Mitigating Regional Accent Bias in ASR Systems

Master thesis report

Zirui Li

Mitigating Regional Accent Bias in ASR Systems

Thesis

to obtain the degree of Master of Science
in Embedded Systems
at Delft University of Technology,
to be defended publicly on July 11th, 2023.

by

Zirui LI

Faculty of Electrical Engineering, Mathematics, Computer Science,
Delft University of Technology, Delft, Netherlands,
born in Changsha, Hunan, China.

Thesis committee:

Dr. Odette Scharenborg,	Technische Universiteit Delft
Dr. Marco Zuniga,	Technische Universiteit Delft

Contents

Abstract

End-to-end Automatic Speech Recognition (ASR) systems improved drastically in recent years and they work extremely well on many large datasets. However, research shows that these models failed to capture the variability in speech production and have biases against the variant caused by the regional accented speech. Moreover, ASR research on regional accents is primarily done in languages used by a large population, like English and Arabic, and the effect of regional accented speech on E2E ASR systems in non-popular languages is still unknown. It is important to know the effect of regional accented speech on E2E ASR systems as it helps researchers to build an inclusive E2E ASR system. In this project, I aim to mitigate the biases against regional accented speech. I select standard speech and regional accented speech from CommonVoice's French and German datasets. I combine the state-of-the-art Conformer Recurrent Neural Network Transducer model with Multi-Domain Adversarial Training (MDAT) to boost the performance of regional accented speech while not hurting the performance of the standard speech. Moreover, since the regional accented speech is typically low-resourced, I study the amount of data required for effective MDAT, as well as the effect of different domain classifiers on the performance of Multi-Domain Adversarial Training. Experimental results show that MDAT can mitigate the biases against regional accented speech in both French and German. The best model in French reduces the bias by around 12% and the best model in German reduces the bias by around 7%. Additionally, MDAT is an effective method for bias mitigation as it can achieve similar performance as the MDAT model trained with the full dataset using only a small amount (e.g. 30 minutes) of untranscribed regional accented speech. Finally, different domain classifier architectures were found to have similar effects on the results of MDAT, thus there is no significant differences among the domain classifier in this project.
Index Terms: bias mitigation, automatic speech recognition, regional accented speech, domain adversarial training

Preface

The passage of time has brought me to a farewell from the Delft University of Technology, which has served as a second home over the span of two enriching years of learning and living. During the latter half of this period, my academic curiosity led me to explore Automatic Speech Recognition (ASR), a field of interest I had been nurturing. My research project focused on creating ASR systems devoid of regional accent bias, a compelling and intricate subject. This rigorous endeavor has considerably clarified my future academic and professional goals.

I wish to convey my heartfelt gratitude towards my mentor, Dr. Odette Scharenborg, for her priceless mentorship and unwavering support throughout my Master's thesis project. Her insightful critique, probing questions, and substantive feedback have been instrumental in my academic and professional development. I greatly admire her kindness and commitment which surpass her professional duties, evident in her invaluable assistance during my challenging times at graduate school. I am eternally grateful for her nurturing and mentorship. I also extend my appreciation to my project supervisor, Dr. Tanvina Patel, whose technical expertise greatly benefited this project. Her meticulous approach helped me refine my experimental design, data interpretation, and academic writing. Without the education, assistance, and feedback from Dr. Scharenborg and Dr. Patel, I would not have been able to complete this project within the set timeframe.

Furthermore, I would like to acknowledge all the members of the Speech Lab. Their valuable input, through constructive discussions and suggestions, significantly enhanced this project. The insightful recommendations from faculty members like Jorge and Zhengjun have been incorporated as vital elements of this thesis. Doctoral and Master's students such as Yuanyuan, Chaufeng, and Jingxian also offered invaluable daily discussions for the project. The collective wisdom of the Speech Lab played a significant role in shaping this project. I also extend my gratitude to Jiaying Su and Zhuojia Pan from ESSEC Business School, Paris, France for their insightful input on the French dataset and results.

My deepest appreciation extends to my parents for their unwavering emotional and financial support. Despite being away from home for two years, their unconditional love and support have been my pillar of strength, helping me overcome numerous challenges and shaping the individual I am today. Additionally, I am grateful to my girlfriend, Yu Zhou, whose steadfast love, care, and support have helped me navigate the demanding aspects of this project and the associated difficulties of this phase.

As my journey at Delft concludes, it simultaneously paves the way for a new beginning. I remain optimistic that the knowledge and experience accrued here will serve as a significant enrichment to my life and career.

Zirui Li
Delft, 30th Jun, 2023

1 Introduction

1.1 Motivation

Automatic Speech Recognition (ASR) is a technology that automatically translates human speech to text. Traditionally, an ASR system is composed of two components that are trained separately, a language model (LM) and an acoustic model (AM). Nowadays, end-to-end (E2E) ASR has gained great popularity, where an E2E ASR system learns the LM and the AM jointly. Modern E2E ASR systems perform extremely well on standard speech. However, there is a large variability in speech production, e.g. due to regional accents [1]. To accurately recognize speech produced by different people with different regional accents, the ASR systems have to model the variability in accents effectively. However, modern ASR systems fail to capture the variability and thus have biases against regional accents [2]. One of the definitions of bias in ASR systems is the performance differences between standard speech and some variations of the standard speech [2, 3]. Possible causes of bias include age [4, 5], gender [6, 7], accents [8–10], and human-made biases in data collection and problem formation [2]. The reason E2E ASR systems have biases against regional accents is that they are trained largely on standard speech, and insufficiently model regional accented speech. For example, the well-known Librispeech dataset [11] contains only standard speech with only little regional accented speech, as it was derived from audiobooks. However, it serves as a primary benchmark[1] of modern ASR systems.

To build an inclusive ASR system, for which the performance on non-standard speech is similar to the performance on standard speech, the effect of accents on ASR systems should be studied and the bias against accents should be mitigated. There are many causes of different accents, such as region, social class, and non-native. The majority of the studies [8–10] is done from a broad view of accents, where they do not care about the cause of the accents. In the predecessor work of this project, Zhang et al. [3] study the effect of non-native accents on end-to-end ASR systems. However, research on the impact of regional accents on ASR systems is limited. So far, there have only been a few studies [12–15] on the impact of regional accented speech on ASR systems, and these studies focus on languages that are used by a large population, like English [12, 14], Mandarin Chinese [15],

[1]See **Benchmark** in `https://paperswithcode.com/dataset/librispeech`

and Arabic [13]. For other languages that are used by a smaller population, the effect of their regional accents is still unknown. Although from a qualitative view, regional accented speech has inferior performance than the standard speech, quantifying the bias will help the researcher to compare different techniques and choose the optimal techniques for bias mitigation. In this project, I would like to see the impact of French and German regional accents on an end-to-end ASR system, as French and German are less popular languages compared to English, Mandarin Chinese, and Arabic but still have datasets that are enough for training E2E ASR systems.

One of the difficulties in accurately recognizing regional accented speech is that regional accented speech is typically low-resourced [16, 17]. This gives insufficient data for model training, and thus the performance of regional accented speech is inferior to the performance of standard speech. Therefore, the model mainly trained on standard speech will perform badly on unseen accents. To tackle this issue, our system should be effective with limited hours of data, and our system should be generalized to unseen accents.

The aforementioned problem could potentially be solved by Domain Adversarial Training (DAT) [18], which is a technique that removes domain-related information through an adversarial game between the feature extractor and the domain classifier. One benefit of DAT is that it does not require the label of target domain data, which fits with the fact that regional accented speech is typically low-resourced. Therefore, DAT can achieve domain adaptation under low-resourced settings [19] and can potentially be generalized to unseen domains [18].

DAT has already been applied to various speech processing tasks recently, including ASR [3, 8, 20], speech enhancement [21], and speech conversion [22]. In all, the wide application of DAT demonstrates itself as a reliable domain adaptation technique in the area of speech processing and inspires us to use DAT in the bias mitigation of regional speech. In the area of ASR, [8] uses DAT for accent speech recognition in Mandarin Chinese, [20] uses DAT for recognizing non-native speech in American English and Britain English, and [3] uses DAT for mitigating non-native biases in Dutch. These methods regard standard speech as one domain and non-standard speech as the other domain. For convenience, I will refer to their methods as Binary Domain Adversarial Training (BDAT). In this project, I would like to explore whether different accents can be treated as different domains since differences also exist among different regional accents. I will refer to this as Multi-Domain Adversarial Training (MDAT) in the rest of this thesis.

Furthermore, these methods all use accent classification as the auxiliary domain classification task, but they use different domain classifier architectures for classification. For example, in [8, 20], they use recurrent neural network + linear architecture for accent classification, while in [3], they are using a classifier that is purely composed of linear layers and non-linear activations for accent classification. Since these papers train DAT using different datasets and even for different tasks, it is difficult to compare them directly. This lead to the question that how different domain classifier architectures affect the performance of DAT. In this project, the effect of the aforementioned two classifiers on DAT performance is studied. It can serve as a guideline for choosing domain classifiers when applying DAT.

1.2 Aim and Research Questions

The project aims to mitigate the bias against regional accented speech compared to standard speech. This should be achieved by boosting the performance of regional accents speech while not degrading the performance on the standard speech. Moreover, our system should treat different regional accents as different classes instead of treating them as one class and should be able to generalize to unseen regional accents. In this way, I can build an unbiased ASR system for any regional accents. Additionally, the regional accented speech is low-resourced, therefore I would also like to study how different amounts of training data influence the performance of bias mitigation. This will act as a reference to estimate what performance the bias mitigation can achieve using the available data.

Based on the motivation and the aim of this project, the research questions for this project are as follows:

- **RQ1:** Will Multi-Domain Adversarial Training boost the performance of seen and unseen regional accented speech, especially under low-resource settings?
- **RQ2:** Is Multi-Domain Adversarial Training better than Binary Domain Adversarial Training?
- **RQ3:** How much data is needed to make Multi-Domain Adversarial Training effective?
- **RQ4:** How do different domain classifiers affect the performance of the Domain Adversarial Training?

1.3 Outline

The thesis is organized as follows: Chapter 2 is going to introduce the background knowledge for this work. Chapter 3 explains the dataset, experimental setup, methods, and different implementations of DAT. Chapter 4 gives the results of all experiments. Finally, in Chapter 5, I will answer the RQs and reach conclusions, and give the direction for future work.

2 Background

In this chapter, I give a brief review of the background knowledge related to this project. First, the experiments in this project are conducted as follows: the acoustic features are extracted from the speech, and data augmentation is applied to them. The tokenization technique is used to break the text into tokens. Then both the acoustic features and the token sequence are used to train the model.

I start with the introduction to speech accents in Section 2.1. In Section 2.2, I introduce some basics of deep learning. In Section 2.3, I introduce the data augmentation technique used in this project. Then I introduce tokenization techniques in Section 2.4. Three building blocks, the Conformer, the RNN Transducer, and the Domain Adversarial Training are reviewed in Sections 2.5, 2.6, and 2.7 respectively. Finally, in Section 2.8, the metrics used in this project are shown.

2.1 Speech Accents

According to [23], an accent is a way of pronouncing a certain language that is different from the standard way, because of the speaker's area of origin, social class, and other factors. From the perspective of speech production, accents are marked by variations in prosody, rate, and fluency when producing sounds [24]. These variations are produced by the muscles that control the speech production system. Specifically, because of the impact of dialects or the first language, the muscle will move differently when trying to produce the standard speech, thus the regional accented speech was produced.

In this study, I focus on regional accents. Based on the previous definition, a regional accent is a way of pronouncing a certain language differently because of the speaker's area of origin. A regional accent starts to develop as human beings spread out into isolated communities, and stresses[1] and peculiarities develop[2].

2.2 Basics of Deep Learning

End-to-End ASR systems are built based on deep learning. Before I discuss an E2E ASR system, I give a brief introduction to the foundation of deep learning.

2.2.1 Fully-Connected Neural Network

The fully-connected neural network is the simplest neural architecture. It is also called a Feed-forward neural network (FFN). A fully-connected neural network consists of one or multiple fully connected layers that connect every neuron in this layer to every neuron in the next layer. Figure 2.1 illustrates this.

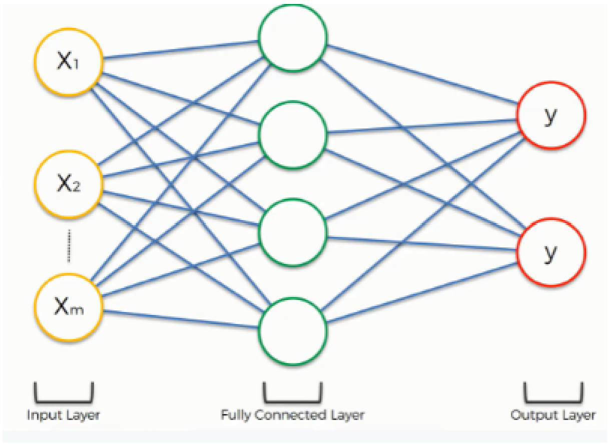

Figure 2.1: A fully-connected neural network with a single hidden layer.

In each layer, a linear transformation is applied to the incoming data, as shown in Equation 2.1. Therefore, every layer in a fully-connected neural network is called a linear

[1]https://en.wikipedia.org/wiki/Stress_(linguistics)
[2]https://en.wikipedia.org/w/index.php?title=Accent%20(sociolinguistics)&oldid=1142713099

layer.

$$y = xW^T + b \tag{2.1}$$

where x is the input for this layer, which could either be the input of the network if this layer is the first layer of the network or the output of the previous layer. W is the weight matrix, and b is the bias. The output y of this layer could be the input for the next layer, or the final output of the network if this is the last layer of the network.

2.2.2 Convolutional Neural Network

The convolutional neural network (CNN) is initially proposed in [25]. An intuitive view of the convolution operation is that the convolution kernel slides over the inputs and computes the inner product between the kernel and the subregion of the inputs that the kernel currently covered. Compared to the fully-connected neural network, CNN can capture local features (e.g. edges in images, or peaks in speech signals) with the position-invariant property, a property that ensures the feature will be captured regardless of the position of the feature. Moreover, CNN is weight-sharing, the weight of the convolutional kernel is shared for all locations in the input signals. Therefore, CNN has fewer parameters than a fully-connected neural network when dealing with high-dimensional input and it is more efficient than the fully-connected neural network in terms of memory, complexity, and optimization, thus can achieve better performance.

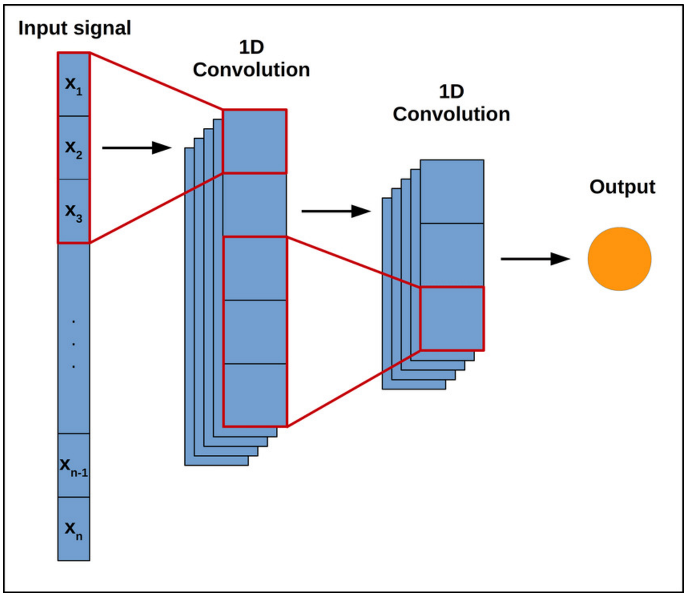

Figure 2.2: 1-dimensional convolution [26]. The input signal has length n and channel 1. Here a kernel with size 3 and channel 5 is used for convolution. Since each channel in the kernel is a 1-dimensional array, it is called a 1-dimensional convolution. It takes the inner product of every adjacent three elements in each channel and sums up the inner product of all channels for one position of the output.

Speech signals are generally represented as (T, n_c), where T is the duration of the signal and n_c is the channel of the signal (e.g. 32 for 32-dimensional MFCC). The duration T of the speech could be too large so it is inefficient for subsequent computation, therefore we use 1-dimensional convolution to reduce T and learn different feature representations in different output channels. Figure 2.2 visualizes the computation of 1-dimensional convolution and Equation 2.2 formulates the computation,

$$y_{n,j} = b_{n,j} + \sum_{c=0}^{n_c-1} \sum_{k=-p}^{p} x_{c,j-k} w_{n,k} \tag{2.2}$$

where $y_{n,j}$ is the output of j-th index at output channel n, $b_{n,j}$ is the bias of j-th index at channel n, which is similar to b in Equation 2.1. n_c is the number of channels in the input signal. $x_{c,j-k}$ is the input value at input channel c, index $j-k$. $w_{n,k}$ is the weight in output channel n index k.

Recently, two advanced convolution operations are proposed, namely Depthwise Convolution, and Pointwise Convolution [27]. Compared to the regular convolution operation shown in Equation 2.2, the Depthwise Convolution does not reduce over the channel dimension ($\sum_{c=0}^{n_c-1}$ in Equation 2.2). For the Pointwise Convolution, it is the same as regular convolution with kernel size 1. Therefore, it only combines information from different channels.

2.2.3 Recurrent Neural Network

Recurrent Neural Networks (RNNs) were initially proposed in [28]. This is also called as a vanilla RNN as it is the basis of other advanced RNNs, such as Long short-term memory [29] and Gated Recurrent Unit [30]. RNN is commonly used for modeling long-term dependency in sequence, such as speech, natural language, and time series.

For a sequence X_t with $t \in [0, T)$, the RNN is formulated as Equation 2.3 and Equation 2.4. Equation 2.3 computes the hidden representation H_t at timestep t, and Equation 2.4 computes the output O_t at timestep t.

$$\mathbf{H_t} = \phi(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h}), \tag{2.3}$$

$$\mathrm{O_t} = \mathrm{H_t W_{hq}} + \mathrm{b_q} \tag{2.4}$$

I explain all notations in the above two equations as follows:

- $\mathbf{H_t}$: Hidden state at current time step t
- ϕ : A non-linear function
- $\mathbf{X_t}$: Input at time t
- $\mathbf{W_{xh}}$: Learned weights for input at time t
- $\mathbf{H_{t-1}}$: Activation at previous time step $t-1$
- $\mathbf{W_{hh}}$: Learned weights: how to use the previous information at $t-1$
- $\mathbf{W}_{\mathrm{hq}}$: Learned weights for output at time t

- $\mathbf{b_h}, \mathbf{b_q}$: Learned bias terms

In this project, I use an advanced RNN architecture called Gated Recurrent Unit (GRU) proposed in [30]. This advanced RNN improves the ability to capture the long-range relationship and mitigates the gradient vanishes/explodes problem in vanilla RNN. Moreover, it has fewer parameters than the LSTM, thus is more computationally efficient.

GRU computes two gates, the reset gate $\mathbf{R_t}$, and the update gate $\mathbf{Z_t}$ at each timestep using the equations in 2.5. $\sigma(\cdot)$ is the Sigmoid function that limits the output within range $(0, 1)$.

$$\begin{aligned} \mathbf{R_t} &= \sigma(\mathbf{X_t W_{xr}} + \mathbf{H_{t-1} W_{hr}} + \mathbf{b_r}) \\ \mathbf{Z_t} &= \sigma(\mathbf{X_t W_{xz}} + \mathbf{H_{t-1} W_{hz}} + \mathbf{b_z}) \end{aligned} \tag{2.5}$$

GRU also computes a candidate state $\tilde{\mathbf{H}}_\mathbf{t}$ using Equation 2.6.

$$\tilde{\mathbf{H}}_\mathbf{t} = \tanh(\mathbf{X_t W_{xh}} + (\mathbf{R_t} \odot \mathbf{H_{t-1}}) \mathbf{W_{hh}} + \mathbf{b_h}) \tag{2.6}$$

Finally, the hidden state $\mathcal{H}_t$ at timestep t is computed in Equation 2.7.

$$\mathbf{H_t} = \mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \tilde{\mathbf{H}}_\mathbf{t} \tag{2.7}$$

2.2.4 Activation Functions

Activation functions sometimes are called non-linearity mappings. It is used to bring non-linearity to the network so that the network could approximate any functions. From an intuitive point of view, it decides whether a neuron should be activated or not. This represents the importance of a particular neuron to the next layer. There are many different activation functions used in deep learning as they are different in physical meaning, performance, and computational efficiency.

ReLU

The ReLU, or Rectified Linear Unit [31] is one of the most commonly used activation functions in deep learning. It only keeps positive values and zeros out all negative values. Therefore only a limited number of neurons will be activated and thus it is computationally efficient. The formula for ReLU is shown in Equation 2.8.

$$f(x) = \max\{0, x\} \tag{2.8}$$

Tanh

Tanh is a widely-used activation function [32]. It scales the input value to the range $(-1, 1)$, which means it zero-centered the data so that the learning will become easier for the next layer. It is formulated as Equation 2.9. The output value of Tanh activation always lies in the range of $(-1, 1)$.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.9}$$

Sigmoid

Sigmoid is another widely-used activation function [32]. It scales the input value to the range $(0,1)$, which is ideal for an output distribution or a filter. It is formulated as Equation 2.10. The output value of Sigmoid activation always lies in the range of $(0,1)$. Sometimes, the symbol σ is used to denote the Sigmoid function.

$$f(x) = \frac{1}{(1+\exp(-x))} \tag{2.10}$$

GLU

The full name of GLU is Gated Linear Unit [33]. I show the formula of GLU in Equation 2.11. Since the Sigmoid function limits the value within the range $(0,1)$, $\sigma(b)$ will be the gate that decides how much information in a will remain in the output.

$$\mathrm{GLU}(a,b) = a \otimes \sigma(b) \tag{2.11}$$

Swish

The Swish activation function is first proposed in [34] to improve ReLU activation by keeping the small negative values, since these values may still be relevant for capturing patterns underlying the data. It is formulated as Equation 2.12.

$$f(x) = x \cdot \sigma(\beta x) \tag{2.12}$$

2.2.5 Dropout

Dropout is an effective technique to prevent neural networks from overfitting. It was originally proposed in [35]. The idea of dropout is straightforward. During training, for each entry in the input signal, the Dropout layer randomly set the entry's value to zero with probability p. This is the same as dropping some connections between two consecutive layers. In this way, neurons will not be optimized too much.

2.2.6 Multi-Head Self Attention

Multi-Head Self Attention, or MHSA, is used to efficiently capture long-term dependency [36], the interaction between elements on different positions. The computation is shown in Equation 2.13. Although both RNN and MHSA can capture long-term dependency, MHSA is more efficient than RNN as it can be parallelized [36] as MHSA is all about matrix multiplication while the computation of one RNN state relies on the computation of the previous state.

$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathrm{head}_1,\ldots,\mathrm{head}_h]\,\mathbf{W}_0 \tag{2.13}$$

where

$$\mathrm{head}_i = \mathrm{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right) \tag{2.14}$$

The Attention function is defined in Equation 2.15.

$$\text{Attention}\,(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.15}$$

2.3 Data Augmentation - SpecAug

SpecAug is a widely-used data augmentation technique in E2E ASR for speech data. It was proposed in [37]. The SpecAug warps the time dimension and masks some values to prevent the neural network from overfitting. The SpecAug is composed of three steps:

1. Time warping. The SpecAug modifies the log mel spectrogram by warping it in the time dimension.

2. Frequency masking. After time warping, a random number of consecutive frequency channels will be masked.

3. Time masking. Finally, a random number of consecutive timesteps will be masked.

2.4 Tokenization

Tokenizers break the sentence into smaller pieces called tokens. It is used for the token inputs to an E2E system, computing the recognition loss, and evaluating the recognition results. Different tokenizers generate different sets of tokens, therefore affecting the recognition results. I introduce two commonly used tokenizers in this section. Byte Pair Encoding (BPE) generates tokens based on the frequency of the subwords, while Unigram Encoding (Unigram) generates tokens based on the probability of the subwords.

2.4.1 Byte Pair Encoding

Byte Pair Encoding, or BPE in short, was first used as a tokenizer in [38]. To train a BPE model, a hyperparameter n needs to be set, which denotes the desired vocabulary size. Then BPE training algorithm does the following things:

1. Calculate the word frequency and break words into characters. Add all characters to the vocabulary as initial tokens.

2. Search for the most frequent pair of existing tokens in the vocabulary. The term *pair* here denotes two adjacent tokens in a word. Merge the pair into a new token, and add the new token to the vocabulary.

3. If the current vocabulary size reaches n, end the training process; otherwise, continue step 2.

When tokenizing text using a trained BPE model, the text will be broken down into characters, then the characters will be greedily merged according to the learned model. If a character is not present in the vocabulary, then a special token *<UNK>* will be used to handle the unknown character.

2.4.2 Unigram Encoding

Unigram Encoding, or Unigram for short, was initially proposed in [39]. Hyperparameter n for the desired vocabulary size also needs to be specified. Different from BPE tokenization, Unigram starts from a large vocabulary and then removes tokens until reaches the desired vocabulary size. The training procedure of a Unigram tokenizer is as follows:

1. Break each word in the corpus into every possible substring. The initial vocabulary consists of all possible tokens. Calculate the probability of each token.
2. At each step of the training, a specific loss, shown in Equation 2.16, is computed over the training corpus based on the current vocabulary, where x_i represents word i, $S(x_i)$ represents all possible tokenizations of word x_i.
3. Compute the difference between the loss before and after removing a specific token from the vocabulary. The token that leads to the smallest difference will be removed.
4. If the current vocabulary size reaches n, end the training process; otherwise, continue step 2.

When tokenizing text using a trained Unigram model, the probability of every possible tokenization result is calculated by multiplying the probability of each token, and the most probable tokenization result is chosen.

$$\mathcal{L} = -\sum_{i=1}^{N} \log\left(\sum_{x\in S(x_i)} p(x)\right) \tag{2.16}$$

2.5 Conformer

The conformer [40] model is a recently proposed speech encoder for ASR that achieves state-of-the-art performance on many public datasets [41]. By integrating convolution with the Transformer architecture, the Conformer can capture both local and global dependencies while being a relatively size-efficient neural net architecture. The Conformer encoder consists of a stack of Conformer encoder layers. By stacking these layers, Conformer can learn hierarchical features from the speech.

$$\begin{aligned}
\tilde{x}_i &= x_i + \frac{1}{2}\,\mathrm{FFN}(x_i) \\
x_i' &= \tilde{x}_i + \mathrm{MHSA}(\tilde{x}_i) \\
x_i'' &= x_i' + \mathrm{Conv}\left(x_i'\right) \\
y_i &= \mathrm{Layernorm}\left(x_i'' + \frac{1}{2}\,\mathrm{FFN}\left(x_i''\right)\right)
\end{aligned} \tag{2.17}$$

The forward pass of one Conformer encoder layer is shown in Equation 2.17. x_i represents the input of the i-th layer. FFN is called the Feed-Forward Network module. MHSA is the Multi-Head Self-Attention module, and Conv is the Convolution module. Usually, we call the dimension of input x_i as the model dimension, and we call the hidden dimension of the FFN the feed-forward dimension. The architectures of the FFN and Conv modules are shown in Figure 2.3 and Figure 2.4, and the MHSA module is similar to I introduced in Section 2.2.6.

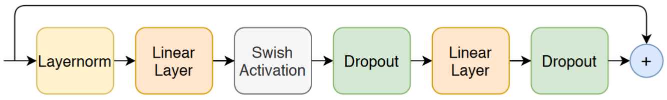

Figure 2.3: Feed-Forward Network module in a Conformer encoder layer [40]. The inputs will be normalized by the Layernorm, then a linear layer will map it from the model dimension to the feed-forward dimension. After the Swish activation and the dropout, a second linear layer will project it back to the model dimension and the dropout will be applied. The final output will be the sum of the input and the output after the second dropout.

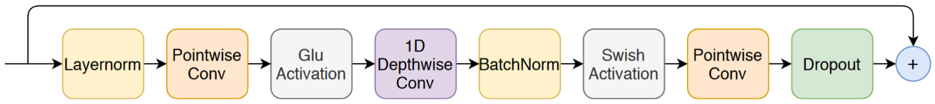

Figure 2.4: Convolution module in a Conformer encoder layer [40]. The inputs will be normalized by the Layernorm, the Pointwise convolution, the GLU activation, and the Depthwise convolution will be applied. Then Batchnorm is used to normalize the intermediate results, followed by a Swish activation. Another Pointwise convolution is applied followed by a dropout. The final output will be the sum of the input and the output after the dropout.

2.6 RNN Transducer

RNN Transducer (RNN-T) [42] is a sequence-to-sequence model (convert a fixed-length input with a fixed-length output where the length of the input and output may differ.) that gains serious attention recently. It achieves state-of-the-art performance on many ASR datasets [43–45]. It does not assume the output is conditionally independent of each other [46], which is a common simplification in ASR. I will introduce the architecture of this model in Section 2.6.1 and then show the computation of the transducer loss in Section 2.6.2.

2.6.1 Architecture

An RNN-T model is composed of four parts, an encoder, a predictor network, a joint network, and a linear classifier with a softmax layer. The architecture of RNN-T is shown in Figure 2.5.

It is worth mentioning that the output $y_{t,u} \in S \cup \emptyset$, where S is the vocabulary and $\emptyset$ is the blank symbol. $\emptyset$ is used to indicate the alignment between the speech frames X_t ($t \in [1,T]$) and tokens y_u ($u \in [1,U]$).

Given an speech audio X, the output y could be generated greedily as follows:

1. Initialized with $t = 1, u = 0$ and $y_{-1} = <BOS>$, where $<BOS>$ means *Begin of sentence*.
2. Compute h_t using $\mathbf{x}_t$ and P_u using $\mathbf{y}_{u-1}$.

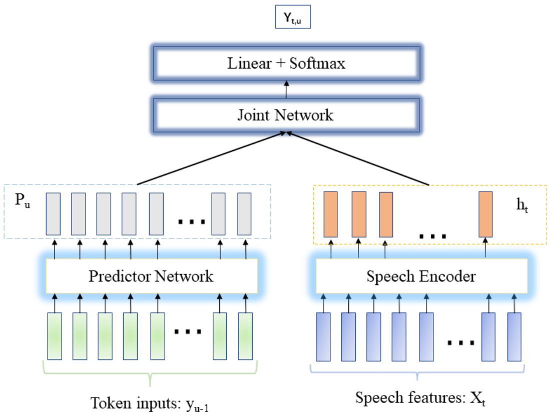

Figure 2.5: RNN Transducer architecture. The speech encoder encodes the speech feature X_t to speech hidden representation h_t. For the predictor network, it consumes the token embedding of y_{u-1} and outputs P_u, where y_{u-1} is the token output at the previous token timestep and P_u is the token hidden representation. The joint network takes both h_t and P_u and joins these two hidden representations by adding these two tensors. Finally, a linear layer with softmax is used to recognize $y_{t,u}$, the token output at speech timestep t and token timestep u.

3. Compute $Y_{t,u}$ using h_t and P_u.

4. If the argmax of $Y_{t,u}$ is a valid token, set $\mathbf{y}_u$ to it. Then set $u = u + 1$ and feed $\mathbf{y}_{u-1}$ back into the predictor.

   If the argmax of $Y_{t,u}$ is $\varnothing$, set $t = t + 1$ (move to the next speech timestep and output nothing).

5. Go back to step 2 unless $t = T + 1$.

2.6.2 Transducer Loss

To train the RNN-T models, we need to compute the RNN Transducer loss. For a pair of (X, y) where X is the speech frames and y is the corresponding token sequence, the transducer loss is computed by summing the probability of all possible alignments between X and y. The alignments are usually represented as a lattice, as shown in Figure 2.6.

To simplify notations, we define

$$\begin{aligned} y(t,u) &= \Pr(y_{u+1} \mid t,u) \\ \varnothing(t,u) &= \Pr(\varnothing \mid t,u) \end{aligned} \tag{2.18}$$

To calculate the sum of all possible alignment efficiently using dynamic programming, a forward variable $\alpha(t,u)$ is defined as the probability of outputting $y_{[1:u]}$ during $x_{[1:t]}$. $\alpha(t,u)$ can be computed recursively as Equation 2.19 with $\alpha(1,0) = 1$.

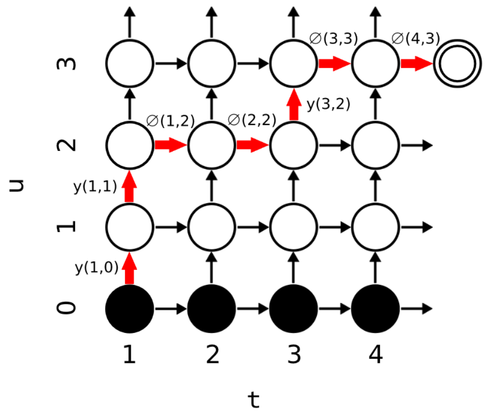

Figure 2.6: An output lattice of RNN Transducer [42]. The horizontal axis is the speech timestep t and the vertical axis is the token timestep u. One possible alignment starts at state ($t = 1, u = 0$) and ends at the final state on the top right of the lattice. When moving up in the lattice, it means that one token at position u has emitted at frame t. On the other hand, when moving right, it means that the current frame is done and the next frame is input to the network.

$$\begin{aligned}\alpha(t,u) = &\alpha(t-1,u)\varnothing(t-1,u)\\ &+\alpha(t,u-1)y(t,u-1)\end{aligned} \tag{2.19}$$

Similar to the purpose of forward variable $\alpha(t,u)$, a backward variable $\beta(t,u)$ is defined as the probability of outputting $y_{[u+1:U]}$ during $x_{[t:T]}$. $\beta(t,u)$ can be computed recursively as Equation 2.20 with $\beta(T,U) = \varnothing(T,U)$.

$$\beta(t,u) = \beta(t+1,u)\varnothing(t,u) + \beta(t,u+1)y(t,u) \tag{2.20}$$

The transducer loss is computed as Equation 2.21,

$$\mathcal{L} = -\log \Pr(\boldsymbol{y}^* \mid \boldsymbol{x}) \tag{2.21}$$

where

$$\Pr(\boldsymbol{y}^* \mid \boldsymbol{x}) = \alpha(T,U)\varnothing(T,U) = \sum_{(t,u):t+u=n} \alpha(t,u)\beta(t,u) \tag{2.22}$$

2.7 Domain Adversarial Training

Before talking about Domain Adversarial Training (DAT), I would like to first explain two terminologies: the source domain and the target domain. In DAT, the source domain is the domain that all other domains will be adapted to, and the target domain is the domain that needs adaptation.

DAT [18] is one of the commonly used methods for domain adaptation. Domain adaptation is the technology for a model trained in one or more source domains to achieve similar performance in different but related target domains. For DAT, the adaptation is achieved by generating domain-invariant features, the features are not distinguishable among different domains.

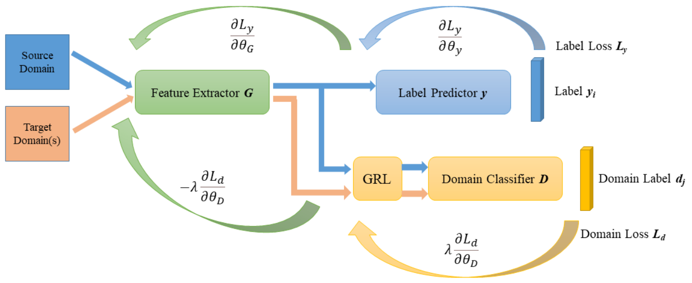
Source Domain
Target Domain(s)
Feature Extractor G
Label Predictor y
GRL
Domain Classifier D
$\frac{\partial L_y}{\partial \theta_G}$
$\frac{\partial L_y}{\partial \theta_y}$
Label Loss L_y
Label y_i
Domain Label d_j
Domain Loss L_d
$-\lambda \frac{\partial L_d}{\partial \theta_D}$
$\lambda \frac{\partial L_d}{\partial \theta_D}$

Figure 2.7: An illustration of the DAT forward computation and backpropagation. DAT contains three parts, a feature extractor $\mathcal{G}$, a label predictor y, and a domain classifier $\mathcal{D}$. **GRL** represents Gradient Reverse Layer. λ is a coefficient used to balance two losses.

The framework of DAT is shown in Figure 2.7. In the forward computation of DAT, both data from the source domain and the target domain(s) will be sent to the feature extractor $\mathcal{G}$ to generate the corresponding features. Then only the features of the source domain will be sent to the label predictor y to train the label predictor. On the other hand, features from both the source and target domains will pass through the Gradient Reverse Layer (GRL) and the domain classifier $\mathcal{D}$. The Gradient Reverse Layer (GRL) keeps the inputs untouched during the forward pass but reverses the upstream gradients during the backward pass. The domain classifier will try to classify the domain of each feature. Two losses, the label loss L_y and the domain loss L_d are computed.

In the backpropagation, the gradient of the label loss L_y will be propagated normally. On the contrary, the gradient of the domain loss L_d will be backpropagated normally in the domain classifier, and it will be negated in GRL. Therefore, the gradient received by the feature extractor $\mathcal{G}$ is not in the direction of minimizing the loss but rather maximizing the loss. In this way, the feature extractor will generate features that are indistinguishable to the domain classifier, and features for all domains look similar to each other.

2.7.1 Uniform Target DAT

One limitation of DAT is that the domain classifier $\mathcal{D}$ will converge earlier than the feature extractor $\mathcal{G}$ [47], therefore no gradient will be received from $\mathcal{D}$, and $\mathcal{D}$ will reach early convergence as well. Applying a small coefficient λ could alleviate this, which introduces an extra hyperparameter for tuning. Since deep neural networks are computationally expensive, this is not an ideal solution.

I apply a Uniform Target DAT similar to [47]. The diagram for Uniform Target DAT is shown in Figure 2.8. Instead of using GRL to reverse the gradient, a uniformly distributed vector is used as the domain label to prevent early convergence. The uniformly distributed

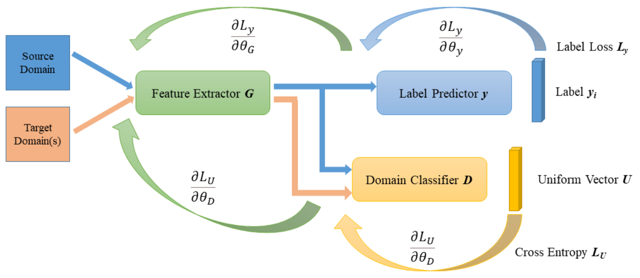
Source Domain
Target Domain(s)
Feature Extractor G
Label Predictor y
Domain Classifier D
Label Loss L_y
Label y_i
Uniform Vector U
Cross Entropy L_U
$\frac{\partial L_y}{\partial \theta_G}$
$\frac{\partial L_y}{\partial \theta_y}$
$\frac{\partial L_U}{\partial \theta_D}$
$\frac{\partial L_U}{\partial \theta_D}$

Figure 2.8: An illustration of the Uniform Target DAT forward computation and backpropagation. The architecture is similar to Figure 2.7, only the GRL is removed. The forward computation is the same as Figure 2.7. The domain loss L_d is computed between the domain prediction and a uniformly distributed vector. The backpropagation is the same as a normal neural network.

vector has the maximum information entropy, thus leading to the maximum confusion among different domains. A cross-entropy loss is computed between the uniformly distributed vector and the prediction of the domain classifier. In this way, the gradient from the classifier will continuously improve the feature extractor to generate domain-invariant features, and no early convergence will be reached.

2.8 Metrics

In this section, I introduce the metrics for evaluating E2E ASR and bias mitigation.

2.8.1 Word Error Rate

Word Error Rate (WER) is a commonly-used metric for evaluating an ASR system. This metric is calculated based on the Levenshtein distance [48] between the prediction and the reference. When calculating the metric, three kinds of error are considered:

- Insertions (I): The prediction contains additional words/characters that are not present in the transcript;
- Deletions (D): The prediction does not contain words/characters that are present in the transcript;
- Substitutions (S): The prediction contains words/characters that replace words/characters in the transcript;

The WER is calculated as Equation 2.23,

$$WER/CER = \frac{S+D+I}{N} \tag{2.23}$$

where N is the number of words in the transcript.

2.8.2 Speech Bias

The definition of bias in this project follows the definition in [2], which is defined as the performance difference (e.g. Word Error Rate) between standard and non-standard speech. In [2], every studied property is divided into two domains with one standard domain and one non-standard domain. To accommodate the multi-domain scenario, where there are one standard speech (standard domain) and multiple regional accents (non-standard domain), I adapt the definition to Equation 2.24,

$$Bias = \frac{\sum_{i=1}^{N} perf_i}{N} - perf_{std} \tag{2.24}$$

where N is the number of regional accents, $perf_i$ is the performance on the non-standard domain i, and $perf_{std}$ is the performance on the standard speech. This means the bias in this project is defined as the difference between the average performance of all regional accents and the performance on the standard speech.

3 Methodology

In this chapter, I introduce the dataset, experimental setup, Domain Adversarial Training, and the domain classifier architecture used in this project in detail. In Section 3.1, a detailed description of the dataset in the project is given, including how and why these accents are chosen. Furthermore, the experimental plan is shown in Section 3.2 so that the setup of different experiments and the corresponding research question are shown. In Section 3.3, I introduce the baseline. Two different types of Domain Adversarial Training methods are introduced in Section 3.4. Finally, I present the different domain classifier architectures used in this project in Section 3.5. All the experiments are implemented using SpeechBrain [49] toolkit.

3.1 Dataset

The dataset used in this project is selected from Mozzila CommonVoice's [50] French and German datasets. CommonVoice [50] is a large open-source dataset for read speech. The dataset not only contains speech and its transcripts, but also other metadata like the gender, age, and most importantly the accent region of the speech. All its data is contributed to and validated by its community. Contributors contribute to the dataset by reading words on the screen or by validating the correctness of other contributors' speech. The accent of a particular speech is decided by its contributor, as the contributor can decide its accent in his/her user profile from predefined categories.

One thing about the CommonVoice dataset is that it is a read-speech dataset, and contributors were assigned random sentences to read regardless of their accents. Therefore it can be assumed that the transcripts over all accents have identical text distributions. Therefore, for this project, I further assume that regional accents only lead to articulation differences that affect the acoustic model, while the language model is indifferent to the accents.

In this project, the selected regional accents should have at least a total duration that is enough for testing. Based on this criterion, the standard speech and the six most frequent regional accents from the corresponding language constitute the dataset. The details of French and German datasets are shown in Table 3.1 and Table 3.2 respectively.

For the French dataset, the reasons to choose these accents are as follows: *Belgique*, *Suisse*, and *France, Sud Ouest* are traditional French-speaking areas. The Quebec area in Canada uses French as the official language, therefore I also regard *Canada* accent as a regional accent for French. For *La Réunion* and *Bénin*, they were French colonies and they use French as their official language, so the corresponding accents are viewed as regional accents as well.

For the German dataset, all accents belong to different sub-regions of modern Germany, except for *Österreichisches*. However, *Österreichisches* is a German-speaking country in the past and present, so it is also a regional accent of German.

The data of a particular accent is decided whether or not to be split into train, dev, and test sets based on its total duration. For accents that are split into the train, dev, and test sets, the ratio for splitting is 0.75 : 0.125 : 0.125. For accents that are not split, they are used as the test set to test the generalizability to the unseen accents.

	Hours		
Accents	Train	Dev	Test
France (standard)	431.50	72.05	71.87
Canada	10.53	1.74	1.76
Belgique	9.78	1.60	1.62
Suisse	4.69	0.79	0.78
La Réunion	-	-	1.57
Bénin	-	-	1.48
France, Sud Ouest	-	-	0.48

Table 3.1: The size of the train, dev, and test set of each French accent in the French dataset.

	Hours		
Accents	Train	Dev	Test
Deutsch (standard)	452.82	75.3	75.5
Nordrhein Westfalen	72.62	12.07	12.08
Österreichisches	24.54	4.08	4.15
Alemanischer	-	-	1.10
Niederrhein	-	-	1.06
Ruhrpott Deutsch	-	-	0.74
Saarland Deutsch	-	-	0.45

Table 3.2: The size of the train, development, and test set of each German accent in the German dataset.

In Table 3.1 and Table 3.2, the accent with red text represents the standard speech (**std** for short), and the accents with green color represent regional accents that are split into train, development, and test set (**seen_acc** for short). The blue set contains accents that are only in the test set (**unseen_acc** for short), and they are used to test the model's generalizability to unseen accents.

All transcripts are processed using the CommonVoice preparation scripts[1] in the GitHub repo of SpeechBrain [49]. This script will normalize the transcripts by removing all symbols and capitalizing all characters for both the French and German datasets.

3.1.1 Subsets of the seen_acc training sets

To answer **RQ3: How much data is needed to make Multi-Domain Adversarial Training effective**, I sample subsets of different total duration from the **seen_acc** training set of each accent in each language. Specifically, the sampled duration starts at 0.5 hours and ends at 4.0 hours with a step size of 0.5 hours. I sample each accent in the **seen_acc** set with the target duration.

3.2 Experimental set-up

3.2.1 Tokenizer

For all experiments introduced in this section, the tokenizer used in each experiment is trained using only the transcripts of the standard speech. The consideration behind this is that the regional accented speech is used unsupervised for training the speech recognizer in experiments, therefore the predictor network that RNN-T used for token processing cannot receive any information regarding the transcripts of the regional accented speech.

3.2.2 Baseline

The baseline model is a Conformer-RNN-T model similar to the model in [40]. The baseline model is trained on:

- The **std** training set

and evaluate on

[1]`https://github.com/speechbrain/speechbrain/blob/develop/recipes/CommonVoice/common_voice_prepare.py`

- **std, seen_acc**, and **unseen_acc** test sets using WER as the metric

to see how much bias a model only trained on standard speech have.

To obtain a solid baseline for both French and German datasets, experiments will be carried out with both BPE and Unigram tokenizers with different vocabulary sizes for both languages. The best model in each language will be used as the baseline model for the corresponding language.

3.2.3 Research Questions 1, 2

To answer the following research questions:

- RQ1: Will Multi-Domain Adversarial Training boost the performance of seen and unseen regional accented speech, especially under low-resource settings?
- RQ2: s Multi-Domain Adversarial Training better than Binary Domain Adversarial Training?

BDAT, MDAT, and MDAT with Uniform Target models are trained:

- The **std** training set (with transcripts),
- The **seen_acc** training set (without transcripts),

and evaluate on

- **std**, **seen_acc**, and **unseen_acc** test sets using WER as the metric.

When training BDAT models, I treat standard speech as one domain and combine *Belgique, Canada*, and *Suisse* as the other domain. In the German case, I similarly treat standard speech as one domain and combine *Nordrhein Westfalen* and *Österreichisches* as the other domain.

The performance comparison between two MDAT models and baseline on **std** and **seen_acc** test set answers research question 1. For research question 2, the performance of two MDAT models on **unseen_acc** test sets can answer. The performance comparison between two MDAT models and BDAT answers research question 3.

3.2.4 Research Question 3

To answer **RQ3: How much data is needed to make Multi-Domain Adversarial Training effective**, the best DAT model of each language from the previous step is trained using:

- The **std** training set (with transcripts),
- sampled subsets of the **seen_acc** training set (without transcripts) mentioned in Section 3.1.1.

and evaluate on

- **std**, **seen_acc**, and **unseen_acc** test sets using WER as the metric.

The comparison among the best model trained with the full dataset in section 3.2.3 and sampled datasets with different total durations on WER gives the answer to research question 3.

3.2.5 Research Question 4

To answer **RQ4: How do different domain classifiers affect the performance of the Domain Adversarial Training**, I train MDAT models using the following domain classifier architectures:

- **DC1:** RNN + Linear layer
- **DC2:** Average over the time dimension + Linear layer

on

- The **std** training set (with transcripts),
- The **seen_acc** training set (without transcripts).

and evaluate on

- **std**, **seen_acc**, and **unseen_acc** test sets using WER as the metric.

Specifically, the best DAT model, which is the DAT model with DC1, in section 3.2.3 for each language is selected and trained with a domain classifier using DC2. The performance comparison between the model using DC1 and the model using DC2 answers research question 5.

3.3 The baseline model

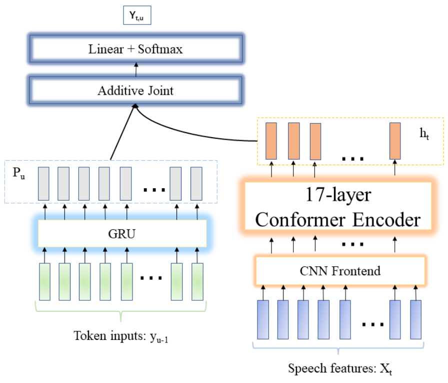

Figure 3.1: The baseline model. X_t is the speech feature at speech timestep t. y_{u-1} is the token embedding of the output at token timestep $u-1$. h_t is the latent representation of speech feature X_t, and P_u is the latent representation of token embedding y_{u-1}. $Y_{t,u}$ is the output token distribution at speech timestep t and token timestep u. The CNN frontend and the Conformer encoder consume X_t and transform them into h_t. GRU consumes y_{u-1} and generates P_u. h_t and P_u are added together and passed to the linear layer to obtain the prediction $Y_{t,u}$.

The Conformer-RNN-T [40] baseline model is shown in Figure 3.1, which consists of five modules, a CNN frontend, a Conformer Encoder for speech features, a Predictor network for token inputs, a joint network to combine the latent speech representations and the token latent representations, and the Linear + Softmax layer for the final recognition.

The CNN frontend is a 3-layer CNN. Each layer has an output channel of 32. The kernel sizes for these three layers are 5, 5, and 1 respectively. The strides of these layers are 2, 2, and 1 respectively. The frontend consumes speech features with shape $(B, T, 80)$, where B is the batch size, T is the duration of the speech, and 80 is the dimension of MFCC features. It produces an output of $(B, T, 20, 32)$ and then be reshaped to $(B, T, 640)$. Finally, a linear layer is used to map the dimension from 640 to the model dimension of the Conformer encoder.

The Conformer Encoder I am using is composed of 17 Conformer layers. Each layer has a model dimension of 512 and a feed-forward dimension of 2048. I apply Relative Positional Encoding [51] for the self-attention mechanism. The output shape of the Conformer Encoder is $(B, T, 512)$.

The Predictor network is a single layer of the GRU network. Its hidden size is 640. For the Transducer Joint Network, I simply use the Additive Joint. The joint dimension is 640. The final classifier takes a 640-dimensional input and maps the output to the number of tokens.

A combination of a single linear layer with a dropout layer is used between the Conformer Encoder and the Joint Network, as well as between the Predictor Network and the Joint Network, to make the dimension compatible between these modules.

3.4 Domain Adversarial Training

In this section, I introduce the implementations of different Domain Adversarial Training techniques. For all DAT implementations, the accent classification is the auxiliary task and the output of the Conformer encoder is used as the feature for accent classification since only the acoustic features need to be adapted.

3.4.1 Binary Domain Adversarial Training and Multi-Domain Adversarial Training

The architecture of BDAT and MDAT is shown in Figure 3.2. Additionally from Figure 3.1, the GRL and the accent classifier are used to classifier the domain of X_t. The auxiliary task I use is accent classification. From an architectural perspective, there is no difference between these two methods. However, from the data perspective, BDAT treats standard speech as one domain and all other non-standard speech as the other domain. Thus the accent classifier only has outputs of shape $(B, 2)$ where B is the batch size and the two values in each row represent the probability of the two classes. For MDAT, each accent in the training set is a unique domain. Therefore, the number of outputs depends on the number of accents in the training set.

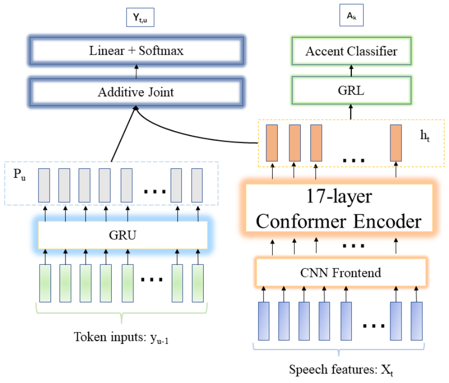

Figure 3.2: The BDAT/MDAT model. X_t is the speech feature at speech timestep t. y_{u-1} is the token embedding of the output at token timestep $u-1$. h_t is the latent representation of speech feature X_t, and P_u is the latent representation of token embedding y_{u-1}. $Y_{t,u}$ is the output token distribution at speech timestep t and token timestep u. A_k is a one-hot vector for the accent label, with the kth position equal to 1. For BDAT, $k \in \{0,1\}$, and for MDAT, $k \in \{0,...,num\_accent-1\}$. In addition to the computation mentioned in Figure 3.1, h_t is also passed to the accent classifier, which includes the GRL layer, to obtain the domain prediction A_k.

3.4.2 MDAT with Uniform Target

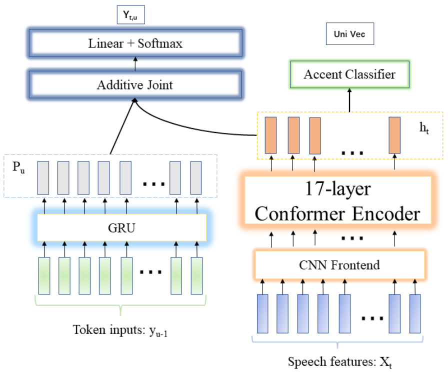

Figure 3.3: The Uniform Target DAT model. X_t is the speech feature at speech timestep t. y_{u-1} is the token embedding of the output at token timestep $u-1$. h_t is the latent representation of speech feature X_t, and P_u is the latent representation of token embedding y_{u-1}. $Y_{t,u}$ is the output token distribution at speech timestep t and token timestep u. Uni_Vec is a uniformly distributed vector of size k, where k is the number of domains in the dataset. Each element in the Uni_Vec has the value of $\frac{1}{k}$. Different from the computation mentioned in Figure 3.2, there is no GRL here, the accent classifier directly consumes h_t, and the domain classification loss is computed between the domain prediction and the uniform vector.

Uniform Target Domain Adversarial Training in this project is proposed by [47]. The architecture for Uniform Target Domain Adversarial Training is shown in Figure 3.3. Different from Figure 3.2, there is no GRL in Figure 3.3 and the domain target is a uniformly distributed vector. Uniform Target DAT directly fits the output with a uniformly distributed vector, so that the maximum entropy or the maximum confusion could be reached [47]. Thus, the trained encoder will generate uniformly distributed features across the feature space, and the accent-dependent information is not distinguishable [47].

3.5 Different Domain Classifier Architectures

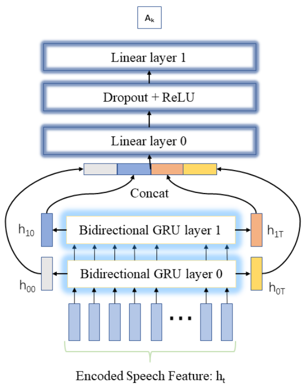

Figure 3.4: The DC1 classifier. The two-layer bidirectional GRU classifier consumes inputs h_t with shape $(B,T,640)$, and the hidden size is 640. The final states of both ends of each GRU layer, which are $(h_{00}, h_{10}, h_{0T}, h_{1T})$ in the figure, are concatenated as the classification features. The features are input to a two-layer linear classifier with the first linear layer having a dimension of 2560 and the second having a dimension of 640. A_k is a one-hot vector for the accent label, with the kth ($k \in \{0, ..., num\_accent - 1\}$) position equal to 1.

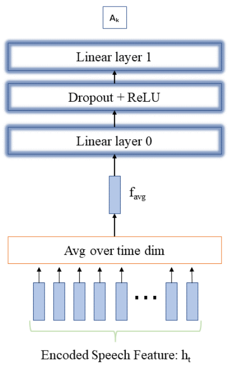

Figure 3.5: The DC2 classifier. The classifier consumes inputs h_t with shape $(B,T,640)$. The acoustic features are first averaged over the time dimension and transform the inputs h_t to feature f_{avg} with shape $(B,640)$. Then a two-layer linear classifier takes the averaged feature for classification, with the dimension of the first linear layer being 640, the dimension of the hidden layer being 640, and the dimension of the output is the number of domains in the dataset.

The architecture for the DC1 and DC2 domain classifiers mentioned in section 3.2.5 is plotted in Figure 3.4 and Figure 3.5 respectively. DC1 in Figure 3.4 is an RNN-based classifier. The speech hidden representation h_t is passed to a two-layer bidirectional GRU and the final states of both directions of both layers are concatenated as the feature for domain classification. DC2 in Figure 3.5 takes h_t of shape $(B,T,640)$, averaging the time dimension, and transforms the feature to shape $(B,640)$. It directly passes the averaged feature to a two-layer linear network for accent classification.

4 Results

This chapter presents the results of the experiments. Section 4.1 presents the results of the baseline experiments. Section 4.2 shows the results of the DAT experiments. In Section 4.3, the results of how much data is needed for DAT are shown. Finally, I will show the results of DAT with different domain classifiers.

4.1 Baseline model results

The model with the best performance in terms of the average WER over all accents is selected as the baseline. Separate baseline models for French and German are created.

4.1.1 French

Details		Accents								
Tokenizer	# of tokens	standard	Belgique	Canada	Suisse	Bénin	La Réunion	Sud Ouest	Avg	Bias
BPE	1000	**9.36**	**23.21**	29.90	24.40	34.28	**24.44**	**24.09**	**24.24**	17.36
BPE	2000	9.82	23.47	30.30	24.22	34.17	25.42	25.94	24.76	17.43
BPE	4000	10.49	23.78	31.16	24.57	34.56	24.88	24.92	24.91	**16.82**
BPE	5000	10.71	23.77	31.60	24.74	34.74	24.98	25.94	25.21	16.92
Unigram	1000	9.51	23.26	**29.89**	**23.69**	**34.16**	25.16	24.52	24.31	17.27
Unigram	2000	9.74	23.46	29.66	24.52	34.70	24.41	24.77	24.47	17.18

Table 4.1: %WER of the baseline on French standard and accented test set.

The results of French baseline are shown in Table 4.1, the first two columns denote the tokenizer and the number of tokens used by the model. The test WER of the standard speech and six regional accents are shown in the following seven columns. The average WER over all accents and the bias are shown in the last two columns.

For Table 4.1, we can observe that BPE tokenization with 1000 tokens performs the best in 4 out of 7 accents, including the standard speech. It has the lowest average WER across all accents in the French dataset. If the tokenizer is changed to Unigram, or the number of tokens is increased, the overall performance degrades. Therefore, I choose the BPE tokenizer with 1000 tokens as our baseline for French, and all our French experiments in the following sections will be configured in the same way.

Additionally, in Table 4.1, *Canada* and *Bénin* perform worse than other regional accents, with the WER for *Canada* and *Bénin* is about 5% and 10% worse than other accents respectively. The bias is around 17%, and it is similar across all models.

4.1.2 German

Details		Accents								
Tokenizer	# of tokens	standard	Nordrhein Westfalen	Österrei-chisches	Alemani-scher	Nieder-rhein	Ruhrpott	Saarland	Avg	Bias
BPE	1000	5.81	18.28	**18.88**	**15.87**	**17.96**	**14.50**	18.43	**15.68**	**11.51**
BPE	2000	6.18	19.54	19.44	17.54	18.56	15.20	18.27	16.39	11.91
BPE	4000	6.97	20.25	20.33	18.28	19.86	16.25	18.43	17.20	11.93
BPE	5000	7.14	20.75	20.45	18.80	19.68	16.59	19.55	17.57	12.16
Unigram	1000	**5.72**	**18.05**	**18.88**	16.05	18.13	14.72	**18.23**	**15.68**	11.62
Unigram	2000	6.16	18.97	19.16	16.45	18.76	15.79	18.27	16.22	11.74

Table 4.2: %WER of the baseline on German standard and accented test set.

For the German baseline in Table 4.2, we observe that as the number of tokens increases, the performance degrades, which is similar to the French baseline results. Both the BPE with 1000 tokens and Unigram with 1000 tokens perform equally well on the average WER of all accents. The Unigram with 1000 tokens performs slightly better on the standard speech. However, the BPE with 1000 tokens performs better on regional accents, where 4 out of 6 regional accents achieve the best performance. Based on the better results on the accented speech, the BPE tokenizer with 1000 tokens was used as the baseline for German.

In Table 4.2, except that *Alemanischer* and *Ruhrpott* have a WER of around 15%, all other regional accents perform similarly with a WER of 18%. The bias is around 12% for all models.

4.2 DAT

The results of Domain Adversarial Training (DAT) are shown in Table 4.3 and Table 4.4. The first two columns of the table indicate the model and the domain classifier that the model uses. The WER on the standard speech and the six regional accents are shown in the next seven columns. The last two columns show the average WER and the bias.

4.2.1 French

Details		Accents								
Model	domain cls	standard	Belgique	Canada	Suisse	Bénin	La Réunion	Sud Ouest	Avg	Bias
Baseline	-	**9.36**	23.21	29.90	24.40	34.28	24.44	24.09	24.24	17.36
BDAT	DC1	9.45	9.82	17.76	9.58	21.37	9.26	10.91	12.59	3.67
MDAT	DC1	9.57	**9.77**	17.38	**9.33**	**20.82**	**9.13**	**10.04**	**12.29**	**3.18**
MDAT	DC1_Uni	9.43	10.12	**17.02**	9.86	21.30	9.38	10.44	12.51	3.59
MDAT	DC2	9.59	10.06	17.41	9.44	21.34	9.35	10.19	12.48	3.38
Best Abs Impv		-0.07	13.44	12.88	15.07	13.46	15.31	14.05		

Table 4.3: %WER of the Multi-Domain Adversarial Training on French standard and accented test set. **DC1** denotes the RNN + Linear layer domain classifier. **DC1_Uni** has the same domain classifier with **DC1** but implements Uniform Target DAT. **DC2** is the domain classifier that Averaging over the time dimension + Linear layer.

In Table 4.3, compared with the **Baseline** row, all DAT models greatly improve the WER of all accents, though the performance on the standard speech slightly deteriorates. MDAT with **DC1** performs the best. It achieves the best WER on 5 out of 7 accents and reaches the lowest average WER over all models in Table 4.3.

For our implementation of the MDAT model with **DC1_Uni**, its performance is worse than the regular MDAT model with **DC1**. It only exceeds the performance of the regular implementation in *Standard* and *Canada*. From the perspective of the overall average WER, it is worse than the regular DAT by about 0.22%.

It is worth noticing that both of the MDAT models outperform the BDAT model. This fits with the expectation, as I believe MDAT provides better modeling of the variability of the regional accents. However, from the perspective of bias, although MDAT with **DC1** has the lowest bias among all MDAT models, this comes at the cost of the standard speech.

4.2.2 German

In Table 4.4, all DAT models achieve huge improvement on WER and all MDAT models outperform the BDAT model, which is consistent with the results of the French experiments. Additionally, similar to the French experiments, the MDAT model with **DC1** performs the best among all models. However, all DAT models have a higher WER on standard speech than the baseline, as the WER for the baseline on standard speech is 5.81% and the WER of the standard speech for all DAT models is above 6%.

Details		Accents								
Model	domain cls	standard	Nordrhein Westfalen	Österrei-chisches	Alemani-scher	Nieder-rhein	Ruhrpott	Saarland	Avg	Bias
Baseline	-	**5.81**	18.28	18.88	15.87	17.96	14.50	18.43	15.67	11.51
BDAT	DC1	6.03	10.47	**10.35**	7.64	10.03	6.97	11.12	8.94	3.40
MDAT	DC1	6.02	10.01	10.42	7.71	**9.39**	**6.88**	10.54	**8.71**	3.14
MDAT	DC1_Uni	6.05	**9.84**	10.72	**7.61**	9.45	7.12	**10.27**	8.72	**3.12**
MDAT	DC2	6.09	10.01	10.55	7.67	9.77	7.01	10.41	8.79	3.15
Best Abs Impv		-0.21	8.44	8.53	8.26	8.57	7.62	8.16		

Table 4.4: %WER of the Multi-Domain Adversarial Training on German standard and accented test set. **DC1** denotes the RNN + Linear layer domain classifier. **DC1_Uni** has the same domain classifier with **DC1** but implements Uniform Target DAT. **DC2** is the domain classifier that Averaging over the time dimension + Linear layer.

From the view of bias, the MDAT model with **DC1** performs the best, as it achieves the lowest bias of 3.12% among all models in Table 4.4. However, this is achieved through a higher WER on standard speech. For the MDAT model with **DC1**, the WER on standard speech is 0.03% lower than the MDAT model with **DC1_Uni**, while its bias is only 0.02% higher than its counterpart. Therefore, we believe the MDAT model with **DC1** is the best model in Table 4.4.

Based on Table 4.3 and Table 4.4, I decide to use MDAT with **DC1** for experiments in Section 4.3, as this is the best model for both languages.

4.3 RQ3: How much data is needed?

The WER of the MDAT model trained with different sizes of the dataset is shown in this section. The first column of each table represents the size of the dataset. The WER of each accent is given in the following columns. Finally, the average WER and the bias are given in the last two columns.

4.3.1 French

Details	Accents								
Duration (hr)	standard	Belgique	Canada	Suisse	Bénin	La Réunion	Sud Ouest	Avg	Bias
full	9.57	**9.77**	17.38	9.33	20.82	9.13	10.04	12.29	3.18
0.5	9.51	10.23	17.11	9.77	20.79	9.09	11.00	12.50	3.49
1.0	**9.34**	9.93	17.61	9.50	20.75	8.90	10.75	12.40	3.57
1.5	9.44	10.27	16.90	9.45	20.58	9.59	9.52	12.25	3.28
2.0	9.41	10.11	16.96	9.94	20.53	9.37	10.35	12.38	3.47
2.5	9.38	9.98	**16.61**	10.11	**20.45**	9.20	10.38	12.30	3.41
3.0	9.47	10.07	17.03	9.79	21.33	9.07	9.89	12.38	3.39
3.5	9.51	9.84	17.38	**9.17**	21.50	**8.84**	**9.43**	**12.24**	**3.18**
4.0	9.50	10.52	17.67	10.20	22.00	9.63	10.35	12.84	3.90

Table 4.5: %WER when different amounts of hours of accented speech are used for training the French model.

The results for French are shown in Table 4.5. In Table 4.5, We can see that with only half an hour's untranscribed regional accent data, the model can achieve a performance that is comparable to when using all available accented speech data, with some accents even outperforming the full dataset condition. Moreover, our model achieves the lowest average WER with 3.5 hours of data. However, the differences in average WER and bias are insignificant among all models.

4.3.2 German

Details	Accents								
Duration (hr)	standard	Nordrhein Westfalen	Österrei-chisches	Alemani-scher	Nieder-rhein	Ruhrpott	Saarland	Avg	Bias
full	6.23	9.63	10.68	7.67	9.85	6.93	10.12	8.73	2.92
0.5	5.86	9.22	10.34	7.15	9.59	6.79	10.04	8.43	3.00
1.0	5.71	9.44	9.94	7.01	**8.94**	7.19	10.27	8.36	3.09
1.5	5.78	9.45	10.02	7.02	9.91	7.08	9.81	8.44	3.10
2.0	5.86	9.37	10.01	6.94	9.39	6.49	9.84	8.27	2.81
2.5	5.75	9.35	**9.91**	**6.89**	9.03	**6.21**	**9.77**	**8.13**	**2.78**
3.0	5.80	9.43	10.00	7.37	9.34	6.79	9.96	8.38	3.02
3.5	5.81	9.51	10.32	7.46	9.69	7.14	10.04	8.57	3.22
4.0	**5.70**	**8.99**	10.00	7.40	9.66	6.47	10.62	8.41	3.16

Table 4.6: %WER of the different hours of accent training set in German.

The results for German are shown in Table 4.6. In Table 4.6, with only half an hour's untranscribed regional accent data, the model also achieves a performance that is better than the full data, all accents have achieved better performance than the full dataset. Moreover, with 2.5 hours of data, our model achieves the best average WER of 8.13%. On the other hand, the differences in average WER and bias are insignificant among all models, similar to the French results.

4.4 Different domain classifiers

In this section, I discuss the impact of different domain classifiers on DAT performance. The results for each language are shown in Table 4.3 and Table 4.4. Specifically, I compare the row using domain classifier **DC1** with the row **DC2**.

For the French experiments, the domain classifier **DC2** yields worse performance than the MDAT model with **DC1**. It has an average WER of 12.48%, which is 0.19% higher than the **DC1** implementation.

Similarly, in German experiments, the domain classifier **DC2** yields the worst performance among all MDAT models. It has an average WER of 8.79%, which is 0.08% higher than the **DC1** implementation.

5 Discussions and Conclusions

In this chapter, I will discuss the results listed in Chapter 4. Through the discussion in Section 5.1, the answers to our research questions will be derived. In Section 5.2, the conclusions are drawn. Finally, in Section 5.3, I will point out the future direction of this research.

5.1 Discussion

For our **RQ1: Will Multi-Domain Adversarial Training boosts the performance of seen and unseen regional accented speech, especially under low-resource settings**, by comparing the baseline row and the row with MDAT and **DC1**, we can tell for both French and German, DAT greatly boosts the performance of seen and unseen regional accented speech but brings a slight deterioration on the standard speech, regardless of different DAT implementations. For French, the results in Table 4.3 show that DAT brings a huge absolute improvement on WER ranging from 12.8% to 15.3% for all regional accents. DAT clearly works well under low-resourced settings in French, as *Suisse* only has 4.7 hours of data while achieving the lowest WER of 9.33% and a maximum absolute improvement of 15.07%. For German, the results in Table 4.4 also show a considerable improvement over all accents. The best absolute improvement on WER of all regional accents is around 7.6% to around 8.5%. We cannot tell whether DAT can work under low-resourced settings from the German experiment, as the German training set has sufficient data as shown in Table 3.2.

However, we see a small deterioration in both the standard French and the standard German, as all DAT models obtain worse results than the baseline model on standard speech. For French, the degree of deterioration varies from 0.09% to 0.23%, while for German, the degree of deterioration is around 0.25% for all DAT models. This indicates that bias mitigation using DAT may cause a small deterioration of performance on standard speech.

For our **RQ2: Is Multi-Domain Adversarial Training better than Binary Domain Adversarial Training?**, we can compare **BDAT** using **DC1** with **MDAT** using **DC1** in Table 4.3 and Table 4.4. In Table 4.3, we can see that MDAT performs better than BDAT in French. MDAT has better performance over 6 out of 7 accents, except on standard speech. Similarly, in Table 4.4, we see MDAT outperforms BDAT on 5 out of 7 accents, and MDAT has a better average WER. Combining the results of both French and German, we can see that MDAT is better than BDAT, although the performance gap between MDAT and BDAT is small.

For our **RQ3: How much data is needed to make Multi-Domain Adversarial Training effective**, we can see Table 4.5 and Table 4.6. These tables show that with untranscribed speech data of only 0.5 hours of total duration, MDAT can achieve a considerable improvement compared to the baseline and yields an average performance that approaches the average performance trained with the full dataset. However, the differences in the performance among different models are trivial and an increasing amount of data does not necessarily improve the performance in this project.

To further explore the pattern between the amount of training data and the performance of the DAT model, the CER tables for French and German are shown in Section 6.1. Through the CER tables, the conclusion remains the same as the amount of training data has an insignificant impact on the performance of DAT in this project.

For our final **RQ4: How do different auxiliary classifiers affect the performance of the Domain Adversarial Training**, we can tell from Table 4.3 and Table 4.4, where we can compare the row starting with **DC1** with the row with **DC2** in each table. For French in Table 4.3, we see that the model with **DC1** outperforms the model with **DC2** on all accents. Besides, the model with **DC1** trivially outperforms the model with **DC2** on average WER by 0.19%. Similarly, for German, the model with **DC1** outperforms the model with **DC2** on 4 out of 7 accents. It has a trivial 0.08% average WER higher than the model with **DC2**. The results for the French and the German experiments show that different auxiliary classifiers have little impact on the performance of DAT.

Compare with our predecessor's work [3], our DAT boosts the performance of regional accent speech, while DAT in [3] improves the WER of read speech while worsening the WER of conversational speech. Since our experiments are only carried out based on read speech data, combined with the results in [3], we can tell that DAT can mitigate bias in read speech.

Finally, I would like to discuss the bias metric I am using in this project. From Table 4.1 to Table 4.6, there are two things about the bias metric that are worth noticing. First, the reduction in the bias is brought by two aspects. One is the improvement in WER/CER of regional accented speech and the other is the deteriorated WER/CER of standard speech. For the second aspect, this is not the expected behavior of bias mitigation. Therefore, the bias metric is only valid when the performance on the standard domain does not deteriorate.

5.2 Conclusions

In this project, I aim to mitigate the bias against regional accented speech in a state-of-the-art ASR system. I show that Multi-Domain Adversarial Training can reduce the bias against regional accented speech, and that also works for unseen accents. Moreover, I show that Multi-Domain Adversarial Training has a better performance than Binary Domain Adversarial Training used in [3]. Furthermore, for French and German, with enough transcripted standard speech, half an hour of untranscribed accent speech can achieve a good result. Finally, I see that different domain classifiers have similar impacts on the performance of Domain Adversarial Training. Domain classifiers used in this project have no significant impact on the performance of DAT.

5.3 Future work

Based on the contents of this project, I can point out some future directions to further develop this work.

To start with, all models in this project are tested without utilizing external language models. Although RNN Transducer is an end-to-end model that does not require an external language model, using an external language model usually improves performance. Moreover, to better incorporate the external language model, the internal language model [52] of the RNN Transducer should be suppressed. Otherwise, there could be a domain mismatch between the internal language model and the external language model, which leads to the degradation of the performance. The internal language model is learned through the training of the prediction network. To suppress the internal language model, I could use the Internal Language Model Estimation [53] and the Internal Language Model Training [54, 55] to remove or adapt the internal language model so that no domain mismatch between the internal language model and the external language model.

Finally, I would like to see the performance of more auxiliary classifiers, since only two domain classifiers are studied in this project. Neural architectures like 1-dimensional convolution and Quasi RNN [56] could be used to build the auxiliary classifiers.

References

[1] M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris *et al.*, “Automatic speech recognition and speech variability: A review,” *Speech communication*, vol. 49, no. 10-11, pp. 763–786, 2007.

[2] S. Feng, O. Kudina, B. M. Halpern, and O. Scharenborg, “Quantifying bias in automatic speech recognition,” *arXiv preprint arXiv:2103.15122*, 2021.

[3] Y. Zhang, Y. Zhang, B. M. Halpern, T. Patel, and O. Scharenborg, “Mitigating bias against non-native accents,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2022, 2022, pp. 3168–3172.

[4] T. Pellegrini, I. Trancoso, A. Hämäläinen, A. Calado, M. S. Dias, and D. Braga, “Impact of age in asr for the elderly: preliminary experiments in european portuguese,” in *Advances in Speech and Language Technologies for Iberian Languages: IberSPEECH 2012 Conference, Madrid, Spain, November 21-23, 2012. Proceedings.* Springer, 2012, pp. 139–147.

[5] F. Claus, H. Gamboa Rosales, R. Petrick, H.-U. Hain, and R. Hoffmann, “A survey about asr for children,” in *Speech and Language Technology in Education*, 2013.

[6] M. Adda-Decker and L. Lamel, “Do speech recognizers prefer female speakers?” in *Ninth European Conference on Speech Communication and Technology*, 2005.

[7] M. Garnerin, S. Rossato, and L. Besacier, “Investigating the impact of gender representation in asr training data: A case study on librispeech,” in *3rd Workshop on Gender Bias in Natural Language Processing.* Association for Computational Linguistics, 2021, pp. 86–92.

[8] S. Sun, C.-F. Yeh, M.-Y. Hwang, M. Ostendorf, and L. Xie, “Domain adversarial training for accented speech recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 4854–4858.

[9] Y.-C. Chen, Z. Yang, C.-F. Yeh, M. Jain, and M. L. Seltzer, “Aipnet: Generative adversarial pre-training of accent-invariant networks for end-to-end speech recognition,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2020, pp. 6979–6983.

[10] A. Jain, M. Upreti, and P. Jyothi, “Improved accented speech recognition using accent embeddings and multi-task learning.” in *Interspeech*, 2018, pp. 2454–2458.

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

[12] W. Barry, C. Hoequist, and F. Nolan, “An approach to the problem of regional accent in automatic speech recognition,” *Computer Speech & Language*, vol. 3, no. 4, pp. 355–366, 1989.

[13] G. Droua-Hamdani, S.-A. Selouani, and M. Boudraa, "Speaker-independent asr for modern standard arabic: effect of regional accents," *International Journal of Speech Technology*, vol. 15, no. 4, pp. 487–493, 2012.

[14] M. Najafian, A. DeMarco, S. Cox, and M. Russell, "Unsupervised model selection for recognition of regional accented speech," in *Fifteenth annual conference of the international speech communication association*, 2014.

[15] F. Weninger, Y. Sun, J. Park, D. Willett, and P. Zhan, "Deep learning based mandarin accent identification for accent robust asr." in *INTERSPEECH*, 2019, pp. 510–514.

[16] A. Rajpal, A. R. MV, C. Yarra, R. Aggarwal, and P. K. Ghosh, "Pseudo likelihood correction technique for low resource accented asr," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7434–7438.

[17] N. Das, S. Bodapati, M. Sunkara, S. Srinivasan, and D. H. Chau, "Best of both worlds: Robust accented speech recognition with adversarial transfer learning," *arXiv preprint arXiv:2103.05834*, 2021.

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[19] D. Grießhaber, N. T. Vu, and J. Maucher, "Low-resource text classification using domain-adversarial learning," *Computer Speech & Language*, vol. 62, p. 101056, 2020.

[20] H. Hu, X. Yang, Z. Raeesy, J. Guo, G. Keskin, H. Arsikere, A. Rastrow, A. Stolcke, and R. Maas, "Redat: Accent-invariant representation for end-to-end asr by domain adversarial training with relabeling," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6408–6412.

[21] C.-F. Liao, Y. Tsao, H.-Y. Lee, and H.-M. Wang, "Noise adaptive speech enhancement using domain adversarial training," *arXiv preprint arXiv:1807.07501*, 2018.

[22] H. Du, L. Xie, and H. Li, "Noise-robust voice conversion with domain adversarial training," *Neural Networks*, vol. 148, pp. 74–84, 2022.

[23] "The new oxford american dictionary." New York, 2005.

[24] M. Celce-Murcia, D. M. Brinton, J. M. Goodwin *et al.*, *Teaching pronunciation: A reference for teachers of English to speakers of other languages.* Cambridge University Press, 1996.

[25] Y. LeCun, B. Boser, J. Denker, D. Henderson, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[26] A. Shenfield and M. Howarth, "A novel deep learning model for the detection and identification of rolling element-bearing faults," *Sensors*, vol. 20, no. 18, p. 5112, 2020.

[27] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[29] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[31] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.

[32] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[33] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” in *International conference on machine learning.* PMLR, 2017, pp. 933–941.

[34] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[37] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” *Interspeech 2019*, Sep 2019. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

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

[39] T. Kudo, “Subword regularization: Improving neural network translation models with multiple subword candidates,” *arXiv preprint arXiv:1804.10959*, 2018.

[40] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” *arXiv preprint arXiv:2005.08100*, 2020.

[41] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, “Recent developments on espnet toolkit boosted by conformer,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2021, pp. 5874–5878.

[42] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012.

[43] A. Andrusenko, A. Laptev, and I. Medennikov, “Towards a competitive end-to-end speech recognition for chime-6 dinner party transcription,” *arXiv preprint arXiv:2004.10799*, 2020.

[44] F. Boyer, Y. Shinohara, T. Ishii, H. Inaguma, and S. Watanabe, “A study of transducer based end-to-end asr with espnet: Architecture, auxiliary loss and decoding strategies,” in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU).* IEEE, 2021, pp. 16–23.

[45] N. Moritz, F. Seide, D. Le, J. Mahadeokar, and C. Fuegen, “An investigation of monotonic transducers for large-scale automatic speech recognition,” in *2022 IEEE Spoken Language Technology Workshop (SLT).* IEEE, 2023, pp. 324–330.

[46] L. Lugosch, “Sequence-to-sequence learning with transducers,” Nov 2020. [Online]. Available: https://lorenlugosch.github.io/posts/2020/11/transducer/

[47] S. Mirsamadi and J. H. Hansen, “Multi-domain adversarial training of neural network acoustic models for distant speech recognition,” *Speech Communication*, vol. 106, pp. 21–30, 2019.

[48] V. I. Levenshtein *et al.*, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[49] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, “SpeechBrain: A general-purpose speech toolkit,” 2021, arXiv:2106.04624.

[50] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, “Common voice: A massively-multilingual speech corpus,” *arXiv preprint arXiv:1912.06670*, 2019.

[51] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” *arXiv preprint arXiv:1803.02155*, 2018.

[52] E. Variani, D. Rybach, C. Allauzen, and M. Riley, “Hybrid autoregressive transducer (hat),” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2020, pp. 6139–6143.

[53] Z. Meng, S. Parthasarathy, E. Sun, Y. Gaur, N. Kanda, L. Lu, X. Chen, R. Zhao, J. Li, and Y. Gong, “Internal language model estimation for domain-adaptive end-to-end speech recognition,” in *2021 IEEE Spoken Language Technology Workshop (SLT).* IEEE, 2021, pp. 243–250.

[54] Z. Meng, N. Kanda, Y. Gaur, S. Parthasarathy, E. Sun, L. Lu, X. Chen, J. Li, and Y. Gong, "Internal language model training for domain-adaptive end-to-end speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2021, pp. 7338–7342.

[55] W. Zhou, Z. Zheng, R. Schlüter, and H. Ney, "On language model integration for rnn transducer based speech recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2022, pp. 8407–8411.

[56] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," *arXiv preprint arXiv:1611.01576*, 2016.

[57] D. Ulyanov, "Multicore-tsne," https://github.com/DmitryUlyanov/Multicore-TSNE, 2016.

[58] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[59] "Sctk toolkit." [Online]. Available: https://github.com/usnistgov/SCTK

6 Appendix

6.1 RQ3: How much data is needed? - CER perspective

Details	Accents								
Duration (hr)	standard	Belgique	Canada	Suisse	Bénin	La Réunion	Sud Ouest	Avg	Bias
full	3.34	3.58	7.26	3.53	8.72	3.12	3.46	4.72	**1.61**
0.5	3.31	3.77	7.32	3.72	8.76	3.19	3.73	4.83	1.77
1.0	**3.23**	**3.55**	7.55	3.48	8.71	**3.06**	3.64	4.75	1.77
1.5	3.27	3.71	7.32	3.53	8.58	3.28	**3.10**	4.68	1.65
2.0	3.28	3.67	7.26	3.57	8.49	3.25	3.21	4.68	1.63
2.5	3.28	3.56	**7.02**	3.76	**8.45**	3.23	3.36	**4.67**	1.62
3.0	3.27	3.59	7.19	3.56	8.79	3.21	3.23	4.69	1.66
3.5	3.29	3.57	7.39	**3.39**	9.06	3.14	3.20	4.72	1.67
4.0	3.28	3.81	7.58	3.76	9.36	3.36	3.49	4.95	1.95

Table 6.1: %CER of the different hours of accent training set in French.

Details	Accents								
Duration (hr)	standard	Nordrhein Westfalen	Österrei-chisches	Alemani-scher	Nieder-rhein	Ruhrpott	Saarland	Avg	Bias
full	1.76	2.58	3.34	2.00	2.70	1.76	2.81	2.42	0.77
0.5	1.63	2.42	3.22	1.87	2.60	1.65	2.76	2.31	0.79
1.0	1.60	2.52	3.13	**1.75**	**2.36**	1.74	2.78	2.27	0.78
1.5	1.63	2.54	3.08	1.78	2.70	1.69	2.82	2.32	0.81
2.0	1.65	2.53	3.07	1.74	2.58	1.67	2.65	2.27	0.72
2.5	1.62	2.54	3.04	1.85	2.42	**1.51**	**2.59**	**2.22**	**0.71**
3.0	1.63	2.69	**3.06**	2.00	2.61	1.61	2.68	2.33	0.81
3.5	1.66	2.77	3.31	1.98	2.62	1.92	2.87	2.45	0.92
4.0	**1.59**	**2.34**	3.11	1.90	2.63	1.64	2.93	2.31	0.84

Table 6.2: %CER of the different hours of accent training set in German.

The CER tables of the French and the German when training with different amount of data are shown in Table 6.1 and Table 6.2. Similar to the WER tables in Section 4.3, the differences in CER among different models are insignificant. Therefore, the amount of training data has an insignificant impact on the performance of DAT in this project.

6.2 Feature Maps

To see how the distribution of the accent data changes before and after applying DAT, the feature maps of the baseline model and all DAT models are shown in Table 4.3 and Table 4.4. The output from the linear layer that consumes the output of the Conformer encoder is used as the feature for all plots. To reduce the dimension, we use the Multicore-TSNE [57], which is a fast implementation of the TSNE algorithm [58], to reduce the dimension from 640 to 2. We use default parameters to run TSNE, except the perplexity is set to 5. Figure 6.1 to Figure 6.10 shows the feature maps we generated.

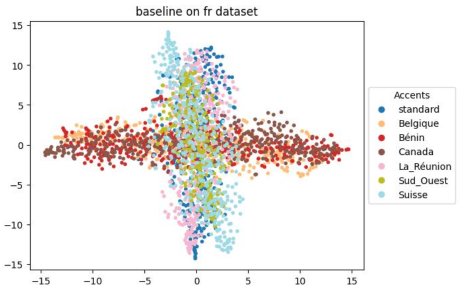

Figure 6.1: Baseline model trained with French dataset.

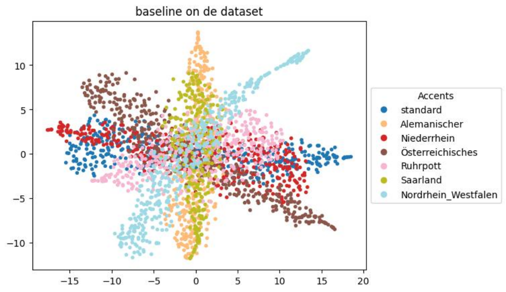

Figure 6.2: Baseline model trained with German dataset.

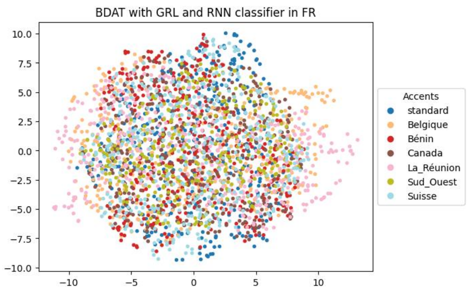

Figure 6.3: BDAT trained with French dataset.

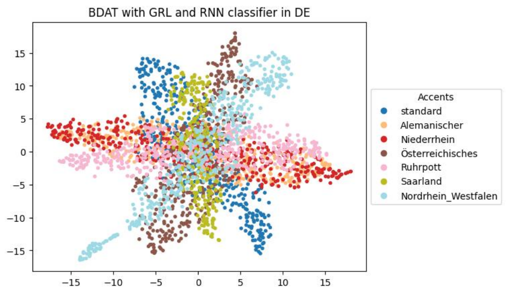

Figure 6.4: BDAT trained with German dataset.

In Figure 6.1 and Figure 6.2, we see that the features for different accented speech are rotations of the standard speech. For each accent, figures are narrowly scattered. After applying different DAT methods, the features from different accents either mixed (e.g. Figure 6.3, Figure 6.5, and Figure 6.10) or lie as the concentrated rotation of the standard speech (e.g. Figure 6.6, Figure 6.8, and Figure 6.9).

It is interesting to see that whether the features from different accents are visually indistinguishable does not imply the performance of different models. For example, MDAT with **DC1** achieves the best performance over all models in German experiments, while MDAT with **DC2** performs the worst. However, when we compare Figure 6.6 and Figure 6.10, MDAT with **DC2** generates indistinguishable features while features generated by MDAT with **DC1** are clearly distinguishable

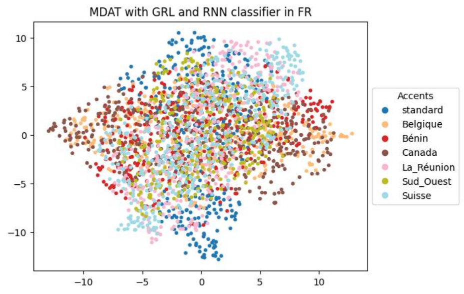

Figure 6.5: MDAT with GRL+RNN trained with French dataset.

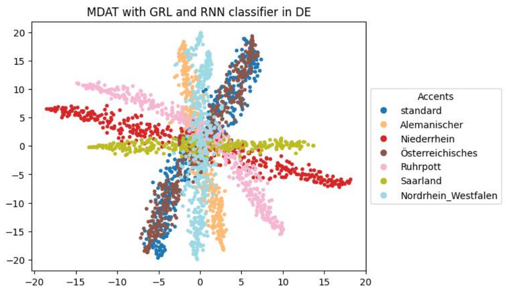

Figure 6.6: MDAT with GRL+RNN trained with German dataset.

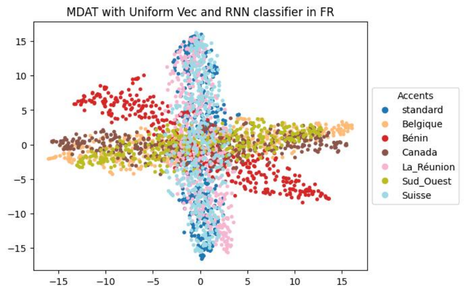

Figure 6.7: MDAT with Uniform Vec trained with French dataset.

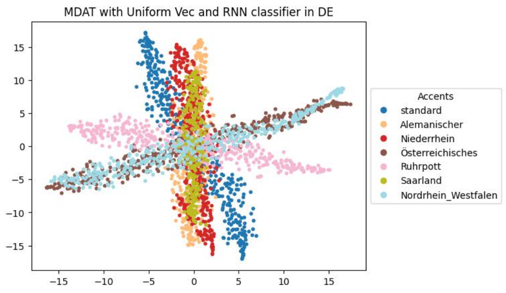

Figure 6.8: MDAT with Uniform Vec trained with German dataset.

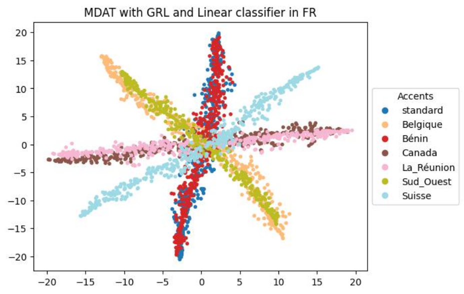

Figure 6.9: MDAT with GRL + Avg trained with French dataset.

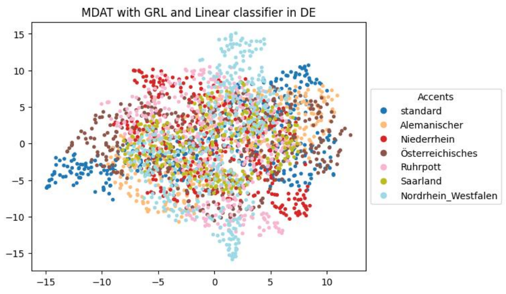

Figure 6.10: MDAT with GRL + Avg trained with German dataset.

6.3 Error Analysis: *Bénin* and *Canada* as examples

```
 1:   38  ->  les ==> la
 2:   31  ->  à ==> <eps>
 3:   28  ->  sur ==> sous
 4:   28  ->  d ==> de
 5:   26  ->  est ==> <eps>
 6:   24  ->  et ==> <eps>
 7:   24  ->  <eps> ==> l
 8:   23  ->  aux ==> au
 9:   20  ->  dans ==> d
10:   18  ->  est ==> a
11:   12  ->  l ==> <eps>
12:   12  ->  de ==> du
13:   12  ->  des ==> d
14:   11  ->  l ==> les
15:    9  ->  leur ==> la
16:    8  ->  un ==> <eps>
17:    8  ->  <eps> ==> qu
18:    8  ->  ces ==> ses
19:    8  ->  d ==> <eps>
20:    7  ->  de ==> <eps>
```

Table 6.3: The top-20 confusion pairs in the Bénin accent of the baseline.

We have observed a higher WER for *Bénin* and *Canada* in Table 4.3, before and after the application of DAT. Therefore, we use the SCTK toolkit [59] to analyze the recognition results of these two accents in the baseline and DAT experiments. The error analysis will provide a better insight into what DAT improves and what hampers the further improvement of a specific accented speech.

```
 1:   26  ->  du ==> de
 2:   24  ->  de ==> <eps>
 3:   21  ->  le ==> la
 4:   19  ->  ils ==> il
 5:   18  ->  de ==> des
 6:   16  ->  à ==> <eps>
 7:   15  ->  et ==> <eps>
 8:   14  ->  le ==> les
 9:   12  ->  de ==> d
10:   12  ->  la ==> l
11:   11  ->  un ==> <eps>
12:   10  ->  le ==> <eps>
13:    9  ->  un ==> le
14:    9  ->  au ==> aux
15:    9  ->  en ==> <eps>
16:    8  ->  l ==> <eps>
17:    8  ->  ses ==> ces
18:    8  ->  en ==> un
19:    8  ->  il ==> ils
20:    8  ->  aux ==> au
```

Table 6.4: The top-20 Confusion pairs in the Canada accent of the baseline.

In Table 6.3 and Table 6.4, one of the major errors is that both the references (left side of the arrow) and the hypotheses (right side of the arrow) have the same starting substring, but the subsequent substrings are different. This fits the common sense of regional accented speech, as words with the same starting substring are likely to have similar pronunciations and the regional accented speech will make these words indistinguishable.

```
 1:   24  ->  les ==> la
 2:   13  ->  est ==> <eps>
 3:   12  ->  et ==> <eps>
 4:    8  ->  d ==> de
 5:    8  ->  à ==> <eps>
 6:    7  ->  l ==> <eps>
 7:    7  ->  un ==> <eps>
 8:    6  ->  aux ==> au
 9:    6  ->  il ==> elle
10:    6  ->  la ==> l
11:    6  ->  sur ==> sous
12:    5  ->  de ==> <eps>
13:    5  ->  des ==> d
14:    5  ->  ses ==> ces
15:    5  ->  sur ==> <eps>
16:    4  ->  <eps> ==> en
17:    4  ->  <eps> ==> et
18:    4  ->  <eps> ==> le
19:    4  ->  au ==> <eps>
20:    4  ->  au ==> aux
```

Table 6.5: The top-20 Confusion pairs in the Bénin accent of the MDAT model with **GRL+RNN**.

```
 1:   15  ->  de ==> <eps>
 2:   10  ->  ces ==> ses
 3:   10  ->  de ==> des
 4:   10  ->  le ==> la
 5:    9  ->  le ==> les
 6:    8  ->  et ==> <eps>
 7:    8  ->  la ==> l
 8:    8  ->  un ==> <eps>
 9:    8  ->  à ==> <eps>
10:    7  ->  <eps> ==> à
11:    7  ->  ils ==> il
12:    7  ->  le ==> <eps>
13:    6  ->  d ==> de
14:    6  ->  du ==> <eps>
15:    5  ->  au ==> aux
16:    5  ->  du ==> de
17:    5  ->  est ==> <eps>
18:    5  ->  la ==> les
19:    5  ->  les ==> des
20:    5  ->  ses ==> ces
```

Table 6.6: The top-20 Confusion pairs in the Canada accent of the MDAT model with **GRL+RNN**.

We show the confusion pairs of *Bénin* and *Canada* after training the MDAT model with **GRL+RNN** in Table 6.5 and Table 6.6. We can tell that after training with DAT, the misrecognition of the words that have the same starting substring is greatly mitigated. The number of such misrecognition in the top-20 confusion pairs has reduced significantly for both of the accents. This demonstrates the effectiveness of the DAT.

However, both of these two accents still suffer from higher WER than other accents. This is caused by the misrecognition of the empty symbol *<eps>*. This leads to the majority of confusion pairs. Some words are aligned as empty symbols, while some positions that should be aligned to empty symbols are actually aligned as words. I believe this is related to the ineffective internal language model and can be alleviated by incorporating an external language model.