgment of translation quality. In this approach, a sentence in the in-domain set is treated as a candidate sentence and all sentences in the corpus that share with it at least one keyword are considered possible references. The task is to find a list of reference sentences with BLEU score greater than a predefined threshold. This procedure is then repeated for the rest of the sentences in the in-domain set. Authors report that 20% relative WER reduction is achieved by using this technique to select LM training data for spoken dialog system.

Edit distance. In [130], edit distance based data selection technique was proposed. The edit distance is an effective metric widely used in spelling correction and in quantifying similarity of DNA sequences. It is defined as a minimum number of operations (e.g. insertion, deletion and substitution) required to transform one string into the other. For example, in data selection, it can be used to compute minimum number of operations, performed at word or character level, required

to transform one sentence to another. In this approach, for each sentence in the in-domain set, the sentences with minimum edit distances are selected from the corpus. However, the edit distance imposes very strict selection criteria which favors sentences with exact match of word overlap, order and position. Therefore, we should ensure that our corpus is large enough so that it contains highly similar sentences to the in-domain sentences. Authors claim that for the machine translation system, the edit distance based data selection produces a LM that outperforms LMs obtained by VSM (TFIDF+cosine similarity) and CED based techniques.

Log-likelihood. Klakow [157] proposed data selection method based on log-likelihood selection criteria. His method starts with a small set of in-domain data and a large corpus, then he iteratively removes ‘bad’ text segments from the corpus. To decide which text segments to remove, he employed log-likelihood change ΔF_i , computed as follows:

$$\Delta F_i = \sum_{w \in V} N_{ID}(w) \log \frac{P(w)}{P_{s_i}(w)} \quad (3.4)$$

where V is a vocabulary, $N_{ID}(w)$ is a count of a word w in the in-domain set, $P(w)$ and $P_{s_i}(w)$ are probabilities assigned by the LM trained on the entire corpus and LM trained on entire corpus after removal of the text segment s_i with index i , respectively.

The selection process can be completed in one iteration where the top k (predefined constant) text segments whose removal resulted on largest log-likelihood change are discarded. Alternatively, selection process can be performed in several iterations where we rebuild a new LM on the remaining text segments and compute the log-likelihood changes again. This process is repeated until desired number of text segments are left. However, this technique is based on brute force approach where different combinations of text segments are evaluated. Moreover, we need to retrain LM every time we remove a text segment. Therefore, it might be time intensive given a very large collection or finer level data selection is required, e.g. sentence selection. Furthermore, this technique was shown inferior to CED data selection techniques [143].

Text classifiers. It is also possible to train a binary text classifier to select relevant data. To train such classifier, however, we need annotated data, i.e. positive and negative samples, which are rarely available. Alternative option is to train a

binary text classifier from positive and unlabeled text segments [158–160]. These techniques are usually based on two-step strategy [161, 162]. In the first step, reliable negative text segments from the unlabeled corpus are identified. For example, by employing VSM based data selection techniques, but instead of the most similar text segments to the in-domain set, the most dissimilar text segments are selected. In the second step, the positive samples from the available in-domain set and identified negative samples are used to train a classifier.

3.2 Two-pass LM Adaptation Framework based on Data Selection

In this section, we present proposed two-pass adaptation framework based on data selection. The proposed LM adaptation framework is depicted in Fig. 3.2. The adaptation process is carried out in three stages: 1) first-pass recognition, 2) LM adaptation, and 3) second-pass recognition, each stage will be thoroughly described below.

FIGURE 3.2: Proposed two-pass LM adaptation framework based on data selection.

First-pass recognition. During the first-pass recognition, the input utterances from the target domain are recognized using baseline ASR system with the background LM. As prior knowledge about the target domain is unavailable, the background LM is typically trained on generic background corpus which consists of

text assembled from various domains such as broadcast news, sports, entertainment, science and so on. The motivation is to increase robustness of the baseline ASR to input test utterances from any domain.

Although the generic ASR will have good domain coverage, thanks to generic background LM, its performance will be sub-optimal on domain-specific utterances. One of the main reasons are commonly used words which are domain agnostic. In generic domain systems, these words might overtake infrequent domain-specific words. For example, a word from music and culture related domain such as ‘Tchaikovsky’ might be recognized as an acoustically similar and more frequent word sequence ‘check off ski’. Given that recognition of domain-specific words is crucial, especially for underlying spoken language understanding systems, it is important to adapt the domain of ASR system. The most straightforward and effective approach to adapt ASR’s domain is to bridge the gap between training and test data of LM.

LM adaptation. In this stage, we adapt the background LM to match the target domain of incoming test utterances. Since the target domain is unknown, we propose to extract corresponding information from the transcripts (or word lattice and N-best hypotheses list) produced during the first-pass recognition. However, the recognition errors present in the first-pass transcripts might corrupt the target domain information. Nevertheless, Clarkson and Robinson [95] observed that even highly erroneous text segments possess useful domain information and can still benefit the adaptation process. Specifically, they manually corrupt a text segment by randomly replacing words. The corrupted text segment was used to compute interpolation weights for topic-mixture LM. They found that the topic-mixture LM performs better than a baseline LM as long as corruption rate is lower than 50%. In another work, Lecorvé et al. [94] used only incorrectly recognized parts of the ASR to crawl additional text from the external sources such as web pages. The crawled text was used to train adapted LM which is interpolated with a baseline LM to generate the final LM. Surprisingly, the final LM obtained more than 10% relative perplexity improvement. They concluded that despite the presence of errors, the ASR output contains valuable information which can be effectively utilized by adaptation techniques. However, unlike existing two-pass adaptation approaches that directly apply the recognized output to modify the parameters of LM, we instead propose to adapt the training data using the data selection techniques. Specifically, the derived target domain information is used to select

relevant data from the huge background corpus consisting of text from various domains. For example, the selection procedure can be performed by employing one of the data selection techniques described in Section 3.1.3. By doing so we achieve two important benefits: 1) we avoid using erroneous ASR outputs which might contain grammatical mistakes to build LM, and thus, prevent the error propagation and 2) we substantially increase the potential in-domain training data size, since we can select any amount of data from the enormous background corpus.

In our experiments, we evaluated two state-of-the-art data selection techniques: VSM and CED, thoroughly described in subsections 3.1.3.2 and 3.1.3.1 respectively. The VSM techniques are frequently used in information retrieval tasks where they have been shown highly effective. Similarly, the CED based techniques have been also shown effective in machine translation and named entity recognition tasks. Specifically, these techniques were used to select in-domain sentences from the background corpus by comparing them against ASR output. As will be described in the experiment section, we tried to select different portion of in-domain sentences to identify the optimal quantity.

Lastly, selected in-domain sentences are used to train new domain-specific LM. The new LM is expected to better fit the target domain as it was trained on the estimated target domain data. In addition, the new LM can be considered as an adapted version of the original background LM as it was trained on the same, but pruned background corpus. Importantly, the adapted LM will capture essential linguistic regularities present in the text as complete sentences were retained during the data selection process.

Conceptually, our proposed adaptation method is completely different from the existing techniques which perform adaptation at the model level which aim to manipulate parameters of the model. Instead, our proposed method performs adaptation at the data level, i.e. it aims to manipulate the training data. Note that our proposed adaptation approach doesn't rely on any prior target domain data or knowledge. All necessary information is extracted from the ASR output.

Second-pass recognition. During the second-pass recognition, the adapted LM replaces the background LM in the ASR system to redecode the input utterances. Since the sentences 'similar' to the ASR output were used to train the adapted LM, we expect this LM to be representative of the target domain, and hence help to

improve the performance of the ASR system. The decoded output of the second-pass are used as a final transcripts.

3.3 Experiment

This section discusses the evaluation of the proposed two-pass LM adaptation framework, including the experimental data, the ASR system configurations, and obtained results.

3.3.1 Datasets

Speech Corpus. As the speech corpus, we employed TED-LIUM corpus [163] which consists of monologue talks given during TED (Technology, Entertainment, Design) conferences. In TED talks, experts from different fields give presentations on specific topics, e.g. green technology, space exploration and so on. The demographics of the corpus are shown in Table 3.1. The train set was used to train the acoustic model of the ASR system, while the test set was used for evaluation. The detailed information about speakers and topics covered in different talks of the test set is given in Table 3.2.

Characteristics	Train	Test
Total duration	118hr 4min	2hr 49min
Number of talks	774	11
Number of words	1.6M	27.5k

TABLE 3.1: TED-LIUM corpus characteristics.

Text Corpus. The text corpus was adopted from ‘1 Billion Word Language Model Benchmark’ project [164]. It consists of about 30 million sentences with a total of 700 million words after normalization. The corpus consists of text collected from different sources, including newswire text, news commentary text, and parliament talks. This corpus will be used as a background corpus to train the generic background LM, and we will select in-domain data from this corpus.

Pronunciation Dictionary. We used the pronunciation dictionary provided by Kaldi’s [35] TED-LIUM recipe. The dictionary contains around 180k words.

No.	Speaker	Topic name
1	Aimee Mullins	The opportunity of adversity
2	Bill Gates	Innovating to zero
3	Dan Barber	How I fell in love with a fish
4	Daniel Kahneman	The riddle of experience vs. memory
5	Eric Mead	The magic of the placebo
6	Gary Flake	Is Pivot a turning point for web exploration
7	James Cameron	Before Avatar ... a curious boy
8	Jane McGonigal	Gaming can make a better world
9	Michael Specter	The danger of science denial
10	Robert Gupta	Music is medicine, music is sanity
11	Tom Wujec	Build a tower, build a team

TABLE 3.2: TED-LIUM corpus test set details.

3.3.2 ASR and LM Configurations

The ASR system was built using the Kaldi speech recognition toolkit [35]. We followed the TED-LIUM’s recipe for the training and the testing procedures.

Acoustic model. The AM was built as the context-dependent deep neural network hidden Markov model (DNN-HMM). The DNN of the AM was pre-trained with restricted Boltzmann machine (RBM), and then trained with cross entropy training, and sequence discriminative method by using the state-level minimum Bayes risk (sMBR) criterion. The mel-frequency cepstral coefficient (MFCC) features were transformed by using linear discriminant analysis, maximum likelihood linear transformation (MLLT), and then feature space maximum likelihood linear regression (fMLLR).

Language model. The LMs were trained using the SRILM toolkit [165]. The generic background LM of the ASR system, used in the first-pass recognition, was trained as a trigram model and smoothed using the modified Kneser-Ney (KN) method [69]. Similarly, the adapted LMs, used in the second-pass recognition, were also modified KN smoothed trigram models. We used words in the pronouncing dictionary, which contains around 180k words, to specify vocabulary for LMs. Since our goal is to adapt domain of the ASR system by manipulating internal LM used during the decoding stage, we didn’t experiment with word lattice or N-best hypotheses rescoring.

3.3.3 Experiment Setup and Results

This experiment was conducted to evaluate if the proposed idea of data selection within the background corpus can improve ASR system performance. We considered the utterances in the 11 talks of the test set to be the same domain, e.g. TED talk domain. Although this is strictly not true, they are considered as same domain due to the nature of their talks. Thus, we first accumulated all the generated transcripts of 11 talks from the first-pass and used them jointly as the in-domain set for data selection. Data was selected using the VSM based and entropy based techniques. In VSM approach, we employed TFIDF vector representation with the cosine similarity metric. In entropy based, we employed the CED score. As the CED score is computed based on the LMs, we evaluated 3 types of LMs: 1-gram, 2-gram and 3-gram. We also examined how performance is affected by different amounts of data selected, i.e. from 0.1% until 100% of the background corpus.

We evaluated both WER and perplexity of the adapted LM. The WER in the evaluation transcripts after the second-pass recognition are shown in Fig. 3.3. The perplexities of the adapted LMs computed on the reference transcripts are illustrated in Fig. 3.4. Note that 100% corresponds to the usage of entire background corpus which will be identical to the result obtained from the first-pass recognition.

FIGURE 3.3: WER results obtained by the proposed LM adaptation framework.

Both WER and perplexity results show that a judicious choice of adaptation text improves the performance. As shown in Fig. 3.3, for all feature types considered, up to 1%-1.4% absolute WER reductions which are equivalent to 7%-10% relative

FIGURE 3.4: Perplexity results of adapted LMs computed on reference data.

WER reductions, were obtained by selecting only 1%-10% of the data in the background corpus to train the adapted LMs. The adapted LMs reduced the perplexity by up to 31%-47% relatively as compared to the perplexity of the background LM. The WER after the first-pass recognition was 14.3%. The results indicate that the target domain data contained by the background corpus can be selected and used for adaptation to reduce the WER.

The results also show that when too little data was selected, e.g. 0.1%, the adapted LM deteriorated the ASR system performance, as can be observed from the left tails of the plots in Fig. 3.3 and 3.4 that indicate higher WERs and perplexities. In such cases, although the selected data matches the target domain well, the amount of data is insufficient for training the parameters in the adapted LMs. On the other hand, when too much data was selected, out-of-domain data will be selected as the adaptation data. In such cases, the WERs of the ASR system and the perplexities of the adapted LMs became slightly higher, as shown by the right tails of the plots in Fig. 3.3 and 3.4.

In the test set of TED-LIUM corpus, the talks were actually given by different speakers on different topics (see Table 3.2), e.g. technology, science, global issues, and etc. As such we can further categorize them into different topics. In next experiment, we will evaluate the proposed adaptation framework under more specific domain criterion.

LM Adaptation to Finer-level Domain: In this experiment, the data selection technique was evaluated to see if a data in a finer-level domain, i.e. topic of the talk, can be selected from the background corpus for LM adaptation. Here, the experiment discussed in previous section was repeated, except that for each talk the LM was adapted separately.

For this experiment, the CED based data selection was performed by using only 2-gram LM which has shown the best result in the previous experiment. The WER results obtained in this experiment, referred to as ‘specific-domain’, along with the WER results obtained in previous experiment, referred to as ‘TED-domain’, are depicted in Fig. 3.5 for CED and Fig. 3.6 for VSM.

FIGURE 3.5: WER results for finer-level domain data selection using CED with 2-gram LM.

For the CED technique, achieved WER reductions are similar to the results in previous experiment, as shown in Fig. 3.5. Except for 0.1% case where significant WER reduction, i.e. 1.9% absolute, was achieved comparing to the previous experiment. This is because in this experiment each talk is adapted individually, i.e. target domain is narrower, as such, less data is required to train parameters of the adapted LM.

On the other hand, the VSM based data selection showed much better WER performance as compared to previous experiment, as shown in Fig. 3.6. For example, further WER reduction of 9% relative is achieved for the case when 1% of the background corpus is selected. Similar as in CED case, here we also observed WER

FIGURE 3.6: WER results for finer-level domain data selection using VSM with TFIDF features.

reduction for 0.1% case from 16.5% to 14.7% which is equivalent to 1.8% absolute reduction.

In this experiment, the WER reductions achieved by VSM based technique are slightly better than the ones achieved by CED. For example, the maximum WER reduction achieved by the VSM based technique is 12% relative over the baseline system, i.e. 100% of background corpus, whereas CED technique achieved 10% relative WER reduction. This observation is different in previous experiment where the VSM based technique gave the least WER reduction as compared to other CED based technique with different LMs, as shown in Fig. 3.3. This is explainable as it was the topic domain being considered in this case, and the TFIDF is very effective feature to capture the topic information. Such improvements suggest that the proposed LM adaptation framework is effective at selecting more specific domain data which resulted in further WER reduction.

The overall experiment results of best performing data selection techniques for different domains are summarized in Table 3.3. The baseline method corresponds to the case where the entire background corpus is used to train generic LM, i.e. no adaptation is performed. The best result is achieved by VSM based data selection applied to topic level domain; 1.6% absolute WER improvement (11.2% relative) over the baseline by using 5% of overall background corpus.

Method	Selected data portion (%)	Domain	WER (%)
Baseline	100	-	14.3
VSM	10	TED	13.3
CED (1-gram)	5	TED	12.9
CED (2-gram)	5	TED	12.9
CED (3-gram)	5	TED	13.1
VSM	5	Topic	12.7
CED (2-gram)	1	Topic	12.9

TABLE 3.3: The best WERs (%) achieved by different data selection techniques.

In addition, we also illustrate the experiment results for a single speaker (see Figure 3.7). We picked two speakers covering different topics from the test set. Selected speakers and their corresponding topics are: (1) Bill Gates - Innovating to zero, and (2) Dan Barber - How I fell in love with a fish. The results are on par with the general trend discussed above where adapting to finer-level domain achieves slightly better results with less data selected.

FIGURE 3.7: The comparison of WER results obtained for single speakers (Bill Gates and Dan Barber) using two different data selection techniques (CED and VSM) under two different domain settings (TED and specific).

3.4 Summary

In this chapter, we presented a novel two-pass LM adaptation approach based on data selection. Different from existing two-pass adaptation approaches, the proposed approach exploits the recognized utterances during the first-pass and data selection techniques to select relevant sentences from the large and generic background corpus. The selected sentences are then used to train domain-specific LM which is employed by the decoder during the second-pass recognition. The domain-specific LM will be representative of target domain as it was built using judiciously selected relevant sentences, i.e. similar to input test utterances. Without any prior knowledge and additional data from the external sources, the WER has been reduced by up to 10% relatively and perplexity has been reduced by up to 47% relatively, as evaluated on the 11 talks of the test set from the TED-LIUM corpus. When the LM was adapted individually for each talk, which is more specific domain, further 1.6% WER reductions have been achieved. In addition, such improvements were achieved by selecting only 1%-10% of the data from the background corpus.

The obtained results in this chapter suggest that proposed data selection based LM adaptation framework is effective at improving recognition performance of the ASR systems. We demonstrated that domain-specific LM is superior to generic LM which has better domain coverage and trained using more data. In addition, we evaluated two levels of domain, i.e. coarse and fine, and found that finer level domain adaptation achieves better results with less data selected. The proposed method is especially useful for the scenarios where the domain-specific data is unavailable and annotating such data is expensive. Such scenarios include, but not limited to, the transcription tasks for private companies, government and military related entities. Thus, the effective solution to enhance the transcription performance is to efficiently select relevant data from the general open domain corpus. Note that the proposed two-pass adaptation approach is inapplicable for online adaptation. Instead, it should be used in use-cases where speech can be transcribed offline at later time, for example, when computational power is available.

Chapter 4

Neural LM Vocabulary Adaptation by Embedding Matrix Augmentation¹

In this chapter, we present a novel vocabulary adaptation method for neural language models (NLM). In Section 4.1, we first briefly revise the overall architecture of the NLMs and describe the vocabulary mismatch problem including its cause and impact on the performance of automatic speech recognition systems.

In Section 4.2, we review existing vocabulary coverage adaptation methods. We split these methods into two groups depending on their goal, i.e. vocabulary expansion or underrepresented word enriching.

In Section 4.3, we present a novel vocabulary coverage adaptation method based on embedding matrix augmentation. The proposed method can be applied for both vocabulary expansion and rare word representation enriching. Subsequently, in Section 4.4, we evaluate the efficacy of the proposed method on the N-best hypotheses rescoring task. We compare our method against strong baseline systems where significant word error rate improvements are achieved. The analysis of the experiment results and discussion of important findings are also provided.

Lastly, Section 4.5 summarizes this chapter by discussing the important contributions and observations, and listing the potential future directions.

¹The work in this chapter has been published in [166–168].

4.1 Introduction

The neural language models (NLM) are highly used in automatic speech recognition (ASR) systems to rescore the n-best output hypotheses, especially the word-level NLMs which have been shown to achieve the state-of-the-art results [63, 72, 169]. One of the main advantages of word-level NLM (henceforth NLM) is its strong generalization capability achieved by learning the dense representation of words and using them to estimate a probability distribution function over the word sequences. However, this concept requires each word to appear a sufficient amount of times in the training data in order to learn reliable word representations. Moreover, the words not present in the training data are excluded from the vocabulary coverage of NLM by mapping them to a special symbol `<unk>`. Consequently, the NLMs tend to be biased against the rare and out-of-vocabulary words which are usually important domain-specific words.

To address this problem, in this work, we propose a method to adapt the vocabulary coverage of the pre-trained NLM. Our method is based on the embedding matrix augmentation that allows both to expand the vocabulary coverage and to enrich the representations of the rare words in NLM. Specifically, our method employs similar words to augment the word embedding matrices, also known as projection matrices, of pre-trained NLM while keeping other parameters intact. The proposed method will be presented in more details in the following sections.

This section provides the readers with the necessary background knowledge required to understand our proposed vocabulary adaptation methodology. In subsection 4.1.1, we first briefly revise the neural language model architecture decoupled into three main parts: input projection layer, middle layers and output projection layer. Specifically, we describe the functionality of each part, interaction between them and provide the mathematical formulations. Since our proposed method mainly affects the projection layers, in subsection 4.1.2, we focus on the internals of the projection layers. Lastly, subsection 4.1.3 describes the vocabulary coverage mismatch problem, including the causes of the problem and its impact on the performance of automatic speech recognition systems.

4.1.1 Neural LM Architecture

The NLMs became indispensable component of ASR systems assisting them to achieve the state-of-the-art performance [63, 72, 169]. In ASR, NLMs are mostly used to rescore a word lattice or N-best output hypotheses list. Besides, they are incorporated within some end-to-end ASR architectures, e.g. implicit language models within encoder-decoder models [45, 46]. In general, NLM architectures can be classified into two main categories: feedforward [63] and recurrent [72]. Our proposed vocabulary adaptation method is applicable to both architecture types. However, in this work, we will focus on recurrent architecture which has been shown superior to feedforward architecture [170, 171].

Recurrent NLM. The recurrent NLM architecture can be decoupled into three main parts: 1) input projection layer, 2) middle layers and 3) output projection layer (see Figure 4.1). The role and the mathematical formulation of these parts are as follows.

FIGURE 4.1: Recurrent NLM architecture decomposed into three parts.

Input projection layer. The input projection layer is defined using the input word embedding matrix S that holds the dense vector representations of the words. Specifically, the matrix S transforms the one-hot encoding representation of a word $w_t \in \mathbb{R}^{|V|}$ at time t into a dense vector representation s_t , where $|V|$ is a vocabulary size:

$$s_t = Sw_t \quad (4.1)$$

Middle layers. The dense vector s_t and a high-level context feature vector from the previous time step h_{t-1} are then combined by non-linear middle layers, which

can be represented as function $f()$, to produce a new context feature vector h_t :

$$h_t = f(s_t, h_{t-1}) \quad (4.2)$$

The non-linear function $f()$ can employ simple activation units such as ReLU and hyperbolic tangent or more complex units such as LSTM and GRU. The middle layers can be also formed by composing several such functions.

Output projection layer. Lastly, the context vector h_t is fed to the output projection layer which is parameterized by the output embedding matrix $\mathbf{U}$ to produce a high-dimensional vector $y_t \in \mathbb{R}^{|V|}$:

$$y_t = \mathbf{U}^T h_t \quad (4.3)$$

The values of the output vector y_t correspond to the likelihood of words to follow the given context h_t . These values are then normalized by softmax function to form a probability distribution.

Our proposed method modifies the embedding matrices $\mathbf{S}$ and $\mathbf{U}$ while keeping the middle layer $f()$ unchanged. Therefore, in the following subsection we provide more details about the structure and properties of embedding matrices.

4.1.2 Embedding Matrices

As was discussed in the previous subsection, the projection layers of NLM are defined by the embedding matrices $\mathbf{S}$ and $\mathbf{U}$. These embedding matrices typically hold the dense vector representation of words, also referred to as word embeddings [74]. The word embeddings has been shown to incorporate important linguistic regularities [75]. For example, the difference between vectors of woman and man is approximately the same as the difference between vectors of queen and king, i.e. $v(\text{'woman'}) - v(\text{'man'}) \simeq v(\text{'queen'}) - v(\text{'king'})$. Furthermore, they reflect the commonalities and differences between words, and thus empower NLMs with the strong generalization capability. Specifically, in the embedding space, words with the similar semantic meaning and syntactic roles are clustered together (see Figure 4.2 adopted from [172]) [173]. Consequently, NLMs can model any sequence of words including the ones which haven't been seen during the training stage.

FIGURE 4.2: A word embedding space visualized with t-SNE. Each point corresponds to one word.

4.1.3 Vocabulary Coverage Mismatch

In this subsection, we will describe the vocabulary coverage mismatch problem in NLMs. Note that the NLMs can be modeled to operate either at word or at subword (e.g. syllables, morphemes or n-gram characters) units. Although subword-level NLMs can circumvent the vocabulary coverage mismatch problem, they are usually inferior to word-level NLMs² [174]. Specifically, the subword-level NLMs have been shown effective in the end-to-end (E2E) ASR systems. The E2E ASR systems usually operate at subword-level, therefore it is common practice to employ subword-level NLM during the decoding stage by using fusion techniques [78, 175, 176]. On the other hand, the word-level NLMs have been shown to achieve the state-of-the-art results when applied during word lattice or N-best hypotheses rescoring stages for both E2E and hybrid ASR systems as have been shown in recent works [177–179]. It is important to mention that there are some works which attempt to combine advantages of both subword- and word-level features where small but promising results are reported [180]. Since our focus is to rescore hybrid ASR output hypotheses, in this chapter, we will focus on improving the word-level NLM (henceforth just NLM). There are two main issues that cause the vocabulary mismatch problem in NLMs. The first issue is the out-of-shortlist (OOS) words which will be described in 4.1.3.1. While the second issue is the underrepresented words which will be described in 4.1.3.2.

²At least for resource-rich languages such as English.

4.1.3.1 Out-of-shortlist Words

The vocabulary size is an important factor that influences the training time of NLM which can vary from several days up to several weeks depending on the scale of the application [181]. This is due to the expensive softmax computation which is the network’s main bottleneck [182]. Since the softmax computation time is proportional to the number of words in the vocabulary, the vocabulary of NLM V_{NLM} is usually limited. The V_{NLM} is a smaller subset of overall vocabulary set of ASR V_{ASR} , i.e. $V_{NLM} \subset V_{ASR}$. In practice, the V_{NLM} is usually comprised of around 10k-100k most frequent words, whereas the remaining words are marked as OOS³, and ignored or modeled using a special symbol, e.g. $\langle OOS \rangle$ [64, 112]. Consequently, the NLM will tend to underestimate the OOS words, and thus be biased against the hypotheses containing them. For example, if a test speech with many OOS words is input to ASR system, the efficacy of NLM will be suboptimal. Besides, the OOS words are usually important domain-specific keywords which are more valuable than the commonly used words. Therefore, it is important to adapt the vocabulary coverage of NLM to match the vocabulary of the test data.

4.1.3.2 Underrepresented Words

In subsection 4.1.2, we described the word embedding vectors which capture important linguistic regularities, and thus empower NLMs with the strong generalization capability. However, this concept assumes that each word appears a sufficient amount of times in the training data, e.g. at least 10 times, in order to learn reliable vector representations [183]. There are cases when the vocabulary of NLM is designed to include the rare domain-specific words from the domain of the target task. For example, the Singapore English speech recognition task requires the vocabulary set of NLM to incorporate the words related to Singapore, e.g. person, street and restaurant names. Unfortunately, such words might be absent or appear very few times in the training data, especially very rare ones. Consequently, the NLM will fail to learn robust vector representation of such underrepresented words, and thus assign them unreliable probability estimates. Furthermore, the representation

³Note that the OOS is different from the out-of-vocabulary (OOV), the OOV refers to the set of words not present in V_{ASR} , i.e. $V_{OOV} = V_{ASR}^C$. The OOV words can’t be recognized by the basic word-level ASR system. On the other hand, OOS refers to the words present in V_{ASR} , but not in V_{NLM} , i.e. $V_{OOS} = V_{ASR} \setminus V_{NLM}$.

of these rare words will be used as a context for the neighbouring words, as such, the entire word sequence containing the rare word will be underestimated. The problem exacerbates when a rare word is a named entity such as names of persons, locations, organizations and so on which are important keywords for downstream tasks. For example, in a voice search task [15], the correct recognition of location (e.g. street, district and etc.) and organization (e.g. restaurant, shop, and etc.) is crucial. Therefore, for the reliable performance of NLM, it is important to ensure that the representations of rare domain-specific words are properly learned.

In this chapter, we address the NLM vocabulary mismatch problem caused by the word usage difference between training and test data. Specifically, we propose a novel vocabulary adaptation method based on embedding matrix augmentation. The proposed method exploits the properties of the embedding matrices learned by the NLM. Importantly, the proposed method can deal with the both aforementioned issues, i.e. OOS words and underrepresented words, as will be discussed in the following sections. But first, in the next section, we will review the conventional vocabulary adaptation methods addressing these issues.

4.2 Review of Vocabulary Adaptation Methods

This section reviews conventional vocabulary adaptation methods for NLM. We split these methods into two parts corresponding to the issue they address. In the first part 4.2.1, we review the methods addressing the vocabulary mismatch problem caused by the OOS words. In the second part 4.2.2, we review the methods proposed to alleviate the impact of the unreliable rare word representation.

4.2.1 Adaptation Methods for Out-of-shortlist Words

The OOS problem in NLM has been studied extensively and many effective approaches have been proposed. These approaches mostly focus on intelligently redistributing the probability mass of $\langle OOS \rangle$ node among OOS words, optimizing the training speed for large-vocabulary models or training subword-level NLM. Some of these approaches has been also used in combination.

Redistributing probability mass of $\langle OOS \rangle$. Park et al. [112] proposed to mitigate the OOS problem by gathering all OOS words under special node $\langle OOS \rangle$ and explicitly modeling it together with the in-vocabulary words V_{NLM} . The probability mass of $\langle OOS \rangle$ node is then can be redistributed among OOS words by using statistics of simpler language models such as count-based N-gram model as follows:

$$\tilde{P}_{NLM}(w_{t+1}|h_t) = \begin{cases} P_{NLM}(w_{t+1}|h_t) & w_{t+1} \in V_{NLM} \\ \beta(w_{t+1}|h_t)P_{NLM}(\langle OOS \rangle|h_t) & otherwise \end{cases} \quad (4.4)$$

$$\beta(w_{t+1}|h_t) = \frac{P_{N-gram}(w_{t+1}|h_t)}{\sum_{w \notin V_{NLM}} P_{N-gram}(w|h_t)} \quad (4.5)$$

where $P_{NLM}(\cdot)$ and $P_{N-gram}(\cdot)$ are conditional probability estimates according to NLM and N-gram LM respectively, for some word w_{t+1} given context h_t . The N-gram model is trained with whole vocabulary set V_{ASR} , whereas the NLM is trained with the subset V_{NLM} including the $\langle OOS \rangle$ token. The $\beta(\cdot)$ is a normalization coefficient used to ensure the sum-to-one constraint of obtained probability function $\tilde{P}_{NLM}(\cdot)$.

Later, Sundermeyer et al. [169] substantially simplified the OOS word probability estimation with marginal degradation in performance. They proposed to uniformly redistribute the probability mass of $\langle OOS \rangle$ node among OOS words as follows:

$$\tilde{P}_{NLM}(w_{t+1}|h_t) = \begin{cases} P_{NLM}(w_{t+1}|h_t) & w_{t+1} \in V_{NLM} \\ \frac{P_{NLM}(\langle OOS \rangle|h_t)}{|V_{ASR} \setminus V_{NLM}| + 1} & otherwise \end{cases} \quad (4.6)$$

where ‘\’ symbol is the set difference operation. In this way, the vocabulary coverage of NLM is expanded to the full vocabulary size $|V_{ASR}|$ without incurring costly training operations and relying on the statistics of simpler LMs. Currently, this is a standard scheme commonly employed in the state-of-the-art NLMs.

Training speed optimization. Rather than reducing the vocabulary coverage of NLM to subset V_{NLM} , this group of studies keeps the entire set V_{ASR} and focuses on speeding-up the training of large-vocabulary NLMs. One of the most effective ways to speed up the training of NLMs is to approximate the softmax function. The softmax function is used to normalize obtained scores y_t to form a probability

distribution:

$$P_{NLM}(w_{t+1} = j|h_t) = \text{softmax}(y_t^j) = \frac{\exp(y_t^j)}{\sum_{i \in V_{NLM}} \exp(y_t^i)} \quad (4.7)$$

Consequently, its computational cost is proportional to the number of words in the vocabulary and it dominates the training of the whole model which is the network’s main bottleneck [182].

Many techniques have been proposed to approximate the softmax computation. The most popular ones include hierarchical softmax [73, 184, 185], importance sampling [186, 187] and noise contrastive estimation [14, 182]. The comparative study of these techniques can be found in [188, 189]. Other techniques, besides softmax function approximation, to speed up the training of large-vocabulary models can be found in [190].

Subword-level NLM. Another effective method to expand the vocabulary coverage is to train a subword-level NLM. Different from the standard word-level NLMs, they model finer linguistic units such as characters [191] or syllables [174], hence, a larger range of words will be covered. Furthermore, subword-level NLM doesn’t suffer from the OOS problem, though, it performs worse than word-level models [174]. Recently, there has been a lot of research effort aiming to train the hybrid of word and subword level models where promising results are obtained [180, 181, 192].

4.2.2 Adaptation Methods for Rare Word Representations

The continuous vector representations of words are typically derived from large unlabeled corpora using co-occurrence statistics [193, 194]. They became the dominant feature for many natural language processing applications achieving the state-of-the-art results. To generalize well, however, these tasks require many occurrences of each word and fall short if a word appears only a handful of times [195]. Several approaches have been proposed to deal with the rare word problem and most of them can be classified under one of the three main categories shown below.

Morphological word representations. A bunch of proposed works resorts to subword level linguistic units by breaking down the words into morphemes [195–197]. For example, [195] represented words as a function of its morphemes where

the recursive neural network is applied over morpheme embeddings to obtain the embedding for a whole word. While such works have been proven effective to deal with the infrequent word variations, they depend on the morphological analyzer such as *Morfessor* [198] and unable to model words whose morphemes haven't been seen during the training stage.

Character-level representations. To alleviate the rare word problem, finer level linguistic units such as syllables and characters have been also studied [174, 192, 199, 200]. For example, [174] explored both word-syllable and word-character level hybrid NLMs where most frequent words are kept unchanged, while rare words are split into the syllables and characters, respectively. In similar fashion, [199] and [192] examined character-aware NLM architectures which rely only on character level inputs, but predictions are still made at the word level.

Character-level models eliminate the need for morphological tagging or manual feature engineering, and they comprise substantially fewer number of parameters compared to the word-level models. Moreover, these approaches success at capturing properties of morphologically related words, e.g. 'run' vs 'running', but may fail to capture distinctions between semantically unrelated words, e.g. 'run' vs 'rung' [183, 200].

Knowledge-powered word representations. Another direction of works leverage external knowledge to enhance representations of rare words [183, 201, 202]. For example, [183] employed word definitions obtained from the *WordNet* [203] to model rare words on the separate network. Alternatively, [201] proposed to incorporate external knowledge as a regularization term to the original model's objective function. Although these approaches have shown promising results, they highly depend on the availability of external hand engineered lexical resources.

Note that the aforementioned approaches can be also used jointly. For example, by using factored NLM architecture [57] where different feature types can be combined.

4.3 Embedding Matrix Augmentation

In this section, we present a novel vocabulary adaptation method based on embedding matrix augmentation for pre-trained NLMs. Our method is based on the

observation that input and output projection layers of NLM learn the word embedding matrices, and middle layers learn the mapping from the input word embedding vectors to compressed context vectors. Thus, by modifying the input and output word embedding matrices, we can adjust the vocabulary coverage of pre-trained NLM. Meanwhile, the parameters of middle layers are kept unchanged which allows us to avoid expensive re-training. This approach will preserve the linguistic regularities encapsulated within pre-trained NLM as long as the words are properly modified in the input and output embedding spaces. To modify these words, we will use word embedding vectors of semantically and syntactically similar words present in V_{NLM} .

Importantly, the proposed method can be applied to deal with both OOS problem and unreliable rare word representations. In particular, to deal with the OOS problem, we employ our method to expand the vocabulary coverage of pre-trained NLM V_{NLM} , as will be described in subsection 4.3.1. To address the unreliable rare word representation, we employ our method to enrich the rare word representations in pre-trained NLM which will be described in subsection 4.3.2.

4.3.1 Expanding Vocabulary Coverage

We start with the assumption that we are given a pre-trained NLM and a N-best hypotheses list output from the ASR system. The goal is to re-score the N-best list using the NLM. But first, we would like to ensure that the vocabulary coverage of NLM incorporates the majority of the words in N-best list to avoid the sub-optimal performance. A trivial solution would be to extract all OOS words from the N-best list, append them to V_{NLM} and then re-train the NLM. However, this solution has two main drawbacks:

- 1) The NLM re-training process might take from several hours up to several days depending on the scale of the task.
- 2) The extracted OOS words might be absent in the training data, and thus require additional training data collection.

Instead, we propose an efficient method to expand the vocabulary set of a pre-trained NLM without incurring expensive re-training and using additional training

data. The proposed method consists of three main steps: 1) identifying the relevant OOS words, 2) finding similar words and 3) expanding the input and output embedding matrices. These steps will be described next in great details.

Identifying the relevant OOS words. The first step to vocabulary expansion is finding relevant OOS words. This step is important as expanding the NLM vocabulary with irrelevant words absent in the input test data is futile. Moreover, the irrelevant words might further degrade the performance of NLM. To find relevant OOS words either supervised or unsupervised approaches can be employed. For example, in supervised approaches, they are manually collected by human experts. On the other hand, in unsupervised approaches, the subset of most frequent or random OOS words are automatically selected. In this work, we resort to unsupervised approach. However, instead of selecting most frequent or random subset of OOS words, we propose to extract the relevant OOS words from the ASR output, i.e. N-best hypotheses list⁴. The motivation is that generated N-best list will contain the most relevant OOS words that might be present in the input test data.

Finding similar words. Given a list of relevant OOS words, let's call it set V_{new} , the next step is to find candidate words. The candidate words will be used to model selected OOS words. Therefore, they must be present in V_{NLM} set and should be similar to the target OOS word in both semantic meaning and syntactic behavior. Selecting inadequate candidates might deteriorate the linguistic regularities incorporated within pre-trained NLM, thus, they should be carefully inspected. In natural language processing, many effective techniques exist that can find appropriate candidate words satisfying conditions mentioned above. For example, lexical resources with synonym list, e.g. WordNet [204] and Thesaurus⁵, or pre-trained data-driven models that can measure similarity between words, e.g. skip-gram, CBOW and Glove [193, 194].

Expanding word embedding matrices. As was previously mentioned, our method modifies the input and output embedding matrices $\mathbf{S}$ and $\mathbf{U}$, while keeping the middle layers unchanged. Below, we will explain the proposed vocabulary expansion method for each matrix.

1) The matrix $\mathbf{S}$: This matrix holds input word embedding vectors $\mathbf{S} = [s_1 \dots s_{|V_{IS}|}]$ and it's used to project words from discrete form, i.e. one-hot encoding, into lower

⁴A word lattice can be also used to extract relevant OOS words

⁵<https://www.thesaurus.com/>

dimensional continuous space, also known as word embedding space. One of the important properties of the embedding space is that it groups ‘similar’ words into the semantic clusters [173] as was discussed in subsection 4.1.2. Moreover, the embedding vectors have been also shown to capture important syntactic features of language [75]. Subsequently, if two words have similar semantic and syntactic roles, their embedding vectors are expected to belong to the same cluster. As such, words in a cluster can be used to approximate a new word that belongs to the same cluster.

For example, let’s consider a scenario where we want to add a new word ‘Nur-Sultan’, the capital of Kazakhstan, to the vocabulary set of an existing NLM. Here, we can select candidate words from V_{NLM} with similar semantic and syntactic roles such as ‘London’ and ‘Paris’. Specifically, we extract the input word embedding vectors of selected candidates $\mathcal{C} = \{s_{London}, s_{Paris}\}$ and combine them to form a new input word embedding vector $\hat{s}_{Nur-Sultan}$ as follows:

$$\hat{s}_{Nur-Sultan} = \frac{\sum_{s \in \mathcal{C}} m_s s}{|\mathcal{C}|} \quad (4.8)$$

where m_s is some normalized metric used to weigh the candidates. We repeat this procedure for all words in set V_{new} using the corresponding candidate lists.

Obtained new input word embedding vectors are then used to form matrix $\hat{\mathbf{S}} = [\hat{s}_1 \dots \hat{s}_{|V_{new}|}]$, which is appended to the initial matrix $\mathbf{S}$ to form the expanded matrix $\mathbf{S}^+$ as follows:

$$\mathbf{S}^+ = [\mathbf{S} \quad \hat{\mathbf{S}}] \quad (4.9)$$

The input word vector $w_t \in \mathbb{R}^{|V_{NLM}|}$ should be also expanded to accommodate the new words from V_{new} , which results in a new input vector $\hat{w}_t \in \mathbb{R}^{|V_{NLM}|+|V_{new}|}$. The input vector and input word embedding matrix after the expansion are depicted in figure 4.3.

2) The matrix $\mathbf{U}$: This matrix holds the output word embedding vectors $\mathbf{U} = [u_1 \dots u_{|V_{NLM}|}]$. These vectors are compared against context vector h_t using the dot product to determine the score of the next possible word w_{t+1} [182]. Intuitively, for a given context h_t , the interchangeable words with similar semantic and syntactic roles should have similar scores to follow it. Therefore, in the output word

embedding space, interchangeable words should belong to the same cluster. Subsequently, we can use the same procedure and candidates which were used to expand the matrix $\mathbf{S}$ to model the new words in output word embedding space. However, this time we operate in the column space of matrix $\mathbf{U}$. The output vector and output word embedding matrix after the expansion are depicted in figure 4.3.

4.3.2 Enriching Rare Word Representations

We start with the same assumption as in the previous subsection, i.e we are given a pre-trained NLM and N-best hypotheses list output from the ASR system. However, the rare word representation enriching is different from the vocabulary coverage expansion method that aims to reduce the OOS word coverage gap between pre-trained NLM and N-best hypotheses list. Here, we aim to enrich the representation of the rare words already present in the vocabulary set of pre-trained NLM V_{NLM} . This problem is common and more challenging due to the extremely rare domains for which finding a textual training data is difficult. Additionally, this problem has no trivial solution such as re-training the NLM with modified vocabulary, unless an additional data containing the rare words is collected. To this end, we propose an effective method based on word embedding matrix augmentation which doesn't require expensive post-processing operations and additional in-domain training data. The proposed method has three main steps: 1) identifying the rare words, 2) finding similar words and 3) enriching rare word representations.

Identifying the rare words. To identify the rare words, we can simply count the frequency of words in the training data and set a frequency threshold below

FIGURE 4.3: NLM architecture after vocabulary expansion.

which all words are considered rare. This approach, however, might result in too many rare words. To reduce the computation time, we can limit the rare words to those which are present in the produced N-best hypotheses list.

Finding similar words. Given a subset of rare words $V_{rare} \subset V_{NLM}$, the next step is to select a list of similar candidate words $\mathcal{C}$ for each rare word. The selected candidates will be used to enrich representations of rare words, hence, they must be frequent and present in the vocabulary V_{NLM} . In addition, they should be similar to the target rare word both in semantic meaning and syntactic behavior. Note that selecting inadequate candidates might deteriorate the performance of NLM, thus, they should be carefully inspected.

Several effective methods exist that can find appropriate candidate words. For example, using lexical resources that contain synonyms and related words such as WordNet [204] or employing the pre-trained word embeddings from skip-gram, CBOW or Glove models [193, 194] which can also find the similar words.

Enriching rare word representations. Let s_r be an embedding vector of some rare word w_r in space defined by input embedding matrix $\mathbf{S}$ and let $\mathcal{C}_r$ be corresponding set of similar words. We enrich the representation of s_r using the words in $\mathcal{C}_r$ by the following formula:

$$\hat{s}_r = \frac{s_r + \sum_{s_c \in \mathcal{C}_r} m_c s_c}{|\mathcal{C}_r| + 1} \quad (4.10)$$

where $\hat{s}_r$ is the enriched representation of s_r , s_c is an embedding vector of similar candidate word and m_c is a metric used to weigh candidates based on importance. The m_c can be estimated using frequency counts or similarity score where most frequent or most similar candidates are given higher weights.

The Eq. (4.10) typically shifts the embedding of a rare word towards the weighted centroid of its semantic landmark. The motivation is that highly correlated words, in terms of both semantic meaning and syntactic behavior, should be close to each other in the embedding space. We then use the same candidates and formula to update the corresponding rare word embedding u_r in the output embedding matrix $\mathbf{U}$. This procedure is then repeated for the remaining words in subset V_{rare} .

4.4 Experiments

In this section, we describe the experiments conducted to evaluate the effectiveness of the proposed embedding matrix augmentation technique. Particularly, we conducted two sets of experiments. In the first set of experiments, we assessed its application for the vocabulary expansion method (subsection 4.4.1). While in the second set of experiments, we assessed its application for the rare word representation enriching (subsection 4.4.2).

4.4.1 Experiments for Vocabulary Coverage Expansion

This subsection describes the experiments conducted to evaluate the efficacy of the proposed vocabulary expansion method for pre-trained NLMs on ASR task. The ASR is DNN-HMM system built using the Kaldi [35] speech recognition toolkit on the first release of TED-LIUM [163] speech corpus. To highlight the importance of the vocabulary expansion, we used an average sized vocabulary set for LMs and trained the LMs on generic-domain text corpus 1 Billion Word Benchmark (1BW) [164].

As the baseline LMs, we trained two state-of-the-art models. One is based on conventional N-gram LM and other is NLM. The N-gram LM is the modified Kneser-Ney smoothed 5-gram (KN5) [69], while the NLM is recurrent LSTM network [169]. We call our system VE-LSTM which was constructed by expanding the vocabulary of the baseline LSTM. The performances of these three models were compared using both the perplexity and the word error rate (WER) metrics.

4.4.1.1 Experiment Setup and Data

Speech corpus. The TED-LIUM speech corpus is comprised of monologue talks given by experts on specific topics, its characteristics are given in Table 4.1. Its train set was used to build the acoustic model with the ‘nnet3+chain’ setup of Kaldi including the latest developments. Its dev set was used to tune hyper-parameters such as the word insertion penalty, the LM scale and the number of candidates to use to model the new words. The test set was used to compare the performance of proposed VE-LSTM and two baseline models. Additionally, the TED-LIUM

corpus has a pre-designed pronunciation lexicon of 150k words which was used as a vocabulary set for the baseline LMs.

Text corpus. The 1BW text corpus consists of text collected from various domains including the news and parliamentary speeches. Its train set, containing around 700M words, was used to train both baseline LMs. Its validation set of size 141k words was used to stop the training of LSTM model.

Baseline LMs. The baseline KN5 was trained using the SRILM [165] toolkit with 150k vocabulary set. It was used to rescore both the word lattice and 300-best list. Its pruned version KN5_pruned⁶ was used as a background LM to construct the ASR system.

The baseline LSTM was trained as a three-layer network similar to [169] using our own implementation in PyTorch [205]. The LSTM explicitly modeled only the 10k most frequent words⁷ of 150k vocabulary set. The remaining 140k words were modeled by uniformly distributing the probability mass of $\langle OOS \rangle$ node using the Eq. (4.6). Thus, the input w_t and output y_t vector sizes were $10k + 1$ which we call V_{NLM} set. Hence, the baseline LSTM theoretically modeled the same 150k vocabulary set as KN5. The OOS rate with respect to dev and test sets were 6.8% and 5.5%, respectively. The input and output embedding vector dimensions were set to 300 and 1500, respectively. The parameters of the model are learned by truncated backpropagation through time algorithm (BPTT) [206] unrolled for 10 steps. For regularization, we applied 50% dropout on the non-recurrent connections as suggested by [207].

Vocabulary Expanded LSTM. The VE-LSTM model is obtained by expanding the vocabulary of baseline LSTM with OOS words extracted from the ASR output. For example, to construct the VE-LSTM model for the test set, we collect the list of

⁶The LM pruning is a common practice used to facilitate the computationally expensive decoding stage of hybrid ASR systems. The pruning coefficient was set to 10^{-7} .

⁷Plus the beginning $\langle s \rangle$ and end of sentence $\langle /s \rangle$ symbols.

Characteristics	Train	Dev	Test
Number of talks	774	8	11
Number of words	1.3M	17.7k	27.5k
Total duration	122hr	1.5hr	2.5hr

TABLE 4.1: The characteristics of TED-LIUM corpus.

OOS words V_{new} from the recognized hypotheses of the test set. For each OOS word in V_{new} , we then select the appropriate set of candidate words C . The selection criteria will be explained later. Next, selected candidates are used to model the new input and output word embedding vectors of target OOS words as in Eq. (4.8). For simplicity, we didn't weigh the selected candidates. Lastly, these generated new vectors are appended to the input and output word embedding matrices of baseline LSTM model, see Figure 4.3. Consequently, the obtained VE-LSTM will explicitly model $10k + 1 + |V_{new}|$ words, whereas the remaining $140k - |V_{new}|$ words are modeled by uniformly distributing the probability mass of $\langle OOS \rangle$ node using Eq. (4.6).

To select candidate words we used the classical skip-gram model [193]. The skip-gram model was trained with default settings on 1BW corpus covering all unique words. Typically, when presented with a target OOS word, the skip-gram model returns a list of 'similar' words. From this list, we only selected top eight⁸ candidate words which are present in V_{NLM} set.

4.4.1.2 Experiment Results and Discussion

The experiment results are given in Table 4.2. We evaluated LMs using the perplexity and WER metrics. The perplexity was computed on the reference data where the OOS words for vocabulary expansion were extracted from the reference data as well. The perplexity results computed on the test set show that VE-LSTM significantly outperforms KN5 and LSTM models by 31% and 18% relative, respectively.

LM	Perplexity		WER (%)		
	Dev	Test	Re-score	Dev	Test
KN5_pruned	384	341	-	13.05	12.82
KN5	237	218	Lattice	10.85	10.49
			300-best	11.03	10.79
LSTM	211	185	300-best	10.63	10.18
VE-LSTM	163	150	300-best	10.44	9.77

TABLE 4.2: The perplexity and WER results on dev and test sets of TED-LIUM corpus.

⁸This number was tuned on the dev set.

For WER experiment, the KN5 is evaluated on both lattice and 300-best rescoring tasks. The LSTM and VE-LSTM are evaluated only on 300-best rescoring task. We tried to extract the OOS words for vocabulary expansion from different N -best lists. Note that OOS are the words which are present in the ASR vocabulary, but not in the vocabulary of neural LM. We can extract them by taking difference between ASR and LM vocabularies or by inspecting ASR output hypotheses. The former one will produce too many words including the irrelevant OOSs, whereas the later one will include only relevant OOS words which are very likely to be constituent of the input test utterances. Interestingly, the best result is achieved when they are extracted from the 1-best. The reason is that 1-best hypothesis contains words with the highest confidence score, hence, OOS words extracted from it will be reliable. Whereas using other N -best lists will result in unreliable OOS words which confuse the VE-LSTM model. The VE-LSTM outperforms the baseline LSTM model by 4% relative WER. Compared to KN5 used to rescore the word lattice, 7% relative WER improvement is achieved. Such improvements suggest that the proposed vocabulary expansion method is effective.

4.4.2 Experiments for Rare Word Representation Enriching

In this section, we describe experiments conducted to evaluate the effectiveness of the proposed embedding matrix augmentation technique for the rare word representation enriching. Particularly, we first enrich the rare Singapore street name representations in pre-trained NLM and then use the enriched NLM to rescore 100-best hypotheses output from the Singapore English ASR. The ASR system is built by Kaldi [35] speech recognition toolkit using Singapore English speech corpus. To highlight the importance of enriching the rare word representations, we used 1 hour recording of 9 read articles about Singapore streets as an evaluation set⁹ (7.3k words).

We compare our enriched model against three the state-of-the-art LMs including Kneser-Ney smoothed 4-gram¹⁰ (KN4), Kaldi-RNNLM [180] and recurrent LSTM LM [169]. Our model is obtained by enriching the representations of rare Singapore street names in the recurrent LSTM LM, and we call it E-LSTM. The performance of these four LMs is evaluated on the 100-best rescoring task.

⁹https://github.com/khassanoff/SG_streets

¹⁰We also examined other N -gram models and found 4-gram to work best for our case.

4.4.2.1 Experiment Setup and Data

Acoustic model. The acoustic model (AM) is built using ‘nnet3+chain’ setup of Kaldi and trained on 270 hours of transcribed Singapore English data which mostly consist of speech taken from parliament, talk shows and interviews.

Lexicon. The lexicon is constructed by assembling 51k unique words which include around 2k Singapore street names. To avoid ambiguity, the street names consisting of more than one word are joined using the underscore symbol, e.g. ‘Boon lay’ is changed to ‘Boon_lay’. This lexicon was also used as a vocabulary set for LMs.

Text corpus. To train LMs, we used AM transcripts and web crawled Singapore related data which resulted in total 1M in-domain sentences (16M words). In addition, we used Google’s 1 Billion Word (1BW) benchmark corpus [164] to account for generic English word sequence statistics.

Baseline LMs. The KN4 is trained on combined in-domain (AM transcript+web crawled) and generic 1BW data. It was built using SRILM toolkit [165] with 51k vocabulary set. We used KN4 model to rescore both word lattice and 100-best list. Its pruned version KN4_pruned was used during the decoding stage.

The Kaldi-RNNLM is a word-character level hybrid model designed to overcome the rare word problem by decomposing the rare words into character n -grams while keeping the most frequent words unchanged. It was trained as a 2-layer LSTM¹¹ with 800 units in each layer using only in-domain data. The input and output embedding matrices were tied and embedding space dimension was set to 800. For vocabulary, we tried to keep a different number of most frequent words and found 20k to perform best, hence, the remaining 31k words were decomposed into character n -grams.

The recurrent LSTM LM is a word level model which was built using our own implementation in PyTorch [205]. It was trained as a single layer LSTM with 1k units using in-domain data and 51k vocabulary set. The input and output embedding space dimensions were set to 300 and 1000, respectively. The parameters of the model were learned by truncated BPTT [206] and SGD with gradient clipping. We also applied dropout for regularization [207].

¹¹Changing the number of layers and its size didn’t improve WER.

Lastly, our E-LSTM model is obtained by enriching the rare word representations in the pre-trained recurrent LSTM LM. As a case study, we use Singapore street names¹² where frequent streets will be used to enrich the rare street representations. In particular, we first count the frequency of each street name in the in-domain data and then divide them into two subsets of frequent and rare streets using some threshold value. Next, we randomly choose words from the subset of frequent streets and employ them to enrich the representations of all rare streets (*allStreets*) using the equation (4.10). To reduce computation time, we also tried to enrich only the rare street names present in the 100-best hypotheses output (*fromNbest*).

4.4.2.2 Experiment Results and Discussion

The experiment results are shown in Table 4.3. In these experiments, we divide the street names into frequent and rare subsets using the threshold value of 10. To enrich the rare street representations, we used 5 randomly chosen frequent street names¹³. The initial WER without any re-scoring is 17.07%.

The obtained results show that the E-LSTM model outperforms the strong KN4 used to rescore the word lattice by 16% relative WER (from 16.52% to 13.83%). Moreover, it achieves 6% relative WER improvement over Kaldi-RNNLM and LSTM models (from 14.73% to 13.83%). We found that enriching only rare streets present in the 100-best hypotheses (*fromNbest*) achieves a similar result as enriching all rare streets (*allStreets*), while *fromNbest* being much faster.

The state-of-the-art WER results are usually achieved by interpolating NLM and count-based N-gram models which have been shown to complement each other [72, 169]. To this end, we interpolated¹⁴ NLMs with KN4 and achieved further WER reductions. Interestingly, the baseline LSTM model doesn't benefit from KN4¹⁵, while E-LSTM gains additional 2% relative WER improvement (from 13.83% to 13.55%).

Changing the frequency threshold value. To determine the effective frequency threshold range, used to split the street names into frequent and rare subsets, we repeated the experiment with different threshold values as shown in Figure 4.4.

¹²<https://geographic.org/streetview/singapore/>

¹³For consistency, we fix the chosen frequent street names to be same.

¹⁴Interpolation weight for KN4 is set to 0.3.

¹⁵Changing the interpolation weight didn't help.

LM	Perplexity	Rescore	WER
KN4_pruned	436	-	17.07%
KN4	351	Lattice 100-best	16.52% 16.84%
Kaldi-RNNLM	-	100-best	14.73%
+KN4	-	100-best	14.10%
LSTM	295	100-best	14.74%
+KN4	-	100-best	14.95%
E-LSTM (allStreets)	242	100-best	13.87%
+KN4	-	100-best	13.58%
E-LSTM (fromNbest)	234	100-best	13.83%
+KN4	-	100-best	13.55%

TABLE 4.3: The perplexity and WER results on evaluation set. The "+KN4" represent results obtained by interpolating neural LMs with baseline KN4 LM. The E-LSTM is obtained by enriching rare words in baseline LSTM.

We observed that setting it between 5 and 50 is sufficient to achieve good results. On the other hand, setting it too low or high deteriorates the WER as can be seen from the left and right tails of the plot in Figure 4.4.

FIGURE 4.4: WER performance of E-LSTM at different frequency threshold values used to split frequent and rare street names.

Changing the number of frequent words. We also repeated the experiment to determine the optimal number of candidate frequent words to use to enrich the rare word representations. We observed that for all cases the WER results are

LM	Rescore	Accuracy
KN4_pruned	-	37.36%
KN4	Lattice	36.98%
Kaldi-RNNLM	100-best	43.40%
LSTM	100-best	43.02%
E-LSTM (fromNbest)	100-best	59.25%

TABLE 4.4: Recognition accuracy of 265 rare street names.

similarly good. For fast computation, we recommend to use around 3-20 most frequent words. This experiment is incomplete as we didn't examine the quality of selected frequent words which requires more substantial analysis. We leave the further analysis for future work.

Recognition accuracy of enriched rare words. To ensure that WER improvements are achieved as a result of correctly recognizing the enriched rare street names, we compute the recognition accuracy of 265 rare street names (see Table 4.4). The experiment results show that after enriching the baseline recurrent LSTM LM, the recognition accuracy is increased by 16.23% (from 43.02% to 59.25%) achieving the best result among all LMs. Furthermore, we observe that correctly recognizing the rare street names also helps to recover neighbouring words (see Table 4.5). These results confirm the effectiveness of the proposed method.

LM	Examples
KN4_pruned	1) a hawker centre and market began operations at bully plays in nineteen seventy six 2) by nineteen ninety four when the book development guide plan was announced
LSTM	1) a hawker centre and market began operation at bully police in nineteen seventy six 2) by nineteen ninety four when the product development gap plan was announced
E-LSTM	1) a hawker centre and market began operations at boon_lay place in nineteen seventy six 2) by nineteen ninety four when the bedok development guide plan was announced

TABLE 4.5: Examples of correctly recovered neighbouring words after rescoring with E-LSTM.

4.5 Summary

In this chapter, we described our proposed vocabulary adaptation method for NLMs. Specifically, the proposed method can be used to expand the vocabulary coverage or to enrich the rare word representations in pre-trained NLMs. This is achieved by augmenting the embedding matrices of pre-trained NLM, while keeping other parameters unchanged. Importantly, our method doesn't require additional in-domain training data and expensive post-training procedures. We evaluated the proposed method on both vocabulary expansion and rare word representation enriching tasks where the vocabulary adapted NLM was used to rescore the N-best hypotheses output from the state-of-the-art ASR systems. The experiment results suggest that the proposed method significantly improves both perplexity and WER metrics when compared to other strong methods. We believe that our approach of manipulating the input and output projection layers can be easily applied to other neural network models with similar architectures.

Chapter 5

Code-Switching Practice Adaptation by Constrained Output Embeddings¹

The code-switching is a practice of mixing two or more languages within single discourse frequently practiced in multilingual societies such as Singapore and India. The code-switching imposes serious problems to the existing monolingual ASR systems. One of the main problems is a scarcity of labeled code-switching data which is insufficient to build robust code-switching ASR. To address this problem, in this chapter, we propose a method to adapt an End-to-End (E2E) ASR system trained using only monolingual data to handle code-switching utterances.

In Section 5.1, we explain the code-switching phenomenon including the reasons for code-switching, its different types and examples. Additionally, we briefly describe the architecture of end-to-end ASR and emphasize its main advantages over hybrid ASR for the code-switching speech recognition task. In Section 5.2, we review previous works addressed the data scarcity problem in code-switching ASR. In Section 5.3, we propose our novel method for improving the performance of E2E code-switching ASR built using only monolingual data. In Section 5.4, we evaluate the proposed method on Mandarin-English code-switching speech recognition task. Finally, the Section 5.5 summarizes this chapter by pointing out important findings and future work.

¹The work in this chapter has been published in [208].

5.1 Introduction

The code-switching (CS) is a practice of mixing two or more languages within a single discourse which poses a serious problem to many speech and language processing applications. There are two main types of CS practice: intersentential and intrasentential. In intersentential CS (inter-CS), a language switch takes place at the sentence boundaries, whereas in intrasentential CS (intra-CS), a language switch occurs within a single sentence. Among these two types, the intra-CS is considered more challenging. Besides, the labeled intra-CS data is extremely rare, whereas the inter-CS data can be artificially created by combining monolingual data from different languages.

Recently, the CS practice start to immensely grow, especially in the world wide web [209], which attracted the attention of language and speech processing community. Especially, the end-to-end (E2E) code-switching automatic speech recognition (ASR) gained increasing interest where impressive improvements have been reported [167, 210–212]. These improvements, however, are mainly achieved for the CS languages where sufficient amount of transcribed CS data is available such as Mandarin-English [213]. Unfortunately, for the vast majority of other CS languages the labeled CS data, especially the intra-CS type, remains too small.

To address this problem, in this work, we propose a method to improve the intra-CS speech recognition capability of E2E-CS-ASR built using only artificially created inter-CS data. Specifically, our method constrains the output token embeddings of monolingual languages, residing within internal LM of E2E-CS-ASR, to be similar in order to facilitate the easy code-switching between languages. We examined two constraint types based on Jensen-Shannon divergence and cosine distance, though, other distance metrics can be also used. The proposed method can be also viewed as a CS practice adaptation from inter-CS to intra-CS type. The comprehensive description of the proposed method will be provided in the following section.

In this section, we first provide a background information on code-switching practice followed by the brief description of E2E ASR architecture.

5.1.1 Code-switching

The code-switching, the alternating use of two or more languages within single discourse, is a well-known and extensively studied phenomenon [214]. This phenomenon is mainly caused by globalization, immigration and colonization to mention few, where the mix of different cultures and languages take place. Some scholars use the CS term to refer to the mix of dialects and styles as well, for example from formal to informal settings. In this work, we will use it in the sense of mixing different languages.

The CS practice is ubiquitous in multilingual societies such as Singapore (Mandarin-English), USA (Spanish-English), India (Hindi-English, Tamil-English, etc.), Algeria (Arabic-French), Kazakhstan (Kazakh-Russian) and many others. In these societies, the CS is practiced on a regular basis. Usually, multilingual speakers switch from one language to another accidentally, but in some situations, language switching is done deliberately. Especially, multilingual speakers widely employ CS in social network activities such as posts and comments on Facebook and Twitter [209, 215].

Reasons for code-switching. Linguists have proposed several hypotheses regarding the reasons for switching language within single conversation. The popular reasons include quoting someone, to express solidarity and gratitude, to clarify, to express group identification, to soften or strengthen command and to exclude others. In these scenarios, CS allows to maintain exact meaning of a word (or phrase), to make communications more effective and meaningful, to convey the original feelings and emotions, to keep something in secret, to provide inspiration as well as emphasize the spoken words.

Additionally, to explain cognitive reasons and motivations behind code-switching, several linguistic theories have been developed such as constraint-based model [216] and matrix language-frame model [217]. The constraint-based model is established around two simple constraints: free-morpheme and equivalence. The free-morpheme constraint stipulates that code-switching cannot occur between a lexical stem and bound morphemes. The equivalence constraint predicts that switches occur only at points where the surface structures of the languages coincide, or between sentence elements that are normally ordered in the same way by each individual grammar. On the other hand, the matrix language-frame model posits that in

code-switching there is a matrix language (or primary language) and an embedded language (or secondary language). And the elements of the embedded language are inserted into the morphosyntactic frame of the matrix language. The matrix language-frame is widely cited and considered the dominant model of insertional code-switching. For a detailed review of linguistic theories on code-switching phenomena, we refer the interested readers to [218, 219] and the references therein.

Types of code-switching. In literature, several types of CS has been defined, among them the two popular ones are described below:

- **Intersentential:** the language switch occurs at sentence boundaries.
- **Intrasentential:** the language switch occurs within a sentence.

The intersentential CS speech once considered challenging but now can be recognized using multilingual ASR systems with high accuracy. On the other hand, intrasentential CS speech still causes existing ASR systems to fail, and thus, it is main focus of many research groups [220]. Similarly, in this work, we will focus on recognizing intrasentential CS speech.

Code-switching speech recognition. Due to the globally increasing trend of code-switching practice, it has recently attracted much attention from the natural language processing (NLP) community [215, 221, 222]. Especially, the code-switching automatic speech recognition has gained prominence among tech giants such as Google, Microsoft, Amazon, Alibaba, Tencent and many others. Presumably, as a consequence of the increasing popularity of new applications of ASR on IoT and mobile devices such as short message dictation, virtual assistant, voice search and commands where code-switching is prevalent among multilingual speakers.

Although CS-ASR has undergone impressive advancements, most of them are achieved for hybrid deep neural network and hidden Markov model (DNN-HMM) based systems which have several limitations. Firstly, state-of-the-art DNN-HMM based ASR systems mostly operate in a word level which makes them vulnerable to out-of-vocabulary (OOV) problems. Secondly, a huge lexicon containing words from different languages will be needed which will increase the computational complexity of the model, and much human efforts will be required to label word

pronunciations. Lastly, components of DNN-HMM ASR such as acoustic and language models are optimized separately with different objectives which might result in optimization incoherence [48].

In response to the aforementioned limitations, researchers resort to end-to-end ASR architectures which jointly model all components of the hybrid ASR under a single neural network [23, 45]. Additionally, the E2E-ASR outputs character or sub-word level tokens which makes them invulnerable to OOV. In terms of code-switching speech recognition task, different E2E architectures have been explored such as connectionist temporal classification (CTC) [210], attention-based encode-decoder [211] and joint CTC/Attention models [167, 212] where remarkable results are attained. However, all these models are built for languages where sufficient amount of transcribed code-switching data is available such as Mandarin-English [213]. Whereas for the vast majority of other code-switching languages the transcribed data remains too small or even non-existent.

In this work, assuming no code-switching data is available, particularly the intrasentential type, we aim to build E2E-CS-ASR using only monolingual data from each individual language. Specifically, we will improve the CS speech recognition capability of E2E-CS-ASR which is similar to the task of adaptation from disjoint monolingual domains to cross-lingual domain. We note that this setup is different from previous works where the code-switching systems are built using labeled CS data [167, 210–212] and other cross-lingual resources such as word-aligned parallel corpus [223] or dictionary [224]. Our assumption is crucial, since in real-life applications the labeled code-switching data and other cross-lingual resources are extremely rare.

5.1.2 End-to-end ASR Architecture

Figure 5.1 illustrates the baseline E2E-CS-ASR model based on hybrid CTC/Attention architecture [49] which incorporates the advantages of both CTC model [21] and attention-based encoder-decoder model [45]. Specifically, the CTC and attention-based decoder modules share a common encoder network module and are jointly trained within multi-task learning framework. During the beam search process, the predictions of CTC and attention-based decoder modules are combined. The detailed descriptions of these modules are explained next.

FIGURE 5.1: Hybrid CTC/Attention end-to-end ASR architecture with constrained output token embeddings. The output token embeddings are learned by the parametric matrix of linear output projection layer (OutputProj).

Encoder. The shared encoder network takes a sequence of T -length speech features $\mathbf{x} = (x_1, \dots, x_T)$ and transforms them into L -length high-level representations $\mathbf{h} = (h_1, \dots, h_L)$ where $L < T$. The encoder is modeled as a deep convolutional neural network (CNN) based on the VGG network [225] followed by several bidirectional long short-term memory (BLSTM) layers.

$$\mathbf{h} = \text{BLSTM}(\text{CNN}(\mathbf{x})) \quad (5.1)$$

CTC module. The CTC sits on top of encoder and computes the posterior distribution $P_{\text{CTC}}(\mathbf{y}|\mathbf{x})$ of N -length output characters sequence $\mathbf{y} = (y_1, \dots, y_N)$. To compute $P_{\text{CTC}}(\mathbf{y}|\mathbf{x})$, CTC introduces framewise letter sequence with an additional ‘blank’ symbol $\mathbf{z} = (z_1, \dots, z_T)$ and factorizes $P_{\text{CTC}}(\mathbf{y}|\mathbf{x})$ using conditional independence assumption as follows:

$$P_{\text{CTC}}(\mathbf{y}|\mathbf{x}) \approx \sum_{\mathbf{z}} \prod_t P(z_t | z_{t-1}, \mathbf{y}) P(z_t | \mathbf{x}) P(\mathbf{y}) \quad (5.2)$$

where three distribution components are: state transition probability $P(z_t | z_{t-1}, \mathbf{y})$, framewise posterior distribution $P(z_t | \mathbf{x})$ and character-level language model $P(\mathbf{y})$. The state transition probability enforces the monotonic alignment between speech

and character sequences, and is obtained using the set of pre-defined rules (see Eq. (21) in [48]), whereas framewise posterior distribution is modeled as follows:

$$P(z_t|\mathbf{x}) = \text{Softmax}(\text{Lin}(\mathbf{h})) \quad (5.3)$$

where $\text{Lin}(\cdot)$ is a linear projection layer with learnable matrix and bias parameters. Lastly, the Eq. (5.2) is efficiently computed using dynamic programming. The CTC loss is defined as a negative log-likelihood of the ground truth sequences $\mathbf{y}^*$:

$$\mathcal{L}_{\text{CTC}} = -\log P_{\text{CTC}}(\mathbf{y}^*|\mathbf{x}) \quad (5.4)$$

Attention-based decoder module. The attention-based decoder computes the probability distribution $P_{\text{ATT}}(\mathbf{y}|\mathbf{x})$ over the output token sequence $\mathbf{y}$ given the previously emitted tokens $y_{<n}$ and input feature sequence $\mathbf{x}$ using the chain rule:

$$\alpha_n = \text{Attention}(s_{n-1}, \alpha_{n-1}, \mathbf{h}) \quad (5.5)$$

$$c_n = \sum_{j=1}^L \alpha_{n,j} h_j \quad (5.6)$$

$$s_n = \text{LSTM}(s_{n-1}, c_n, \text{InputProj}(y_{n-1})) \quad (5.7)$$

$$P(y_n|y_{<n}, \mathbf{x}) = \text{Softmax}(\text{OutputProj}(s_n)) \quad (5.8)$$

$$P_{\text{ATT}}(\mathbf{y}|\mathbf{x}) = \prod_n P(y_n|y_{<n}, \mathbf{x}) \quad (5.9)$$

where α_n is an attention weight vector produced by $\text{Attention}(\cdot)$ module, c_n is a context vector which encapsulates the information in the input speech features required to generate the next token, s_n is a hidden state produced by unidirectional long short-term memory (LSTM). $\text{InputProj}(\cdot)$ and $\text{OutputProj}(\cdot)$ are input and output linear projection layers with learnable matrix parameters, respectively. The input and output learnable matrices hold input and output embedding representations of tokens, respectively. The loss function of attention-based decoder module is computed using Eq. (5.9) as:

$$\mathcal{L}_{\text{ATT}} = -\log P_{\text{ATT}}(\mathbf{y}^*|\mathbf{x}) \quad (5.10)$$

Finally, the CTC and attention-based decoder modules are jointly trained within multi-task learning (MTL) framework as follows:

$$\mathcal{L}_{\text{MTL}} = \lambda \mathcal{L}_{\text{CTC}} + (1 - \lambda) \mathcal{L}_{\text{ATT}} \quad (5.11)$$

where λ controls the contribution of the losses.

Our proposed method will append additional constraint into the MTL framework which will mainly impact the learnable matrix parameter of $\text{OutputProj}(\cdot)$ layer in Eq. (5.8) as will be explained in the following sections.

Output projection matrix. In E2E-ASR architecture, the output projection matrix, also known as embedding matrix, is responsible for learning embedding vector representations of output tokens (e.g. characters, sub-words or words). During the inference stage, these vector representations are compared against hidden state vector s_n which incorporates both previous context history $y_{<n}$ and acoustic information c_n . The comparison is performed by the means of dot product as follows:

$$\text{OutputProj}(s_n) = W_o s_n + b_o \quad (5.12)$$

where $W_o = [e_1 \ e_2 \ \dots \ e_m]^T$ is an output projection matrix holding embedding vector representations of output tokens and b_o is a bias vector. Subsequently, the output tokens with the embedding vectors closest to hidden state vector s_n will be assigned highest likelihood.

For the code-switching speech recognition, the output projection matrix W_o should learn a multilingual embedding space incorporating several languages. However, if E2E-CS-ASR is trained using only monolingual corpora for each individual language, the output token embedding representations of these languages will diverge from each other, leading to the E2E-CS-ASR which is incapable to switch languages withing single utterance. One potential reason might be the decoder module of E2E-CS-ASR whose current output depends on previously recognized outputs. As such, if decoder has recognized tokens from one language, it will prevent the recognition of tokens from other languages, since during the training stage it hasn't been exposed to such language switching patterns. It can be also explained using Eq. (5.12), let's assume that the E2E-CS-ASR has recognized a several tokens (or words) from one language. Subsequently, the hidden state vector s_n will partially

incorporate this information. Hence, s_n will be closer to the output token embedding representations of the same language since the output token representations of another language are located in disjoint subspace. During the dot product of s_n with the embedding matrix W_o , the output tokens of the same language as previously recognized tokens will be assigned higher likelihood value.

In the following sections, we will propose a method which addresses this problem by forcing the output token representations of monolingual languages to be closer to each other.

5.2 Related Works

In this section, we review the previous works attempted to circumvent the CS labeled data scarcity problem for ASR systems.

5.2.1 Multi-pass Systems

An early approach designed to recognize CS utterances is so-called multi-pass systems [226, 227]. The multi-pass systems don't require any CS data as they are built using only monolingual data. These systems consist of three main stages. In the first stage, the CS utterances are split into monolingual segments using the language boundary detection system. In the second stage, obtained monolingual segments are language labeled using the language identification system. In the last stage, the language labeled segments are decoded using the corresponding monolingual ASR system. The multi-pass systems were shown to be more effective at recognizing a CS speech than simply using a single monolingual ASR corresponding to the matrix (i.e. primary) language.

Although multi-pass systems don't require any CS data, they are prone to error-propagation between different stages. For example, a deficient language boundary detection system in the first stage might ruin the functionality of the following two stages. Furthermore, language boundary detection and language identification systems are considered challenging tasks. To overcome the interdependence between different tasks, the multi-pass systems were replaced by single-pass systems which purely rely on the availability of labeled CS data [228].

5.2.2 Multilingual Systems

The multilingual systems are constructed by simply combining several monolingual corpora to train a single ASR system. Recently, several works have implemented multilingual E2E ASR systems using attention-based encoder-decoder model [220, 229, 230]. While being effective at recognizing intersentential CS utterances, they have been shown ineffective against intrasentential case [220]. One potential reason might be the decoder module whose current output depends on previously recognized outputs. Particularly, when the decoder has recognized tokens from one language, it will prevent recognition of tokens from another language, since its internal memory units haven't been exposed to such language switching patterns during the training stage [220]. To discard the dependency between output tokens, [210] explored the CTC based E2E model. Unfortunately, CTC based multilingual systems achieve insignificant improvements.

5.2.3 Semi-supervised Learning

The semi-supervised approaches have been also explored to improve CS speech recognition capability of ASR systems [231–233]. In these approaches, available seed CS data is first used to build baseline CS-ASR system. The baseline system is then used to transcribe a raw CS speech. Next, the seed and most reliably transcribed CS data are combined to re-build the CS-ASR system. On the account of increased CS train data, the semi-supervised approaches can achieve significant improvements. However, these approaches strongly rely on the availability of seed CS data and highly sensitive to the quality of transcribed CS data used to rebuild the CS-ASR.

5.2.4 Transfer Learning

Another effective solution to deal with the CS data scarcity is to employ transfer learning which have been extensively used in E2E-CS-ASR [210–212]. In transfer learning, the baseline CS-ASR is first pre-trained using abundant monolingual data from different languages similar to multilingual systems. Next, the pre-trained multilingual system is fine-tuned using the CS data. Another slightly different

variant of transfer learning is to pre-train using combined monolingual and CS data, and then fine-tune with CS only data [211]. This variant will prevent the pre-training stage from overfitting to the monolingual patterns. In both variants, the pre-training stage offers good parameter initialization and increased amount of training data which in general improves the recognition performance of ASR systems [234]. However, the fine-tuning stage requires adequate amount of CS data which is extremely rare in real-world applications.

5.2.5 Data Synthesis

Data synthesis techniques have been also explored to generate more CS data [223, 224, 235, 236]. For example, [224, 235] employed machine translation system to generate artificial CS data by translating segments of monolingual sentences. Similarly, [223] also resort to machine translation system, however, they impose additional linguistic constraints to generate grammatically correct CS data. On the other hand, [236] employed generative models and dictionary. However, so far these techniques are only applied to generate textual CS data which can be used to improve CS language models. One potential extension is to convert them into CS speech data using text-to-speech (TTS) system, similar to the back-translation trick employed in machine translation [237]. However, building robust CS-TTS is another challenging problem to solve [238].

5.3 Constraining Output Embedding Matrix

The labeled CS speech data is extremely rare, and thus, insufficient to build robust E2E-CS-ASR systems. A workaround is multilingual E2E-CS-ASR systems built using only monolingual data. While being effective at handling the intersentential CS utterances, the multilingual E2E-CS-ASR systems are impractical against intrasentential case [220]. One of the main reasons is the absence of within utterance language switching patterns in the training data. Consequently, the output embedding matrix of E2E-CS-ASR fails to learn cross-lingual relationships. Therefore, the linguistic representations of different languages are mapped into disjoint latent spaces which hinders the language mixing within single utterance.

In other words, let's assume we are given multilingual E2E-CS-ASR built using two monolingual corpora with the following output token embedding matrix W_o from the Eq. (5.12):

$$W_o = [e_1^{l1} \ e_2^{l1} \ \dots \ e_m^{l1} \ e_{m+1}^{l2} \ e_{m+2}^{l2} \ \dots \ e_{m+n}^{l2}]^T \quad (5.13)$$

The matrix W_o is comprised of $m + n$ embedding vectors e where first m vectors correspond to output tokens of language $l1$, while the last n vectors correspond to output tokens of second language $l2$. During the training stage, these embedding vectors will form two disjoint clusters corresponding to two languages, because in the training utterances embedding vectors of $l1$ and $l2$ are never mixed. Consequently, the likelihood of e_i^{l1} to follow e_j^{l2} and vice versa will be extremely low, preventing the code-switching.

To improve the code-switching capability of multilingual E2E-CS-ASR, we propose an effective method which forces those disjoint latent spaces to be close to each other. Specifically, our method constrains the output token embeddings of monolingual languages within matrix W_o to be similar, and thus, facilitates the language switch within single utterance. The proposed method is inspired by [239, 240] where intermediate feature representations of text and speech are forced to be close to each other. Additionally, the imposed constraints will act as a regularization term to prevent overfitting. As such, our method will contribute to the improvement of overall recognition performance for both monolingual and CS utterances. Note that our method is fully unsupervised and doesn't require any CS data which is different from previously proposed approaches relying on labeled CS data [167, 210–212] or other cross-lingual resources such as word aligned parallel corpus [223] and dictionary [224]. Experimental results demonstrate high effectiveness of the proposed method, achieving significant improvements on Mandarin-English CS speech recognition task.

5.3.1 Constraint Types

To bridge the gap between disjoint latent spaces, we examined two constraint types based on Jensen-Shannon divergence (JSD) and cosine distance (CD). These constraints will typically act as an unsupervised cross-lingual signal source forcing the output token embedding representations of monolingual languages to be similar.

Specifically, JSD will enforce the output token embeddings of monolingual languages to possess similar distributions. On the other hand, CD will enforce the centroids of two distributions to be close to each other. Note that our proposed method is not restricted by these two constraint types, other similarity and distance metrics can be also used.

5.3.1.1 Jensen-Shannon Divergence

The JSD is a method to measure difference between two probability distributions. Therefore, we first assume that given two feature clusters C_1 and C_2 following a z -dimensional multivariate Gaussian distribution:

$$C_1 \sim \mathcal{N}_z(\mu_1, \Sigma_1) \quad (5.14)$$

$$C_2 \sim \mathcal{N}_z(\mu_2, \Sigma_2) \quad (5.15)$$

where μ is a mean and Σ is a covariance matrix. For example, in two language code-switching scenario, the two clusters will correspond to output token embeddings of these two languages, i.e. e^{l1} and e^{l2} . The JSD between these distributions is then computed as:

$$\begin{aligned} \mathcal{L}_{\text{JSD}} = & \text{trace}(\Sigma_1^{-1}\Sigma_2 + \Sigma_1\Sigma_2^{-1}) \\ & + (\mu_1 - \mu_2)^T(\Sigma_1^{-1} + \Sigma_2^{-1})(\mu_1 - \mu_2) - 2z \end{aligned} \quad (5.16)$$

where z is a dimension of output token embedding vector e . The μ_1 and Σ_1 are derived from the output token embedding vectors in language $l1$, whereas μ_2 and Σ_2 are derived from language $l2$. Lastly, we fuse the JSD constraint with the loss function of E2E-CS-ASR from Eq. (5.11) as follows:

$$\mathcal{L}_{\text{MTL}} = \lambda\mathcal{L}_{\text{CTC}} + (1 - \lambda)(\alpha\mathcal{L}_{\text{ATT}} + (1 - \alpha)\mathcal{L}_{\text{JSD}}) \quad (5.17)$$

where $\alpha \in [0, 1]$ controls the contribution of the constraint.

5.3.1.2 Cosine Distance

The CD is a method used to measure the cosine of the angle between two vectors. Here, we also assume existence of two clusters C_1 and C_2 corresponding to the

output token embeddings of two monolingual languages with mean vectors (or centroids) μ_1 and μ_2 , respectively. The CD based constraint between these two mean vectors is computed as follows:

$$\mathcal{L}_{\text{CD}} = 1 - \frac{\mu_1 \cdot \mu_2}{\|\mu_1\| \|\mu_2\|} \quad (5.18)$$

where $\cdot$ is dot product and $\|\cdot\|$ is Euclidean norm.

The CD constraint is integrated into the multi-task learning framework of E2E-CS-ASR in a similar way as Eq. (5.17):

$$\mathcal{L}_{\text{MTL}} = \lambda \mathcal{L}_{\text{CTC}} + (1 - \lambda)(\alpha \mathcal{L}_{\text{ATT}} + (1 - \alpha) \mathcal{L}_{\text{CD}}) \quad (5.19)$$

5.4 Experiment

In this section, we explain experiments conducted to evaluate proposed method of constraining output token embeddings. We first describe the experimental setup followed by the demonstration of obtained results.

5.4.1 Experiment Setup

In this subsection, we describe the experimental setup including the dataset, employed toolkits and baseline E2E-CS-ASR system configurations.

5.4.1.1 Dataset

We evaluate our method on Mandarin-English code-switching language pair from the SEAME [213] corpus. The characteristics of the SEAME corpus are reported in Table 5.1. We used standard data splitting² on par with previous works [167, 208, 231] which consists of three sets: train, test_{man} and test_{eng}. To match the no CS data scenario, where we assume that we only possess monolingual training data, we removed all intrasentential CS utterances from the train set. The test_{man} and test_{eng} sets were used for evaluation. Both evaluation sets are gender balanced

²<https://github.com/zengzp0912/SEAME-dev-set>

and consist of 10 speakers, but the matrix³ language of speakers is different, i.e. Mandarin for test_{man} and English⁴ for test_{eng} .

	train		test_{man}	test_{eng}
	Man	Eng		
# tokens	~216k	~109k	~96k	~54k
# utterances	21,476	17,925	6,531	5,321
(# CS utterances)	(0)	(0)	(4,418)	(2,652)
Duration	15.8 hr	11.8 hr	7.5 hr	3.9 hr

TABLE 5.1: SEAME dataset statistics after removing the CS utterances from the train set. ‘Man’ and ‘Eng’ refer to Mandarin and English languages, respectively.

5.4.1.2 Baseline E2E-CS-ASR configuration

To train our baseline E2E-CS-ASR model, we used the ESPnet toolkit [51] which is considered the state-of-the-art for building E2E ASR systems. The baseline E2E-CS-ASR model is based on hybrid CTC/Attention architecture [48]. We used 80 mel-scale filterbank coefficients with pitch as input features. The encoder module consists of first two layers of VGG network followed by six BLSTM layers each with 512 units. The attention-based decoder module consists of a single LSTM layer with 512 units and employs multi-headed hybrid attention mechanism [241] with 4 heads. The CTC module consists of a single linear layer with 512 units and its interpolation weight in Eq. (5.11) was set to 0.2. The network was optimized using Adadelta with gradient clipping. During the decoding stage, the beam size was set to 30. The baseline model achieves 34.3% and 46.3% mixed error rates (MER)⁵ on test_{man} and test_{eng} respectively, when trained on the entire SEAME train set including the CS utterances.

5.4.2 Experiment Results and Analysis

In this subsection, we describe obtained results followed by the analysis and discussion of important findings.

³The dominant language into which elements from the embedded language are inserted.

⁴Here, English refers to Singaporean English.

⁵The term “mixed” refers to different token units used for English (words) and Mandarin (characters).

5.4.2.1 Results

The experiment results are reported in Table 5.2. We split the test sets into monolingual and CS utterances to evaluate the impact of the proposed method by analyzing them separately. We first report the MER performance of a conventional hybrid DNN-HMM ASR model built using Kaldi toolkit [35] (row 1). The DNN-HMM model was build using the ‘nnet3+chain’ setup of Kaldi including the latest developments, the detailed model specifications can be found in [231]. As you can see, the performance gap between hybrid and E2E ASR architectures in low-resource settings is still very large. However, the E2E ASR systems are much easier to train as its components are jointly optimized under single network. In addition, the E2E ASR doesn’t rely on phonetic lexicons carefully built by linguists. Importantly, the E2E ASR systems don’t struggle from the out-of-vocabulary problem. Moreover, the performance of E2E ASR is consistently catching up the hybrid systems in low-resource settings and already achieve comparable results in large-resource settings [178].

The MER performance of the baseline E2E-CS-ASR model is shown in the second row. We followed the recent trends [167, 211, 212] to obtain a much stronger baseline model. Specifically, we applied speed perturbation (SP) based data augmentation technique [242] to increase the training data size (row 3). In addition, we used byte pair encoding (BPE) based subword units [243] to balance Mandarin and English tokens (row 4). We tried different vocabulary sizes for BPE and found $4k$ units to work best in our case. The application of SP⁶ and BPE strengthened the baseline system by 21.1% and 19.4% MER on test_{man} and test_{eng} , respectively.

The performance of models employing proposed CD and JSD constraints are shown in rows 5 and 6. The interpolation weights for CD and JSD in Eq. (5.19) and (5.17) are estimated empirically where the best results are achieved at 0.9 and 0.97 respectively. Both constraints gain considerable MER improvements over the strong baseline system with SP and BPE (row 4). Notably, we observed that CD constraint is more effective on monolingual utterances, whereas JSD constraint is more effective on CS utterances. To complement the advantages of both constraints, we

⁶Note that the DNN-HMM ASR is also built using speed perturbation.

No.	Model	test _{man}			test _{eng}		
		mono utts.	CS utts.	all	mono utts.	CS utts.	all
1	DNN-HMM	-	-	39.1	-	-	45.2
2	Baseline	57.7	73.3	70.6	73.7	80.6	78.3
3	+ SP	39.4	56.0	53.2	54.2	65.9	62.2
4	+ BPE	38.1	51.8	49.5	52.9	61.4	58.9
5	+ CD	34.4	49.0	46.3	47.2	58.5	55.1
6	+ JSD	34.9	48.8	46.3	47.8	57.6	54.6
7	+ CD	34.0	48.1	45.6	47.2	57.4	54.4

TABLE 5.2: The MER (%) performance of different ASR models built using monolingual data. The test sets are further split into monolingual (mono) and code-switching (CS) utterances.

combined them as follows:

$$\mathcal{L}_{\text{MTL}} = \lambda \mathcal{L}_{\text{CTC}} + (1 - \lambda)(\alpha \mathcal{L}_{\text{ATT}} + (1 - \alpha)(\beta \mathcal{L}_{\text{JSD}} + (1 - \beta) \mathcal{L}_{\text{CD}})) \quad (5.20)$$

where α and β are set to 0.05 and 0.9 respectively. The combination of two constraints significantly improves the MER over the strong baseline model by 3.9% and 4.5% on test_{man} and test_{eng} respectively (row 7). Interestingly, the MER improvement is achieved on both monolingual and CS utterances. We presume one of the reasons might be the regularization effect of constraints preventing from overfitting. These results suggest that the proposed method of constraining the output token embeddings is effective.

5.4.2.2 Analysis

Changing the interpolation weight. We repeat the experiment with different interpolation weights for CD and JSD constraints (hyperparameter α in Eq. (5.19) and (5.17)) to investigate its effect on MER performance. Figure 5.2 shows that the proposed method consistently improves the MER over the strong baseline model with SP and BPE for interpolation weights in range 0.5-0.99. Particularly, the best results are achieved for interpolation weights in range 0.8-0.99, while in range 0-0.5 insignificant or no improvement is achieved. Therefore, we recommend to set the interpolation weights of constraints higher than 0.5 to increase its contribution within MTL loss.

FIGURE 5.2: The impact of CD and JSD constraint interpolation weights on MER performance for test_{eng} (red/top) and test_{man} (blue/bottom) sets.

Visualization of shared output token embedding space. To gain insights from the effects of the proposed method on the shared output embedding space represented by matrix W_o , we visualize it using dimensionality reduction technique based on the principal component analysis (PCA). Figure 5.3 shows the output token embedding vectors of English and Mandarin languages reduced to two dimensional space. Figure 5.3a depicts the output tokens learned without constraints, whereas Figures 5.3b, 5.3c, 5.3d depicts the output tokens learned with the proposed constraints. Note that the learned output token embeddings of monolingual languages strongly diverge from each other when proposed constraints are not employed. On the other hand, imposition of constraints enforces the output token embeddings to be close to each other. Interestingly, we can observe that CD based constraint simply enforces the centroids of the clusters to be closer, while the distribution of the clusters are dissimilar. On the other side, JSD based constraint enforces both centroids and distributions of the clusters to be alike. Visualization of the shared output embedding space confirms that our method is effective at bridging the gap between output token embeddings of monolingual languages.

Applying language model. The state-of-the-art results in ASR are usually obtained by employing a language model (LM) [72, 79, 169]. To examine whether proposed constraints are complementary with LM, we employed LM during the decoding stage. The LM was trained on the entire SEAME train set, including CS utterances, as a single layer LSTM with 512 units and was integrated using shallow fusion technique [244]. In this experiment, we tried different LM interpolation weights changed with a step size of 0.025 and report only the best results (see

FIGURE 5.3: PCA visualization of shared output token embedding space without (a) and with (b,c,d) proposed constraints.

Table 5.3). The LM improves the strong baseline model by 1.0% and 0.5% relative MER for test_{man} and test_{eng} , respectively. On the other hand, the LM improves the strong baseline model with the proposed constraints by 1.3% and 1.2% relative MER for test_{man} and test_{eng} , respectively. Obtained MER improvements suggest that proposed constraints and LM complement each other. Moreover, the proposed method benefits from the LM more than the strong baseline model.

Model	Decoder LM	MER (%)	
		test_{man}	test_{eng}
Baseline + SP + BPE	No	49.5	58.9
	Yes	49.0	58.6
Baseline + SP + BPE + CD & JSD	No	45.6	54.4
	Yes	45.0	53.7

TABLE 5.3: The MER performance after applying the language model during the decoding stages.

5.5 Summary

In this chapter, we address the labeled code-switching data scarcity problem for ASR task. Therefore, we proposed a novel method to improve the code-switching capability of multilingual E2E-CS-ASR model built using only monolingual data. Specifically, our method constrains the output token embeddings of monolingual languages to force them to be similar, and hence, enable E2E-CS-ASR to easily switch between languages. We examined Jensen-Shannon divergence and cosine distance based constraints which are incorporated into the multi-task learning framework of the E2E-CS-ASR. We evaluated the proposed method on Mandarin-English CS language pair from the SEAME corpus where CS utterances were removed from the training set. The proposed method outperforms the strong baseline model by a large margin, i.e. absolute 3.9% and 4.5% MER improvements on test_{man} and test_{eng} respectively. The visualization of the shared output embedding space confirms the effectiveness of the proposed method. In addition, our method is complementary with the language model where further MER improvement is achieved. Importantly, all these improvements are achieved without using any additional linguistic resources such as word-aligned parallel corpus or language identification system. We believe that the proposed method can be easily adapted to other scenarios and benefit other CS language pairs.

For the future work, we plan to test the proposed method on scenarios with a larger amount of monolingual data and examine its effectiveness on E2E-CS-ASR models trained using CS data. We also plan to study the effects of the proposed method in transfer learning approach where it will be used to pre-train the model with external monolingual data.

Chapter 6

Conclusions and Future Work

The LM is a vital component of ASR systems responsible for the generation of linguistically correct output hypotheses. However, LMs are sensitive to the domain mismatch between training and test data which might lead to suboptimal performances. Therefore, in this work, we focused on improving the effectiveness of LM in ASR systems. Particularly, we addressed the LM domain mismatch problem caused by the following three factors: topic (Chapter 3), vocabulary coverage (Chapter 4) and code-switching practice (Chapter 5). To diminish the impact of these factors on ASR performance, we proposed three novel LM adaptation techniques each addressing one of the factors.

Topic adaptation. In this work, we proposed a novel two-pass LM adaptation framework based on data selection. Particularly, our method employs the recognized hypotheses from the first pass and data selection techniques to reselect in-domain data from the background training corpus. The reselected in-domain data is then used to build new LM which is employed during the second pass. Our motivation is instead of modifying the parameters of generic LM, we suggest to modify the background training corpus by filtering out the irrelevant sentences. The adapted LM constructed by our approach is expected to better fit the target domain and to represent better linguistic knowledge as it was built on judiciously selected complete sentences. The proposed framework was implemented using the Kaldi speech recognition toolkit and evaluated on the 11 talks from the TED-LIUM speech corpus covering different topics such as green technology, medicine and others. In performed experiments, the LM adapted using our approach reduced the

perplexity by up to 47% relatively when evaluated on the reference data. Employing it to redecode the input test utterances reduced the word error rate (WER) by up to 10% relatively. Interestingly, such improvements were achieved by selecting only 1% - 10% of the data from the background corpus.

The potential of the proposed data selection based LM adaptation approach is not fully exploited yet. We believe that further research in this field will yield fruitful results. Particularly, to select data, our method employs features based on the surface form of the words, whereas more effective feature representations based on the semantic information such as latent semantic analysis [121] haven't been exploited. Another important research focus would be the optimization of the adaptation time, because currently our method is based on the two-pass approach which is computationally expensive and inapplicable for a real-time speech recognition task. Additionally, the selection criterion of our method is computed locally at the short text segment level, e.g. sentence or paragraph, whereas the overall relevance of the selected data is ignored. Therefore, we believe that our method should be extended such that the overall similarity metric of selected data is taken into the account. Lastly, other potential research direction would be the application of the data selection techniques to adapt an internal LM of end-to-end ASR systems.

Vocabulary coverage adaptation. In this work, we addressed the limitations of RNN LM vocabulary coverage by proposing an efficient method to enrich the vector representations of rare words in pre-trained RNN LMs. Additionally, the proposed method can be used to expand the vocabulary coverage. To achieve this, the proposed method augments the word embedding matrices of pre-trained RNN LM, while keeping other parameters unchanged. Specifically, our method shifts the rare word representation towards its semantic landmark in the embedding space using representations of other semantically and syntactically similar words. One of the important features of proposed method is that it doesn't require any additional in-domain data.

We evaluated the proposed method on two different ASR experiments. First, we evaluated its vocabulary expansion capability where a new word is added to the vocabulary. In this experiment, we rescored the N-best list output from the state-of-the-art TED talks ASR system. The experimental results show that the vocabulary expanded RNN LM achieves 4% relative WER improvement over the conventional

RNN LM. Moreover, 7% relative WER improvement is achieved over the strong Kneser-Ney smoothed 5-gram model used to rescore the word lattice. Such improvements suggest that the proposed vocabulary expansion method is effective.

We next evaluated the rare word representation enrichment capability of our method. In this experiment, we applied our method to first enrich the rare Singapore street names in pre-trained RNN LM and then used it to rescore the 100-best list generated by the state-of-the-art Singapore English ASR system. The enriched RNN LM achieved 6% relative WER improvement over the baseline RNN LM. In comparison to other strong baseline LMs, our method achieved significant WER improvements, i.e. 6% and 16% improvement over Kaldi-RNNLM and Kneser-Ney smoothed 4-gram LM, respectively. Moreover, the enriched RNN LM increased the recognition accuracy of rare street names by 16% absolute. These results confirm the effectiveness of the proposed method.

We believe that our approach of augmenting input and output word embedding matrices is general enough to be applied to other neural network models with similar architecture and be easily adapted to other scenarios. For example, one potential direction would be to extend our method to enrich the vocabulary coverage of word-level end-to-end ASR systems. Furthermore, it can be used to resolve the OOV problems in many natural language processing (NLP) applications such as machine translation, summarizing, question answering and so on.

Code-switching practice adaptation. In this work, we attempted to build an E2E-CS-ASR using only monolingual data without any form of cross-lingual resource. The only assumption we made is an availability of monolingual speech corpus for each of the CS languages. This setup is important and common to many low-resource CS languages, but has not received much research attention. Besides, it will serve as a strong baseline performance that any system trained on CS data should reach. Notice that our approach can be interpreted as a code-switching practice adaptation from intersentential to intrasentential type.

However, the straightforward training of E2E-CS-ASR using only monolingual data is ineffective as have been shown in [220]. One of the main reasons have been reported to be the structure of E2E-CS-ASR, especially the encode-decoder architecture, where output token embeddings are conditioned on previously recognized tokens. Given that the training data consists of only monolingual utterances, the

E2E-CS-ASR will fail to learn cross-lingual relations between monolingual languages, and therefore restrict the language switches within single utterance. To further understand the reason, we systematically examined different layers of E2E-CS-ASR by visualizing learned latent spaces. Interestingly, we noticed that in the final projection layer, the output token embeddings of two monolingual languages are differently distributed and located apart from each other. We hypothesized that the difference between these two monolingual output token embeddings restricts the E2E-CS-ASR model from switching languages.

To verify our hypothesis, in this work, we proposed to impose additional constraints which will enforce the output token embeddings of monolingual languages to be similar. Specifically, we proposed to use Jensen-Shannon divergence and cosine distance based constraints. The former constraint will enforce output token embeddings of monolingual languages to possess similar distributions, while the later constraint simply brings the centroids of two distributions closer to each other. In addition, the imposed constraints will act as a regularization term to prevent overfitting. We evaluated the proposed method on Mandarin-English CS language pair from the SEAME corpus where CS utterances were removed from the train set. The proposed method outperforms the strong baseline model by a large margin, i.e. absolute 3.9% and 4.5% mixed error rate (MER) improvements on test_{man} and test_{eng} , respectively. The visualization of the shared output embedding space confirms the effectiveness of the proposed method. In addition, our method is complementary with the language model where further MER improvement was achieved. Importantly, all these improvements were achieved without using any additional linguistic resources such as word aligned parallel corpus or language identification system. We believe that the proposed method can be easily adapted to other scenarios and benefit other CS language pairs.

For the future work, we plan to analyze the effectiveness of the proposed method on scenarios with a larger amount of monolingual data. In addition, we plan to explore its usefulness for the E2E-CS-ASR models trained using labeled CS data. Another possible application of the proposed method would be in transfer learning where it can be used to pre-train stronger E2E-CS-ASR model using only monolingual data.

List of Author’s Awards, Patents, and Publications

Conference Proceedings

- **Yerbolat Khassanov**, Zhiping Zeng, Van Tung Pham, Haihua Xu and Eng Siong Chng, “Enriching Rare Word Representations in Neural Language Models by Embedding Matrix Augmentation,” in *INTERSPEECH*, 2019.
- **Yerbolat Khassanov**, Haihua Xu, Van Tung Pham, Zhiping Zeng, Eng Siong Chng, Chongjia Ni and Bin Ma, “Constrained Output Embeddings for End-to-End Code-Switching Speech Recognition with Only Monolingual Data,” in *INTERSPEECH*, 2019.
- Zhiping Zeng, **Yerbolat Khassanov**, Van Tung Pham, Haihua Xu, Eng Siong Chng and Haizhou Li, “On the End-to-End Solution to Mandarin-English Code-switching Speech Recognition,” in *INTERSPEECH*, 2019.
- **Yerbolat Khassanov** and Eng Siong Chng, “Unsupervised and Efficient Vocabulary Expansion for Recurrent Neural Network Language Models in ASR,” in *INTERSPEECH*, 2018.
- **Yerbolat Khassanov**, Tze Yuang Chong, Benjamin Bigot and Eng Siong Chng, “Unsupervised Language Model Adaptation by Data Selection for Speech Recognition,” in *ACIIDS*, 2017.

Bibliography

- [1] Dong Yu and Li Deng. *Automatic speech recognition*. Springer, 2016. [1](#), [12](#)
- [2] KH Davis, R Biddulph, and Stephen Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952. [1](#)
- [3] Bruce T Lowerre. The HARPY speech recognition system. Technical report, 1976. [2](#)
- [4] Frederick Jelinek, Lalit Bahl, and Robert Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21(3):250–256, 1975. [2](#)
- [5] Joshua T Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434, 2001. [2](#), [16](#), [17](#), [18](#), [28](#), [35](#)
- [6] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000. [2](#), [16](#), [22](#), [40](#)
- [7] Jerome R Bellegarda. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108, 2004. [3](#), [22](#), [24](#), [26](#), [40](#)
- [8] Salil Deena, Madina Hasan, Mortaza Doulaty, Oscar Saz, and Thomas Hain. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition and alignment. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 27(3):572–582, 2019. [3](#), [35](#)
- [9] Roland Kuhn and Renato De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990. [3](#), [28](#), [40](#)

- [10] Kristie Seymore and Ronald Rosenfeld. Using story topics for language model adaptation. In *5th European Conference on Speech Communication and Technology*, 1997. [3](#), [29](#), [40](#)
- [11] Salil Deena, Madina Hasan, Mortaza Doulaty, Oscar Saz, and Thomas Hain. Combining feature and model-based adaptation of rnnlms for multi-genre broadcast speech recognition. In *INTERSPEECH*, pages 2343–2347. ISCA, 2016. [3](#), [31](#), [33](#), [34](#), [35](#)
- [12] Siva Reddy Gangireddy, Pawel Swietojanski, Peter Bell, and Steve Renals. Unsupervised adaptation of recurrent neural network language models. In *INTERSPEECH*, pages 2333–2337. ISCA, 2016. [3](#), [30](#), [31](#), [32](#)
- [13] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE, 2012. [3](#), [33](#)
- [14] Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark JF Gales, and Philip C Woodland. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In *INTERSPEECH*. ISCA, 2015. [3](#), [33](#), [71](#)
- [15] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. "Your word is my command": Google search by voice: A case study. In *Advances in Speech Recognition*, pages 61–90. Springer, 2010. [5](#), [69](#)
- [16] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 20(1):14–22, 2011. [10](#)
- [17] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 20(1): 30–42, 2011.
- [18] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29, 2012.

- [19] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8599–8603. IEEE, 2013.
- [20] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at Microsoft. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8604–8608. IEEE, 2013. [10](#)
- [21] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 369–376. ACM, 2006. [10](#), [13](#), [91](#)
- [22] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.
- [23] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014. [91](#)
- [24] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.
- [25] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602*, 2014. [10](#), [13](#)
- [26] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. [10](#)

- [27] Hynek Hermansky. Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990. [10](#)
- [28] Sadaoki Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(2):254–272, 1981. [11](#)
- [29] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. [11](#)
- [30] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997. [11](#)
- [31] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002. [11](#), [12](#)
- [32] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008. [12](#)
- [33] Mehryar Mohri. Weighted automata algorithms. In *Handbook of Weighted Automata*, pages 213–254. Springer, 2009. [12](#)
- [34] Abdel-rahman Mohamed, Geoffrey Hinton, and Gerald Penn. Understanding how deep belief networks perform acoustic modelling. *Neural Networks*, pages 6–9, 2012. [12](#)
- [35] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The Kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011. [12](#), [55](#), [56](#), [78](#), [81](#), [102](#)
- [36] David Rybach, Christian Gollan, Georg Heigold, Björn Hoffmeister, Jonas Löf, Ralf Schlüter, and Hermann Ney. The RWTH Aachen University open source speech recognition system. In *10th Annual Conference of the International Speech Communication Association*, 2009. [12](#)

- [37] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The HTK book. *Cambridge University Engineering Department*, 3:175, 2002. [12](#)
- [38] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. *Technical Report SML1 TR2004-0811*, 2004. [12](#)
- [39] Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius - An open source real-time large vocabulary recognition engine. In *7th European Conference on Speech Communication and Technology*, 2001. [12](#)
- [40] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012. [13](#)
- [41] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE, 2017. [13](#)
- [42] Hasim Sak, Matt Shannon, Kanishka Rao, and Françoise Beaufays. Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. In *INTERSPEECH*, pages 1298–1302. ISCA, 2017. [13](#)
- [43] Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems*, pages 5067–5075, 2016. [13](#)
- [44] Tara N Sainath, Chung-Cheng Chiu, Rohit Prabhavalkar, Anjuli Kannan, Yonghui Wu, Patrick Nguyen, and ZhiJeng Chen. Improving the performance of online neural transducer models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5864–5868. IEEE, 2018. [13](#)

- [45] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016. [13](#), [65](#), [91](#)
- [46] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016. [13](#), [65](#)
- [47] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839. IEEE, 2017. [13](#)
- [48] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017. [13](#), [15](#), [91](#), [93](#), [101](#)
- [49] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. *arXiv preprint arXiv:1706.02737*, 2017. [91](#)
- [50] Takaaki Hori, Shinji Watanabe, and John Hershey. Joint CTC/attention decoding for end-to-end speech recognition. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 518–529, 2017. [13](#)
- [51] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplín, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. Espnet: End-to-end speech processing toolkit. In *INTERSPEECH*, pages 2207–2211. ISCA, 2018. [15](#), [101](#)
- [52] Patrick Doetsch, Albert Zeyer, Paul Voigtlaender, Ilia Kulikov, Ralf Schlüter, and Hermann Ney. RETURNN: The RWTH extensible training framework for universal recurrent neural networks. In *IEEE International Conference on*

- Acoustics, Speech and Signal Processing (ICASSP)*, pages 5345–5349. IEEE, 2017. [15](#)
- [53] Ronan Collobert, Christian Puhersch, and Gabriel Synnaeve. Wav2letter: An end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016. [15](#)
- [54] Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESSEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE, 2015. [15](#)
- [55] Dan Jurafsky. *Speech & Language Processing*. Pearson Education India, 2000. [16](#)
- [56] Jeff A Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 4–6. Association for Computational Linguistics, 2003. [17](#), [34](#)
- [57] Andrei Alexandrescu and Katrin Kirchhoff. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 1–4. Association for Computational Linguistics, 2006. [17](#), [34](#), [72](#)
- [58] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992. [17](#), [34](#)
- [59] Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modelling. In *3rd European Conference on Speech Communication and Technology*, 1993. [17](#), [34](#)
- [60] Xuedong Huang, Fileno Allewa, Hsiao-Wuen Hon, Mei-Yuh Hwang, Kai-Fu Lee, and Ronald Rosenfeld. The SPHINX-II speech recognition system: An overview. *Computer Speech & Language*, 7(2):137–148, 1993. [17](#)
- [61] Michael Simons, Hermann Ney, and Sven C Martin. Distant bigram language modelling using maximum entropy. In *IEEE International Conference on*

- Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 787–790. IEEE, 1997. [17](#)
- [62] Holger Schwenk and Jean-Luc Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–765. IEEE, 2002. [17](#)
- [63] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003. [18](#), [64](#), [65](#)
- [64] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007. [17](#), [68](#)
- [65] Ronald Rosenfeld. Adaptive statistical language modeling: A maximum entropy approach. Technical report, 1994. [17](#)
- [66] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language*, 10(3):187–228, 1996. [17](#), [22](#), [26](#)
- [67] Lalit R Bahl, Peter F Brown, Peter V de Souza, and Robert L Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):1001–1008, 1989. [17](#)
- [68] Peng Xu and Frederick Jelinek. Random forests in language modelin. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 325–332, 2004. [17](#)
- [69] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999. [18](#), [56](#), [78](#)
- [70] Jerome R Bellegarda. Exploiting both local and global constraints for multi-span statistical language modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 677–680. IEEE, 1998. [18](#)

- [71] Jerome R Bellegarda. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296, 2000. [18](#)
- [72] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association*, 2010. [18](#), [19](#), [64](#), [65](#), [83](#), [104](#)
- [73] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE, 2011. [18](#), [71](#)
- [74] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013. [18](#), [66](#)
- [75] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013. [18](#), [66](#), [75](#)
- [76] Gwénolé Lecorvé and Petr Motlicek. Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition. In *13th Annual Conference of the International Speech Communication Association*, 2012. [19](#)
- [77] Ebru Arisoy, Stanley F Chen, Bhuvana Ramabhadran, and Abhinav Sethy. Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):184–192, 2013. [19](#)
- [78] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *Proc. Interspeech 2018*, pages 387–391, 2018. [20](#), [67](#)
- [79] Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N Sainath, and Karen Livescu. A comparison of techniques for language model integration in encoder-decoder speech recognition. In *IEEE*

- Spoken Language Technology Workshop (SLT)*, pages 369–375. IEEE, 2018. [104](#)
- [80] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, ZhiJeng Chen, and Rohit Prabhavalkar. An analysis of incorporating an external language model into a sequence-to-sequence model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5828. IEEE, 2018. [20](#)
- [81] Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity - A measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977. [20](#)
- [82] Dietrich Klakow and Jochen Peters. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28, 2002. [21](#)
- [83] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009. [21](#)
- [84] Phil C Woodland. Speaker adaptation for continuous density HMMs: A review. In *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, 2001. [22](#)
- [85] Shigeki Sagayama, Koichi Shinoda, Mitsuru Nakai, and Hiroshi Shimodaira. Analytic methods for acoustic model adaptation: A review. In *ISCA Tutorial and Research Workshop (ITRW)*, pages 67–76, 2001.
- [86] Daniel Povey, Mark JF Gales, Do Y Kim, and Philip C Woodland. MMI-MAP and MPE-MAP for acoustic model adaptation. In *8th European Conference on Speech Communication and Technology*, 2003.
- [87] Kris Hermus, Patrick Wambacq, et al. A review of signal subspace speech enhancement and its application to noise robust speech recognition. *Journal on Advances in Signal Processing (EURASIP)*, 2007(1):045821, 2006.
- [88] Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7893–7897. IEEE, 2013.

- [89] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59. IEEE, 2013. [33](#)
- [90] Pawel Swietojanski and Steve Renals. Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 171–176. IEEE, 2014. [30](#), [32](#)
- [91] Zhizheng Wu, Pawel Swietojanski, Christophe Veaux, Steve Renals, and Simon King. A study of speaker adaptation for DNN-based speech synthesis. In *INTERSPEECH*. ISCA, 2015. [22](#)
- [92] Fabrice Lefevre, J-L Gauvain, and Lori Lamel. Towards task-independent speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 521–524. IEEE, 2001. [23](#)
- [93] Nicola Bertoldi, E Brugnara, Mauro Cettolo, Marcello Federico, and Diego Giuliani. From broadcast news to spontaneous dialogue transcription: Portability issues. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 37–40. IEEE, 2001. [23](#)
- [94] Gwénolé Lecorvé, John Dines, Thomas Hain, and Petr Motlicek. Supervised and unsupervised web-based language model domain adaptation. Technical report, IDIAP, 2012. [25](#), [53](#)
- [95] Philip R Clarkson and Anthony J Robinson. Language model adaptation using mixtures and an exponentially decaying cache. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 799–802. IEEE, 1997. [25](#), [28](#), [30](#), [53](#)
- [96] David M Blei and John D Lafferty. Topic models. In *Text Mining*, pages 101–124. Chapman and Hall/CRC, 2009. [25](#)
- [97] Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černocký. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*. ISCA, 2011. [26](#)

- [98] Frederick Jelinek. Interpolated estimation of markov source parameters from sparse data. In *Workshop on Pattern Recognition in Practice*, 1980. [26](#)
- [99] Dietrich Klakow. Log-linear interpolation of language models. In *5th International Conference on Spoken Language Processing*, 1998. [26](#), [27](#)
- [100] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, 1987. [26](#)
- [101] Reinhard Kneser, Jochen Peters, and Dietrich Klakow. Language model adaptation using dynamic marginals. In *5th European Conference on Speech Communication and Technology*, 1997. [26](#)
- [102] Ke Li, Hainan Xu, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur. Recurrent neural network language model adaptation for conversational speech recognition. In *INTERSPEECH*, pages 3373–3377. ISCA, 2018. [26](#)
- [103] Roland Kuhn. Speech recognition and the frequency of recently used words: A modified Markov model for natural language. In *Proceedings of the 12th Conference on Computational Linguistics*, volume 1, pages 348–350. Association for Computational Linguistics, 1988. [28](#)
- [104] Julien Kupiec. Probabilistic models of short and long distance word dependencies in running text. In *Proceedings of the Workshop on Speech and Natural Language*, pages 290–295. Association for Computational Linguistics, 1989.
- [105] Frederick Jelinek, Bernard Merialdo, Salim Roukos, and Martin Strauss. A dynamic language model for speech recognition. In *Speech and Natural Language*, 1991. [28](#)
- [106] T Joshua and J Goodman. A bit of progress in language modeling extended version. *Machine Learning and Applied Statistics Group Microsoft Research. Technical Report, MSR-TR-2001-72*, 2001. [28](#)
- [107] Yerbolat Khassanov, Tze Yuang Chong, Benjamin Bigot, and Eng Siong Chng. Unsupervised language model adaptation by data selection for speech recognition. In *Asian Conference on Intelligent Information and Database Systems*, pages 508–517. Springer, 2017. [29](#), [39](#)

- [108] Reinhard Kneser and Volker Steinbiss. On the dynamic adaptation of stochastic language models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 586–589. IEEE, 1993. [29](#)
- [109] Rukmini M Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39, 1999.
- [110] Julián D Echeverry-Correa, J Ferreiros-López, Alejandro Coucheiro-Limeres, R Córdoba, and Juan Manuel Montero. Topic identification techniques applied to dynamic language model adaptation for automatic speech recognition. *Expert Systems with Applications*, 42(1):101–112, 2015. [29](#)
- [111] Yangyang Shi, Martha Larson, Pascal Wiggers, and Catholijn M Jonker. K-component adaptive recurrent neural network language models. In *International Conference on Text, Speech and Dialogue*, pages 311–318. Springer, 2013. [29](#)
- [112] Junho Park, Xunying Liu, Mark JF Gales, and Phil C Woodland. Improved neural network based language modelling and adaptation. In *INTERSPEECH*. ISCA, 2010. [30](#), [31](#), [68](#), [70](#)
- [113] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. Recurrent neural network based language modeling in meeting recognition. In *INTERSPEECH*. ISCA, 2011. [30](#), [31](#)
- [114] Hank Liao. Speaker adaptation of context dependent deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7947–7951. IEEE, 2013. [30](#)
- [115] Joao Neto, Luis Almeida, Mike Hochberg, Ciro Martins, Luis Nunes, Steve Renals, and Tony Robinson. Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system. In *European Conference on Speech Communication and Technology*, 1995. [31](#)
- [116] Roberto Gemello, Franco Mana, Stefano Scanzio, Pietro Laface, and Renato De Mori. Linear hidden transformations for adaptation of hybrid ANN/HMM models. *Speech Communication*, 49(10-11):827–835, 2007. [31](#), [32](#)

- [117] Michael Hentschel, Marc Delcroix, Atsunori Ogawa, Tomoharu Iwata, and Tomohiro Nakatani. A unified framework for feature-based domain adaptation of neural network language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7250–7254. IEEE, 2019. [32](#), [33](#), [35](#)
- [118] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2010. [33](#)
- [119] Martin Karafiát, Lukáš Burget, Pavel Matějka, Ondřej Glembek, and Jan Černocký. iVector-based discriminative adaptation for automatic speech recognition. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 152–157. IEEE, 2011. [33](#)
- [120] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003. [33](#), [49](#)
- [121] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. [33](#), [49](#), [108](#)
- [122] Karel Veselý, Shinji Watanabe, Katerina Žmolíková, Martin Karafiát, Lukáš Burget, and Jan Honza Černocký. Sequence summarizing neural network for speaker adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5315–5319. IEEE, 2016. [33](#)
- [123] Tanel Alumäe. Multi-domain neural network language model. In *INTER-SPEECH*, volume 13, pages 2182–2186. ISCA, 2013. [35](#)
- [124] Ottokar Tilk and Tanel Alumäe. Multi-domain recurrent neural network language model for medical speech recognition. In *Baltic HLT*, pages 149–152, 2014. [35](#)
- [125] Salil Deena, Raymond WM Ng, Pranava Madhyashta, Lucia Specia, and Thomas Hain. Semi-supervised adaptation of RNNLMs by fine-tuning with domain-specific auxiliary features. In *INTER-SPEECH*, pages 2715–2719. ISCA, 2017. [35](#)

- [126] Yi Wu, Rong Zhang, and Alexander Rudnicky. Data selection for speech recognition. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 562–565. IEEE, 2007. [40](#), [43](#), [44](#)
- [127] Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 355–362. Association for Computational Linguistics, 2011. [40](#)
- [128] Amittai Axelrod, Philip Resnik, Xiaodong He, and Mari Ostendorf. Data selection with fewer words. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 58–65, 2015. [41](#), [42](#), [47](#)
- [129] Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of EAMT*, volume 2005, pages 133–142, 2005.
- [130] Longyue Wang, Derek F Wong, Lidia S Chao, Junwen Xing, Yi Lu, and Isabel Trancoso. Edit distance: A new data selection criterion for domain adaptation in smt. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 727–732, 2013. [50](#)
- [131] Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 678–683, 2013. [42](#), [47](#)
- [132] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, 2007. [43](#)
- [133] Heng Ji and Ralph Grishman. Data selection in semi-supervised learning for name tagging. In *Proceedings of the Workshop on Information Extraction Beyond The Document*, pages 48–55. Association for Computational Linguistics, 2006. [43](#)

- [134] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014. [43](#)
- [135] Dilek Hakkani-Tür, Giuseppe Riccardi, and Allen Gorin. Active learning for automatic speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages IV–3904. IEEE, 2002. [43](#), [44](#)
- [136] Balakrishnan Varadarajan, Dong Yu, Li Deng, and Alex Acero. Maximizing global entropy reduction for active learning in speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4721–4724. IEEE, 2009.
- [137] Hui Lin and Jeff Bilmes. How to select a good training-data subset for transcription: Submodular active selection for sequences. Technical report, WASHINGTON UNIV SEATTLE DEPT OF ELECTRICAL ENGINEERING, 2009.
- [138] Nobuyasu Itoh, Tara N Sainath, Dan Ning Jiang, Jie Zhou, and Bhuvana Ramabhadran. N-best entropy based data selection for acoustic modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4133–4136. IEEE, 2012. [43](#)
- [139] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 721–726, 2013. [43](#)
- [140] James H Martin and Daniel Jurafsky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009. [44](#)
- [141] Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Ker-Jiann Chen, and Lin-Shan Lee. Chinese language model adaptation based on document classification and multiple domain-specific language models. In *European Conference on Speech Communication and Technology*, 1997. [46](#)

- [142] Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. Toward a unified approach to statistical language modeling for chinese. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(1):3–33, 2002. [46](#)
- [143] Robert C Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the ACL (short papers)*, pages 220–224. Association for Computational Linguistics, 2010. [46](#), [47](#), [51](#)
- [144] Mohammed Mediani, Joshua Winebarger, and Alexander Waibel. Improving in-domain data selection for small in-domain sets. In *Proceedings of IWSLT*, 2014. [47](#)
- [145] Antonio Toral. Hybrid selection of language model training data using linguistic information and perplexity. In *Proceedings of the Workshop on Hybrid Approaches to Translation*, pages 8–12, 2013. [47](#)
- [146] Antonio Toral, Pavel Pecina, Longyue Wang, and Josef Van Genabith. Linguistically-augmented perplexity-based data selection for language models. *Computer Speech & Language*, 32(1):11–26, 2015. [47](#)
- [147] Abhinav Sethy, Panayiotis G Georgiou, and Shrikanth Narayanan. Text data acquisition for domain-specific language models. In *Empirical Methods in Natural Language Processing*, pages 382–389. Association for Computational Linguistics, 2006. [48](#)
- [148] Gerard Salton, Chung-Shu Yang, and Clement T Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, 1975. [48](#)
- [149] Milind Mahajan, Doug Beeferman, and XD Huang. Improved topic-dependent language modeling using information retrieval techniques. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 541–544. IEEE, 1999. [48](#)
- [150] Matthias Eck, Stephan Vogel, and Alex Waibel. Language model adaptation for statistical machine translation based on information retrieval. In *LREC*, 2004.

- [151] Bing Zhao, Matthias Eck, and Stephan Vogel. Language model adaptation for statistical machine translation via structured query models. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2004.
- [152] Yajuan Lu, Jin Huang, and Qun Liu. Improving statistical machine translation performance by training data selection and optimization. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350, 2007. [48](#)
- [153] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007. [49](#)
- [154] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, volume 4, pages 9–56, 2008. [50](#)
- [155] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002. [50](#)
- [156] Ruhi Sarikaya, Agustin Gravano, and Yuqing Gao. Rapid language model development using external resources for new spoken dialog domains. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–573. IEEE, 2005. [50](#)
- [157] Dietrich Klakow. Selecting articles from the language model training corpus. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1695–1698. IEEE, 2000. [51](#)
- [158] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3):103–134, 2000. [52](#)
- [159] Francois Denis, Remi Gilleron, and Marc Tommasi. Text classification from positive and unlabeled examples. 2002.

- [160] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592, 2003. [52](#)
- [161] Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394. Citeseer, 2002. [52](#)
- [162] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and S Yu Philip. Building text classifiers using positive and unlabeled examples. In *ICDM*, volume 3, pages 179–188. Citeseer, 2003. [52](#)
- [163] Anthony Rousseau, Paul Deléglise, and Yannick Esteve. Ted-lium: an automatic speech recognition dedicated corpus. In *LREC*, pages 125–129, 2012. [55](#), [78](#)
- [164] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*. ISCA, 2014. [55](#), [78](#), [82](#)
- [165] Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *INTERSPEECH*. ISCA, 2002. [56](#), [79](#), [82](#)
- [166] Yerbolat Khassanov and Eng Siong Chng. Unsupervised and efficient vocabulary expansion for recurrent neural network language models in asr. *INTERSPEECH*, pages 3343–3347, 2018. [63](#)
- [167] Zhiping Zeng, Yerbolat Khassanov, Van Tung Pham, Haihua Xu, Eng Siong Chng, and Haizhou Li. On the end-to-end solution to mandarin-english code-switching speech recognition. In *INTERSPEECH*, pages 2165–2169. ISCA, 2019. [88](#), [91](#), [98](#), [100](#), [102](#)
- [168] Yerbolat Khassanov, Zhiping Zeng, Van Tung Pham, Haihua Xu, and Eng Siong Chng. Enriching rare word representations in neural language models by embedding matrix augmentation. In *INTERSPEECH*, pages 3505–3509. ISCA, 2019. [63](#)
- [169] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *INTERSPEECH*. ISCA, 2012. [64](#), [65](#), [70](#), [78](#), [79](#), [81](#), [83](#), [104](#)

- [170] Martin Sundermeyer, Ilya Oparin, J-L Gauvain, Ben Freiberg, Ralf Schlüter, and Hermann Ney. Comparison of feedforward and recurrent neural network language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8430–8434. IEEE, 2013. 65
- [171] Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529, 2015. 65
- [172] Christopher Olah. Visualizing representations: Deep learning and human beings, 2015. URL <https://colah.github.io/posts/2015-01-Visualizing-Representations/>. 66
- [173] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, pages 160–167. ACM, 2008. 66, 75
- [174] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. *preprint (http://www.fit.vutbr.cz/imikolov/rnnlm/char.pdf)*, 8, 2012. 67, 71, 72
- [175] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015. 67
- [176] Jaejin Cho, Shinji Watanabe, Takaaki Hori, Murali Karthick Baskar, Hirofumi Inaguma, Jesus Villalba, and Najim Dehak. Language model integration based on memory control for sequence to sequence speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6191–6195. IEEE, 2019. 67
- [177] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language Modeling with Deep Transformers. In *Proc. Interspeech 2019*, pages 3905–3909, 2019. doi: 10.21437/Interspeech.2019-2225. URL <http://dx.doi.org/10.21437/Interspeech.2019-2225>. 67

- [178] Christoph Laichner, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlajter, and Hermann Ney. RWTH ASR Systems for LibriSpeech: Hybrid vs Attention. In *Proc. Interspeech 2019*, pages 231–235, 2019. doi: 10.21437/Interspeech.2019-1780. URL <http://dx.doi.org/10.21437/Interspeech.2019-1780>. 102
- [179] Yongqiang Wang, Abdelrahman Mohamed, Duc Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. Transformer-based acoustic modeling for hybrid speech recognition. *arXiv preprint arXiv:1910.09799*, 2019. 67
- [180] Hainan Xu, Ke Li, Yiming Wang, Jian Wang, Shiyin Kang, Xie Chen, Daniel Povey, and Sanjeev Khudanpur. Neural network language modeling with letter-based features and importance sampling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6109–6113, 2018. 67, 71, 81
- [181] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016. 68, 71
- [182] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *International Conference on Machine Learning*, pages 419–426. Omnipress, 2012. 68, 71, 75
- [183] Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzębski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*, 2017. 68, 72
- [184] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246–252, 2005. 71
- [185] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2009. 71
- [186] Yoshua Bengio and Jean-Sebastien Senecal. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, pages 1–9, 2003. 71

- [187] Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008. [71](#)
- [188] Wenlin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. In *Annual Meeting of the Association for Computational Linguistics*, pages 1975–1985, 2016. [71](#)
- [189] Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In *International Conference on Machine Learning*, 2017. [71](#)
- [190] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. ĀĤernockĀĵ. Strategies for training large scale neural network language models. In *IEEE Automatic Speech Recognition Understanding Workshop (ASRU)*, pages 196–201, 2011. [71](#)
- [191] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *International Conference on Machine Learning*, pages 1017–1024, 2011. [71](#)
- [192] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016. [71](#), [72](#)
- [193] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. [71](#), [74](#), [77](#), [80](#)
- [194] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. [71](#), [74](#), [77](#)
- [195] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Computational Natural Language Learning (CoNLL)*, pages 104–113, 2013. [71](#)
- [196] Angeliki Lazaridou et al. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Annual Meeting of the Association for Computational Linguistics (ACL), Long Papers*, pages 1517–1526, 2013.

- [197] Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. Co-learning of word representations and morpheme representations. In *International Conference on Computational Linguistics (COLING)*, pages 141–150, 2014. [71](#)
- [198] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *TSLP*, 4(1):3, 2007. [72](#)
- [199] Wang Ling et al. Finding function in form: Compositional character models for open vocabulary word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530, 2015. [72](#)
- [200] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, 2017. [72](#)
- [201] Chang Xu et al. Rc-net: A general framework for incorporating knowledge into word representations. In *International Conference on Information and Knowledge Management, CIKM*, pages 1219–1228, 2014. [72](#)
- [202] Manaal Faruqui et al. Retrofitting word vectors to semantic lexicons. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, 2015. [72](#)
- [203] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. [72](#)
- [204] George Miller and Christiane Fellbaum. Wordnet: An electronic lexical database, 1998. [74](#), [77](#)
- [205] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017. [79](#), [82](#)
- [206] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. [79](#), [82](#)
- [207] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, 2014. [79](#), [82](#)
- [208] Yerbolat Khassanov, Haihua Xu, Van Tung Pham, Zhiping Zeng, Eng Siong Chng, Chongjia Ni, and Bin Ma. Constrained output embeddings for end-to-end code-switching speech recognition with only monolingual data. In *INTERSPEECH*, pages 2160–2164. ISCA, 2019. [87](#), [100](#)

- [209] Brenda Danet and Susan C Herring. *The multilingual Internet: Language, culture, and communication online*. Oxford University Press on Demand, 2007. [88](#), [89](#)
- [210] Ke Li, Jinyu Li, Guoli Ye, Rui Zhao, and Yifan Gong. Towards code-switching ASR for end-to-end CTC models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6076–6080. IEEE, 2019. [88](#), [91](#), [96](#), [98](#)
- [211] Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie. Investigating end-to-end speech recognition for mandarin-english code-switching. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6056–6060, 2019. [91](#), [97](#), [102](#)
- [212] Ne Luo, Dongwei Jiang, Shuaijiang Zhao, Caixia Gong, Wei Zou, and Xiangang Li. Towards end-to-end code-switching speech recognition. *arXiv preprint arXiv:1810.13091*, 2018. [88](#), [91](#), [96](#), [98](#), [102](#)
- [213] Dau-Cheng Lyu, Tien-Ping Tan, Eng Siong Chng, and Haizhou Li. Seame: a mandarin-english code-switching speech corpus in south-east asia. In *INTERSPEECH*. ISCA, 2010. [88](#), [91](#), [100](#)
- [214] Peter Auer. *Code-switching in conversation: Language, interaction and identity*. Routledge, 2013. [89](#)
- [215] David Vilares, Miguel A Alonso, and Carlos Gómez-Rodríguez. En-es-cs: An english-spanish code-switching twitter corpus for multilingual sentiment analysis. In *Language Resources and Evaluation (LREC)*, pages 4149–4153, 2016. [89](#), [90](#)
- [216] Shana Poplack. Sometimes I’ll start a sentence in spanish y termino en espanol: toward a typology of code-switching¹. *Linguistics*, 18(7-8):581–618, 1980. [89](#)
- [217] Carol Myers-Scotton. *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press, 1995. [89](#)
- [218] Donald Winford. Code switching: Linguistic aspects. *An Introduction to Contact Linguistics*, pages 126–167, 2003. [90](#)

- [219] Carol Myers-Scotton. Code-switching. *The Handbook of Sociolinguistics*, pages 217–237, 2017. [90](#)
- [220] Shubham Toshniwal, Tara N Sainath, Ron J Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. Multilingual speech recognition with a single end-to-end model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4904–4908. IEEE, 2018. [90](#), [96](#), [97](#), [109](#)
- [221] Tamar Solorio and Yang Liu. Part-of-speech tagging for english-spanish code-switched text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1051–1060. Association for Computational Linguistics, 2008. [90](#)
- [222] Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. Word embeddings for code-mixed language processing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3067–3072, 2018. [90](#)
- [223] Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553, 2018. [91](#), [97](#), [98](#)
- [224] Emre Yilmaz, Henk van den Heuvel, and David A van Leeuwen. Acoustic and textual data augmentation for improved ASR of code-switching speech. *arXiv preprint arXiv:1807.10945*, 2018. [91](#), [97](#), [98](#)
- [225] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [92](#)
- [226] Joyce YC Chan, PC Ching, Tan Lee, and Helen M Meng. Detection of language boundary in code-switching utterances by bi-phone probabilities. In *International Symposium on Chinese Spoken Language Processing*, pages 293–296. IEEE, 2004. [95](#)
- [227] Chung-Hsien Wu, Yu-Hsien Chiu, Chi-Jiun Shia, and Chun-Yu Lin. Automatic segmentation and identification of mixed-language speech using delta-bic and lsa-based gmms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):266–276, 2005. [95](#)

- [228] Dau-Cheng Lyu, Ren-Yuan Lyu, Yuang-chin Chiang, and Chun-Nan Hsu. Speech recognition on code-switching among the chinese dialects. In *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP)*, volume 1, pages I–I. IEEE, 2006. [95](#)
- [229] Suyoun Kim and Michael L Seltzer. Towards language-universal end-to-end speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4914–4918. IEEE, 2018. [96](#)
- [230] Hiroshi Seki, Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R Hershey. An end-to-end language-tracking speech recognizer for mixed-language speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4919–4923. IEEE, 2018. [96](#)
- [231] Pengcheng Guo, Haihua Xu, Lei Xie, and Eng Siong Chng. Study of semi-supervised approaches to improving english-mandarin code-switching speech recognition. *INTERSPEECH*, pages 1928–1932, 2018. [96](#), [100](#), [102](#)
- [232] Emre Yilmaz, Mitchell McLaren, Henk van den Heuvel, and David A van Leeuwen. Semi-supervised acoustic model training for speech with code-switching. *Speech Communication*, 105:12–22, 2018.
- [233] Astik Biswas, Emre Yilmaz, Febe de Wet, Ewald van der Westhuizen, and Thomas Niesler. Semi-supervised acoustic model training for five-lingual code-switched ASR. In *INTERSPEECH*. ISCA, 2019. [96](#)
- [234] Dong Wang and Thomas Fang Zheng. Transfer learning for speech and language processing. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1225–1237. IEEE, 2015. [97](#)
- [235] Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. A first speech recognition system for mandarin-english code-switch conversational speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4889–4892. IEEE, 2012. [97](#)
- [236] Ching-Ting Chang, Shun-Po Chuang, and Hung-Yi Lee. Code-switching sentence generation by generative adversarial networks and its application to data augmentation. *arXiv preprint arXiv:1811.02356*, 2018. [97](#)

- [237] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, 2016. [97](#)
- [238] Sunayana Sitaram and Alan W Black. Speech synthesis of code-mixed text. In *Language Resources and Evaluation (LREC)*, pages 3422–3428, 2016. [97](#)
- [239] Shigeki Karita, Shinji Watanabe, Tomoharu Iwata, Atsunori Ogawa, and Marc Delcroix. Semi-supervised end-to-end speech recognition. In *INTER-SPEECH*, pages 2–6. ISCA, 2018. [98](#)
- [240] Jennifer Drexler and James Glass. Combining end-to-end and adversarial training for low-resource speech recognition. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 361–368, 2018. [98](#)
- [241] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems, NIPS*, pages 577–585, 2015. [101](#)
- [242] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *INTERSPEECH*, pages 3586–3589. ISCA, 2015. [102](#)
- [243] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, 2016. [102](#)
- [244] Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N. Sainath, and Karen Livescu. A comparison of techniques for language model integration in encoder-decoder speech recognition. In *IEEE Spoken Language Technology Workshop, SLT*, pages 369–375, 2018. [104](#)