DEVELOPMENT OF A SEPEDI-ENGLISH CODE-SWITCHED AUTOMATIC SPEECH RECOGNITION SYSTEM USING CONNECTIONIST TEMPORAL CLASSIFICATION

by

AMANDA PHALADI

DISSERTATION

Submitted in fulfilment of the requirement for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

in the

FACULTY OF SCIENCE AND AGRICULTURE

(School of Mathematical and Computer Sciences)

at the

UNIVERSITY OF LIMPOPO

Supervisor: Dr TI Modipa

2025

DEDICATION

I dedicate this dissertation to my beloved siblings: Dikeledi Phaladi, Tshegofatso Phaladi, Mogau Phaladi, and Tebogo Phaladi. May this work serve as a testament to the power of God, determination, and the validity of our dreams. Always remember that anything you set your mind to is achievable. Together, let's honor our parents' sacrifices and make them proud of our accomplishments and resilience. Keep striving, keep believing, and never doubt your potential.

DECLARATION OF AUTHORSHIP

I Amanda Phaladi declare that The development of a Sepedi-English code-switched automatic speech recognition system using connectionist temporal classification is my own original work and all information extracted from other sources is acknowledged as such by means of complete references. I further affirm that I have not submitted this work to any other university for any other degree or examination.

Signature:  Date: 21/10/2024

ACKNOWLEDGEMENTS

Firstly, I would like to honour and appreciate the God of Mount Zion, who gave me the courage, wisdom, and strength throughout the research period. All the honour and glory are directed to Him.

Secondly, I would like to extend my heartfelt gratitude to my supervisor, Dr. Thipe Modipa. His invaluable guidance, support, and encouragement have been instrumental in bringing out the best in me. His insightful feedback and unwavering belief in my abilities have greatly contributed to the success of this research.

I also wish to express my profound appreciation to my parents and my grand mother. Their unwavering faith in me and constant reminders of their belief in my potential have been a source of immense motivation. Their love, support, and prayers have kept me going throughout this journey.

Additionally, I am grateful to my friends who provided emotional support and encouragement. Their presence and kind words have made this journey more bearable.

ABSTRACT

Speech technology includes several approaches and technologies that allow machines to engage with spoken language, which include spoken dialog systems and automatic speech recognition. The end-to-end (E2E) techniques, such as Connectionist Temporal Classification (CTC) and attention-based methods, dominate Automatic Spdeech Recognition (ASR) system development. However, these methodologies have primarily advanced in research for high-resourced languages with extensive speech datasets, leaving low-resource languages relatively underserved. The efficacy of the CTC method specifically for Sepedi, a low-resource language, remains uncertain.

This study addresses this gap by developing and evaluating an automatic speech recognition (ASR) system for Sepedi-English code-switched speech. Utilizing the Sepedi Prompted Code Switching (SPCS) corpus and applying the CTC approach, we implemented an E2E ASR system. We rigorously evaluated the system's performance across various parameters using both the National Centre for Human Language Technology (NCHLT) Sepedi test corpus and the Sepedi Prompted Code Switching corpus.

Our findings demonstrate promising results overall. However, the system faced challenges in accurately recognizing speech from the Sepedi NCHLT test corpus. This study shows the importance of adapting advanced ASR techniques to suit the linguistic characteristics and data limitations of low-resource languages. Addressing these challenges is crucial for expanding the applicability of speech technology to diverse linguistic contexts, ultimately facilitating broader accessibility and usability of ASR systems worldwide.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

In the era of rapidly evolving technology, speech recognition has emerged as a transformative tool with profound implications for human-computer interaction and communication. Speech recognition is pertinent to a technology that enables a device to record uttered phrases via a microphone from a person speaking [1]. This technology processes the audio input, converting it into digital data that can be interpreted by software. The field of speech recognition has been and continues to experience a rapid growth in development. It has numerous applications across various domains, offering a wide range of potential advantages. Moreover, the technology has a broad spectrum of applications, from virtual assistants and chatbots to automated transcription and video closed captioning [1]. Speech technology composes of a series of methods and tools that allow devices to engage with other entities through speech, such as ASR, spoken dialog systems (SDSs), and more [2].

ASR utilizes CTC to effectively manage variable-length audio inputs and produce accurate transcriptions without the need for explicit alignment information during both training and decoding, making it a powerful technique in the ASR pipeline. Recently, CTC has been widely recognized as an appropriate method for the improvement of the performance of ASR systems even under low-resource conditions [3]. Low-resources languages are languages that have received less attention in research, suffer from limited resources, lack advanced computerization, face fewer privileges, are less frequently taught, or have low population density, among other designations which is a relatively large inventory when compared to languages such as English [4]. Researchers have proposed various approaches that incorporate CTC into ASR models, such as the Non-autoregressive transformer with CTC-enhanced decoder input [5], which concentrates on the output of a CTC-based model to improve its accuracy.

1.2. Problem Statement

ASR systems rely heavily on significant quantities of transcribed speech data to learn the statistical patterns of speech sounds and their relationships in a particular language [2]. ASR systems require a vast amount of transcribed speech data to effectively cover the variability and intricacies inherent in a language. As the quantity of data increases, the system's accuracy and robustness typically improve, making scalability a crucial factor in enhancing ASR performance.

Additionally, it was elaborated that the Sepedi language is made up of approximately 32 phonemes, as compared to English with 44 phonemes. Phoneme is defined as a basic unit of sound [6]. Consequently, the limited number of phonemes in the Sepedi language categorizes it as a low-resourced language. Similarly, languages in the Nguni group, such as siSwati,isiNdebele, isiZulu, isiXhosa and those in the Sotho-Tswana family, are also classified as low-resource languages [7].

Low-resource languages are those that have been understudied, lack extensive digital resources, and face various limitations. These languages often have fewer speakers, limited educational support, and minimal technological development. Compared to widely-spoken languages like English, they typically possess a relatively large phoneme inventory [4]. This characteristic, combined with the scarcity oftraining data, poses significant challenges in developing accurate ASR systems for these languages.

Moreover, it was observed that it is difficult to create models that can integrate the increased complexity that is associated with the phenomenon of code-switching as that is what lies at the core of code-switching [8]. This makes it more difficult to develop accurate ASR systems when training data is scarce. Furthermore, creating models that can handle the increased complexity that comes with the phenomenon of code-switching is a difficult task [9]. This challenge is further compounded when the languages involved are under-resourced, as the limited availability of text and acoustic data sets constrain the modeling capacity.

The problem is that current research landscape in ASR has predominantly focused on languages with abundant speech data, leaving a significant gap in the development

of ASR systems for under-resourced languages. This challenge becomes even more pronounced when addressing code-switching, particularly for languages with limited linguistic resources. The task of building accurate ASR systems for these languages remains complex and underexplored. Therefore, this study aims to develop an ASR system for code-switched languages using the CTC approach, addressing the unique challenges posed by such languages.

1.3. Motivation

Recently, speech recognition technology has focused on significant research and development [10]. ASR systems now operate with much higher efficiency and accuracy, courtesy of the developments in deep learning, machine learning and natural language processing [11]. Utilizing a hybrid strategy that included acoustic and language model training, the accuracy of ASR processing code-switched speech in South Africa was improved with limited data [12]. Concurrently, the Markov assumption plays a key role in enabling fast and parallelized decoding in ASR systems [13]. Moreover, without requiring precise congruence between the input and output sequences, the Markov assumption enables the model to calculate, in a single forward pass, the complete output sequence given the input sequence. A notable drawback of the Markov model is that the probability of transitioning out of a state is independent to the states a patient may have encountered prior to entering that state [14].

E2E models optimize every component of the network by using a coherent goal function that aligns with the ASR objective, hence streamlining the optimization operations. Traditional hybrid models, on the other hand, optimize different components separately, which does not guarantee that the global optimum is reached. Moreover, ASR workflow is greatly simplified by E2E models, which create letter or words directly. In contrast, classic hybrid automobiles have a complex design, demanding substantial expertise and years of experience in ASR. Additionally, E2E models are more compact due to their utilization of a single network for ASR. This compactness enables their deployment on high-accuracy devices [15].

In recent times, ASR models based on the CTC approach have demonstrated remarkable performance, particularly when they are fine-tuned from wav2vec models

[16]. As a result, researchers have put forward two techniques that can e utilised to transfer knowledge being - joint classification and joint classification learning to enhance CTC-based models by integrating the contextual knowledge of pre-trained LMs into the ASR systems. Furthermore, a proposed auxiliary loss function for ASR, which utilizes the CTC objective, is both straightforward and effective [13].

CTC and attention-based methods are two primary E2E models utilized in ASR. CTC employs dynamic programming to efficiently address sequential problems while making effective use of Markov assumptions. Employing attention mechanism enables the alignment of acoustic frames with recognized symbols [17]. Attention network has recently shown promising performance in E2E ASR as an alternative to RNNs [18]. Attention-based models can achieve cutting-edge performance in ASR, however, they often entail greater complexity and computational demands than models based on CTC.

Despite the CTC being a widely utilized technique for ASR in many languages, most of the research and development has focused on high-resource languages which have substantial volumes of speech data available for training and evaluation [19]. Hence, evaluating the performance of the system and the analysis can provide understanding of the efficacy of CTC in low-resource language ASR and identify areas for improvement in Sepedi ASR technology.

1.3.1. Aim

The aim of study is to develop an automatic speech recognition system for Sepedi-English code-switched language using Connectionist Temporal Classification approach.

1.3.2. Objectives

The objectives of the study are to:

i Analyse the Sepedi Prompted Code Switching corpus.

ii Train an automatic speech recognition system for code-switched language using the CTC approach.

iii Evaluate the implemented code-switched ASR system.

1.3.3. Research Questions

The present study will answer the following questions:

i What is the type of analysis suitable for the SPCS corpus?

ii How do you train an automatic speech recognition system for code- switched language using CTC?

iii What is the evaluation process for the trained ASR system?

1.3.4. Significance of the study

This study contribute to the field of ASR development of a Sepedi/English code-switching ASR system that utilizes CTC for efficient and accurate transcription of spoken language. We use CTC by virtue of its ability to handle different languages and dialects, as well as the efficiency of CTC decoding in parallelized systems. CTC approach also eliminates the need for explicit alignment between the signal of the input speech and the corresponding transcription, reducing the need for manual annotation and simplifying the training process. Our efforts in creating a proficient and precise code-switching ASR system aim to drive the evolution of speech recognition technology, with broad potential uses including language education, transcription of spoken words into text, and operation of voice-activated devices.

1.4. Dissertation Outline

This dissertation is organised into five chapters.

- In Chapter 2, an an extensive literature review of ASR is presented and different models used in ASR are also discussed.
- Chapter 3 presents the methodology and design of the proposed ASR system using CTC approach. The proposed system is explained in detail and the analysis of the data used.

- In Chapter 4, the examination of the system to assess its performance is discussed.
- In Chapter 5, a conclusion of the findings is drawn and recommendations are given for possible future research.

2. LITERATURE REVIEW

2.1. Introduction

In this chapter, we examine and evaluate research and ongoing research endeavors concerning ASR employing CTC. Additionally, we highlight the shortcomings of speech recognition in low-resource languages.

2.2. Sepedi-English Code-Switching data

Bilingual speakers commonly engage in a phenomenon known as code-switching (CS), in which they use words from two languages within a single discourse [13]. CS can occur in two different ways, depending on where the switching of languages happens: intra-sentential CS, in which switches occur within the announcement, articulation and inter-sentential CS, in which switches occur at the boundaries of utterances. There are two types of code-switching depending on the location of language switches [17]. Intra-sentential code-switching occurs. This chapter examines and evaluates research and ongoing research endeavors concerning ASR employing CTC. Additionally, it underscores the deficiencies of speech recognition in low-resource languages. Single utterance, while inter-sentential code-switching occurs between different utterances and it is considered more challenging as the acoustic differences between mixed languages within a single utterance can be more significant than those between different utterances.

Furthermore, it encompasses of two additional types of code-switching: Situational Code Switching: This type takes place when a transition in social setting is marked by a change in language. It reflects how speakers adapt their language use based on the context or situation they find themselves in. Conversational (Metaphorical) Code Switching: This occurs when conversations take place adhering to he announcements and articulations, whereby speakers switch between languages metaphorically or conversationally. It could involve using terms or expressions from both lan-

guages to convey a specific meaning or tone in the conversation.

Furthermore, there are two additional types of code-switching [20]. The first is situational code-switching, which occurs when a social setting is marked by a change in language transitions. This type reflects how speakers adapt their language use based on the context or situation they find themselves in. The second type is conversational code-switching, which occurs during conversations within utterances. In this form, speakers switch between languages metaphorically or conversationally, using terms or expressions from both languages to convey a specific meaning or tone in the conversation. A groundbreaking statistical language model for code-switching speech incorporates syntactic inversion constraints commonly observed in such speech. This model comprises a code-switch prediction model, a translation model, and a reconstruction model [21].

Code-switched prediction models learn from word-aligned parallel sentences to determine permissible code-switching points. The primary objective of a code-switched prediction model is to forecast whether there will be a code-switch between the current and following tokens [22]. This task involves analyzing the contextual cues and linguistic patterns to predict these transitional points in bilingual communication.

The application of neural-based models, specifically recurrent neural network language models (RNNLMs) and factored language models (FLMs), for language modeling in code-switching speech contexts is thoroughly examined in [23].

A new computational approach that employs the Matrix Language Frame theory to produce synthetic code-switching data and overcome the problem of data scarcity is introduced in a study by Lee et al. [24]. This method uses augmented parallel data to supplement the actual code-switching data. Similarly, the morphosyntactic characteristics of hybrid nominal phrases within a collection of Igbo-English intrasentential code-switching data are examined in [25], utilizing the Matrix Language Frame (MLF) model to analyze shifts between languages within a bilingual clause. Furthermore, a novel technique for acquiring the ability to produce code-switching sentences through parallel corpora is presented by Winata et al. [26]. This proposed model employs a Seq2Seq architecture, coupled with pointer networks, to synchronize and select words from monolingual sentences, creating grammatically correct code-switching

sentences.

A Sepedi-English code-switched speech corpus was created to analyze the factors influencing code-switching when designing ASR systems for Sepedi-English code-switched speech [27]. The methodology involved recording radio broadcasts, counting code-switched instances, and transcribing them to establish the Sepedi Prompted Code-Switched Corpus (SPCS).

End-to-end artificial voice recognition system that uses a transformer-based converter design architecture intended for code-switched speech recognition was presented in Dalmia et al. [28]. Their research suggests three changes to the standard paradigm to adequately address distinctive features of code-switching. To handle code-switching scenarios' minimal resource requirements, they first provide two auxiliary loss functions. Furthermore, to enhance the label encoder training specifically for intra-sentential code-switching, a unique mask-based training technique is presented that is combined with language ID information. To better recognize code-switching, they finally suggest a multilabel / multi-audio encoder structure that would use large monolingual speech corpora. Using the SEAME dataset, a readily available Mandarin – English code-switching corpus, the effectiveness of these suggested methods is shown.

In [12], they investigate the constructiveness of utilizing out-of-domain monolingual data to enhance ASR for code-switched speech. Their experiments focus on a newly introduced dataset comprising code-switched South African soap opera speech across five languages. Specifically, they examine whether incorporating monolingual data from unrelated, larger corpora can improve ASR accuracy for English–isiZulu code-switched speech. They train TDNN BLSTM acoustic models using different configurations of training data and explore the use of artificially-generated bilingual English–isiZulu text to augment language model training data. Despite differences between the soap opera and monolingual speech, the study concludes that incorporating monolingual out-of-domain data can improve speech recognition accuracy for English-isiZulu code-switched speech.

It is important to improve ASR models for code-switching while preserving monolingual accuracy since fine-tuning them on code-switched speech might negatively

impact performance on monolingual speech. Strong results are obtained from both code-switched and monolingual tests sets when models are trained on code-switched speech using the Learning Without Forgetting (LWF) framework without having access to the original training data. Regularization procedures are suggested to fine-tune models for code-switching while compromising monolingual accuracy when applicable, leading to improved WER relative to baselines using pooled data and straightforward fine-tuning [29].

In response to the lack of resources in this field, a spontaneous code-switched Egyptian Arabic-English speech corpus is created and made available, along with a code-switched ASR system. Both transformer-based and DNN-based hybrid E2E models are developed and used by the ASR system. A comprehensive comparison between these approaches is presented in the context of a low-resource, orthographically unstandardized, and morphologically rich language pair. Meanwhile, both the systems yield similar overall recognition results, each demonstrates unique strengths. Effective methods for integrating hypotheses from both systems at both sentence and word levels are also proposed, resulting in an overall WER improvement [30].

An E2E ASR pipeline tailored for code-switched speech, where a low-resourced language is combined with a high-resourced language, is presented by Yue et al. [31]. The challenge of low-resourcefulness in acoustic data, which particularly impacts E2E ASR systems, is addressed by integrating a specialized decoding scheme and applying neural network-based language model rescoring techniques to enhance transcription accuracy in archives containing code-switching Frisian-Dutch speech. To optimize the use of accessible linguistic materials in both languages, this approach uses a multi-graph decoding mechanism, generating distinct search areas for each monolingual section. Furthermore, limited in-domain code-switching text is used to further adjust a recurrent neural network that has been trained beforehand with cross-lingual embedding for language model rescoring.

2.3. Traditional Code-switching Speech Recognition Systems

Code-switching, the practice of switching between multiple languages within a single discourse, grant a significant challenge for speech recognition systems aiming

to accurately transcribe multilingual speech. Traditional methodologies have been instrumental in addressing this challenge, laying the foundation for the recognition and interpretation of code-switched speech. Additionally, these methods have been successfully applied to monolingual corpora, demonstrating their versatility and effectiveness. This section explores two fundamental techniques within traditional code-switching speech recognition systems: Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM).

2.4. Hidden Markov Model performance in ASR

The hidden Markov model (HMM) is a probabilistic model that can effectively capture the hidden states underlying observation data. In an HMM, observations at different time points are associated with a certain probability distribution corresponding to a possible hidden state. The mathematical foundations of HMMs were established by Baum and Petrie in 1966 [32].

Although Markov sources first developed and investigated in the late 1960s and early 1970s, their statistical applications have become more well-known in recent years [33]. HMMs provide a framework for modeling the joint probability of a collection of random variables, incorporating both observations and states [34]. They have been particularly successful in modeling the dynamics of carefully dictated speech [35]. However, when applied to conversational speech, HMMs often face significant performance degradation. When employed for model conversational speech, their performance deteriorates and there is a prevalent hypothesis that achieving satisfactory transcription performance on this type of data will require more advanced models.

When employed for model conversational speech, their performance diminishes, leading to a widespread belief that achieving satisfactory transcription accuracy on such data might necessitate the use of more advanced models. There is a common hypothesis that achieving acceptable transcription performance for this type of data will require more sophisticated models [35].

2.5. GMMs performance in ASR

The GMM (Gaussian Mixture Model) is a probabilistic model that represents a finite mixture of Gaussian distributions [34]. It is commonly used as a data clustering method where each cluster is modeled as a combination of different Gaussian components [36]. The goal of clustering with GMM is to assign observations to the appropriate components by estimating the parameters of each cluster [36].

GMMs have been employed in various domains, including image segmentation, audio classification, and instrument recognition [37]. These presuppose that the vector traits have a man and deviation Gaussian distribution. GMMs may accurately depict the feature distribution of a specific speaker by permitting a combination of these Gaussians, which qualifies them for speaker identification applications [38].

The application of GMMs in speaker recognition has been widely studied and demonstrated their efficiency in text-independent speaker recognition [39], [40], [41]. Recent studies have highlighted the effectiveness of GMMs in various pattern recognition techniques [42]. Advances in parameter estimation, computations, and scoring have further improved the performance of GMMs in speaker recognition tasks [38].

The GMM excels at soft clustering discrete observations. However, in speech processing, where observations occur sequentially, it faces a challenge: accurately determining transitions between clusters. Understanding the occurrence and co-occurrence of language phonemes requires careful analysis of training samples [43]. Their limitations in handling sequential data and modeling transitions between different linguistic components make them less suitable for code-switching tasks in speech processing.

CTC-based systems in ASR were trained multilingually, utilizing a global phonetics that are derived from the combined phonetics of each source language. Various language combinations and the inclusion of Language Feature Vectors (LFVs) were evaluated. While multilingual systems typically suffer in performance compared to monolingual ones, the proposed approach reduces this performance gap, whether using graphemes or phonemes. The system achieved the lowest token error rate (TER) and word error rate (WER) [44].

2.6. Deep Learning Code-switching Speech Recognition techniques

In the realm of multilingual communication, code-switching presents a fascinating challenge for speech recognition systems. Deep learning approaches have emerged as powerful tools to tackle this complexity, offering innovative solutions to recognize and transcribe code-switched speech. This section explores three prominent deep learning architectures for code-switching speech recognition: Speech Recognition using Deep Neural Networks (DNNs), Speech Recognition using RNNs and Convolutional Neural Networks (CNNs).

2.7. Speech Recognition using DNNs

DNN achieves notable accuracy improvements in various speech recognition tasks, primarily due to their deep and wide network structures with a vast number of parameters [45]. DNN models can perform well in tasks even without a deep understanding of the underlying complexity. However, if there is a good understanding of the task, there may be more efficient ways to achieve the same level of performance compared to using DNNs [46]. The DNN model is a neural network architecture characterized by its feed-forward structure and the inclusion of multiple hidden layers. Optimizing DNNs with numerous hidden layers can be challenging, sometimes requiring extension beyond the boundaries of the test dataset. Due to the sequential nature of speech signals, DNNs encounter difficulty in directly modeling them [47]. DNN models with multiple hidden layers are capable of capturing complex patterns in data. However, such models can be challenging to optimize effectively, especially when dealing with diverse linguistic features present in code-switched data.

2.8. Recurrent Neural Networks-based in Speech Recognition

Recurrent Neural Networks (RNNs) have gained significant popularity and have been effectively utilized in various domains, including speech recognition, machine translation, and image captioning [48]. The fundamental characteristic of RNNs lies in their ability to process sequential data, where they maintain a state that encapsulates the

information encountered so far [49]. This internal memory and chain-like structure allow RNNs to capture dependencies and patterns in sequential data. However, RNNs face challenges during training due to the vanishing or exploding gradient problems [50].

These issues arise when gradients propagated through the network either shrink exponentially or grow uncontrollably as they traverse through the recurrent loops [51]. The vanishing gradient problem causes RNNs to struggle in capturing long-term dependencies, as gradients diminish over time. Conversely, the exploding gradient problem leads to unstable and ineffective training.

2.9. Convolutional Neural Networks-based in Speech Recognition

CNNs show effectiveness in reducing spectral variations and capturing correlations within acoustic features for ASR [52]. The CNN is a type of feed-forward artificial neural network that incorporates convolutional and pooling layers. CNNs require input data to be structured in a specific format. When processing speech signals, this involves using features arranged either along the frequency or time dimensions to ensure that the convolution operation is applied correctly [53].

Building ASR systems competitive with standards approaches is made possible by the ability to explicitly model the correlation of phonemes and uncompressed signal from speech within the CNN framework. Analysis has shown that within the initial two convolutional layers, the CNN acquires and represents the phone-specific spectral envelope details from 2-4 ms speech segments. It has been concluded that the CNN-based approach yields ASR trends similar to those of standard short-term spectral-based ASR systems under mismatched (noisy) conditions, with the CNN-based approach being more robust [54].

2.10. End-to-End Code-Switching Speech Recognition System

In the context of code-switching speech recognition, E2E systems have appeared as a promising frontier, offering a holistic approach to transcribing multilingual speech. This section explores two cutting-edge methodologies within E2E systems: Listen, Attend,

and Spell (LAS) and CTC. These techniques represent a departure from traditional models, aiming to directly map acoustic signals to code-switched text, providing a streamlined and efficient solution to the complexities of code-switching.

2.11. Listen, Attend, and Spell in End-to-End

A neural network called the Listen, Attend, and Spell (LAS) model is used to translate oral speech into texts that can be written. The model successfully incorporates all essential components of a speech recognition system. It consists of two primary parts: a speller, an attention-based recurrent network decoder that produces character sequences as output, and a listener, which functions as a tapered recurrent network encoder taking filter bank spectra as input [55]. Importantly, this network produces character sequences while avoiding assumptions of their independence.

According to [56], while simplifying the training and decoding procedures, this approach poses challenges for an intergrate model that can be used to adapt when inconsistencies that may arise between training and testing data, especially in scenarios with dynamic changes in this information.

On the opposing side of the spectrum, we have non-streaming models like LAS, which have demonstrated superior performance over traditional ASR systems [57]. However, it's worth noting that LAS models aren't inherently designed for streaming since they necessitate processing the entire audio segment [58].

The outcomes generated by LAS-based models don't explicitly hinge on the audio's length. Instead, these models depend on token generation to indicate sentence endings. This characteristic might lead the LAS model to produce extraneous and unwanted tokens or potentially overlook certain tokens. To address this concern, a solution involves employing a CTC-based model that's trained to predict phoneme units. This CTC model is then utilized to reevaluate and rescore the result beams generated by the LAS model [59].

There are two main types of E2E architectures namely, CTC and attention-based methods [60]. Attention-based methods consist of an encoder network and an attention-based decoder. The encoder network maps the acoustic speech input into a high-level

representation, while the attention-based decoder recognizes symbols based on previous predictions. By utilizing an attention mechanism, the model can selectively focus on specific parts of the input from the encoder while disregarding irrelevant information [61].

Attention-based models have found wide application in sequence-to-sequence learning systems, achieving impressive results in various natural language understanding tasks. However, these models face challenges in efficient execution due to their memory-bound nature, caused by a large number of parameters [62].

2.12. Connectionist Temporal Classification in End-to-End

E2E training methods, such as CTC, have revolutionized ASR by allowing the training of RNNs for sequence labeling tasks without requiring knowledge of the input-output alignment. This has opened up possibilities for solving problems where the alignment is unknown [63]. The CTC approach, proposed in [64], is a technique that utilizes RNNs in tasks of sequence labeling wherein the alignment is unknown between input sequences and target labels [64]. The CTC objective allows for training E2E systems that directly predict grapheme sequences without requiring frame-level alignments of the target labels during training [60].

CTC is a loss function designed for sequence labeling tasks involving inputs and label sequences of variable lengths [65]. In acoustic modeling, CTC eliminates the need for pre-defined frame-level labels by autonomously learning alignments between speech frames and their corresponding label sequences. CTC outputs a character for each frame of the input, making the output sequence as long as the input sequence. Subsequently, a collapsing function is applied to merge sequences of identical characters, resulting in a shorter sequence [66].

The primary benefit of CTC is that it has the ability to eliminate the need for data segmentation alignment. This enables deep learning techniques like CNNs and RNNs to play a more significant role in the ASR process [67]. Unlike traditional models that often output phonemes or other small units, CTC directly outputs the final target form without the need for additional processing, simplifying the construction and training of

E2E models [67]. This allows a single network structure to map the input sequence directly to the label sequence, facilitating E2E speech recognition.

Moreover, to its simplicity and efficiency, the use of the intermediate CTC loss provides regularization during training, enhancing performance, and requires minimal code modification [1]. Furthermore, the overhead during both the training and inference processes is negligible, making CTC a practical choice for ASR systems.

CTC-based systems in ASR were trained multilingually, utilizing a global phone set derived from the combined phone sets of each source language. Various language combinations and the inclusion of Language Feature Vectors (LFVs) were evaluated. While multilingual systems typically suffer in performance compared to monolingual ones, the proposed approach reduces this performance gap, whether using graphemes or phonemes, and the system achieved the lowest token error rate (TER) and word error rate (WER) [44].

Enhancing code-switched ASR systems in low-resource settings by employing data augmentation techniques with code-switched Text-to-Speech (TTS) synthesis is explored in [68]. The approach involves using Mixup, a method for generating new training samples through linear interpolation, applied to both TTS and real speech samples. Additionally, a novel loss function is introduced, used alongside TTS samples to promote code-switched predictions. The findings demonstrate significant enhancements in ASR performance, with absolute WER reductions and noticeable improvements in code-switching accuracy, offering promising avenues for improving code-switched ASR tasks, particularly for Hindi-English language pairs in low-resource environments.

Authors in [69] present an approach to enhance the accuracy of self-attention mechanisms when dealing with long sequence data. It tackles the issue of accuracy decline in self-attention due to differences in lengths between training and inference data segments. The approach is applied to CTC-based ASR. Experimental results on the Corpus of Spontaneous Japanese (CSJ) and TEDLIUM 3 benchmarks exhibit a significant enhancement in accuracy for long sequence data, eliminating the necessity for windowing techniques.

While usually demanding extensive training data, E2E ASR is comparatively straight-

forward to implement both conceptually and linguistically. An E2E ASR system is introduced to transcribe code-switched speech in Bantu languages and English using the South African Corpus of Multilingual Code-switched Soap Opera Speech (Soap Opera corpus). Despite the low-resource nature of the dataset, containing 21 hours of speech in four Bantu languages and English, typically requiring thousands of hours for training, the authors employ fine-tuning and evaluation of XLSR-53 wav2vec 2.0 models pre-trained on data from 53 languages. Fine-tuning and evaluation are conducted specifically on the Soap Opera corpus data. Furthermore, the study explores pre-training and fine-tuning wav2vec 2.0 models on the Soap Opera corpus, incorporating data from the NCHLT corpus during pre-training in subsequent experiments. XLSR-53 models fine-tuned on Soap Opera corpus data demonstrate the best performance among the tested models [70].

A CTC-based E2E ASR model tailored for intra-sentential English-Mandarin code-switching was introduced, undergoing joint training on monolingual datasets followed by fine-tuning using the mixed-language corpus. For decoding, beam search is employed, integrating CTC predictions with language model scores. This approach effectively utilizes monolingual corpora, detects language transitions, and results in an improvement in Character Error Rate (CER) [71].

An adaptation technique for E2E speech recognition systems involves integrating multiple ASR hypotheses into the computation of the CTC loss function, helping to mitigate the impact of errors in ASR hypotheses on CTC loss calculations. The method calculates the CTC loss using diverse ASR hypotheses derived from decoding unlabeled data in semi-supervised adaptation scenarios where some adaptation data lack labels. Experiments conducted in clean and multi-condition training settings and multi-condition training data, and subsequently tested on Aurora-4 test data, illustrate considerable relative reductions in WER [72].

2.13. Speech Recognition System Evaluation Matrix

Evaluating the performance of speech recognition systems is crucial for assessing their accuracy and effectiveness in transcribing speech, especially in the context of code-switching. This section explores key evaluation metrics used in assessing

speech recognition systems: Phone Error Rate (PER), Letter Error Rate (LER), WER, and Character Error Rate (CER). By understanding these metrics, we gain insights into how code-switching speech recognition systems are evaluated and how their performance is quantified.

2.14. Phone Error Rate in Speech Recognition System

Phone recognition systems are typically evaluated using the phone error rate (PER), which calculates the minimum edit distance encompassing substitution, deletion, and insertion errors [73]. While PER allows for comparisons and rankings of phone recognizers, it may not provide sufficient detail to comprehend the specific nature of errors and nuances in phone recognition. Studies such as [2] and [44] have utilized PER as a performance metric. The PER metric considers all discrepancies between the recognizer's hypothesis and the manually annotated reference at the phone level, defined as follows:

$$PER = \frac{I + S + D}{C + S + D} \tag{2.1}$$

With C, I, S, D respectively referring to the number of correct detections, insertions, substitutions and deletions [74].

2.15. Letter Error Rate in Speech Recognition System

This is a natural measure for tasks such as speech or handwriting recognition where the aim is to minimize the rate of transcription mistakes [64]. Letter Error Rate (LER) is calculated by computing the edit distance between the hypothesized words and the reference transcription. The edit distance is the number of insertions, deletions, and substitutions needed to convert the hypothesized words into the reference transcription. LER measures the error rate at the letter level with the formula below:

$$LER = \frac{D - S - I}{N} \tag{2.2}$$

where D is the total number of deletions in the recognised strings, S is the total number of substitutions, I the total number of insertions, and N the total number of labels in the test set [75].

2.16. Word Error Rate in Speech Recognition System

WER is calculated by computing the edit distance between the hypothesized words and the reference transcription. The edit distance is the number of insertions, deletions, and substitutions needed to convert the hypothesized words into the reference transcription. The WER is computed as the ratio of the edit distance to the total number of words in the reference transcription [5]. On the other hand, in [5], [9], and [10], WER was employed to measure the effectiveness of decoding using a language model with CTC.

WER measures the difference between the recognized transcription and the reference transcription in terms of the number of word errors. It measures the percentage of incorrectly transcribed words (Substitutions (S), Insertions (I), Deletions (D)). It is defined as follows:

$$WER = \frac{S + D + I}{N} \tag{2.3}$$

where N is the number of words in the reference text [76].

2.17. Character Error Rate in Speech Recognition System

The attention-based encoder-decoder learns a direct mapping from acoustic frames to character sequences. At each output time step, the model generates a character based on the input data and the previous characters in the sequence. Because the attention mechanism does not rely on conditional independence assumptions, it has frequently been observed to enhance Character Error Rate (CER) [77]. In [11] and [67], evaluations were performed using the CER as the performance metric.

CER measures the error rate at the character level. CER depends on the relative sizes of the number of errors and correct words. Since there are hopefully many more correct words than errors, CER is not very satisfying to measure error detection

performance [78]. Formula:

$$CER = \frac{S + D + I}{N} = \frac{Numbero\,f\,incorrectclassi\,f\,ications}{Hypothesizedwords} \tag{2.4}$$

where: S is the number of substitutions which are characters in the reference that are incorrectly recognized, D is the number of deletions which are characters missing in the recognized transcription compared to the reference, I is the number of insertions which are extra characters present in the recognized transcription compared to the reference, and N is the total number of characters in the reference.

2.18. Chapter Summary

Despite significant advancements in speech recognition technologies, notable research gaps persist, particularly in the context of code-switching and low-resource environments. Current approaches, including DNNs, RNNs, and CNNs, often struggle with robustness in diverse linguistic features and real-time processing. Additionally, E2E systems like LAS face challenges in adapting to variations in training and testing datasets. To address these gaps, this research focuses on developing hybrid models that integrate the strengths of RNNs and CNNs, using CTC techniques that enable the systems to predict variable-length output sequences without requiring frame-level alignment. This allows for more efficient handling of unaligned data in code-switched scenarios. Furthermore, exploring innovative data augmentation methods for low-resource, code-switched languages will enhance performance and accuracy.

3. METHODOLOGY

3.1. Introduction

This chapter offers a comprehensive exploration of the methodology employed to develop an ASR system using the CTC approach. We discuss the specific system parameters and neural network architecture used, shedding light on the technical basis of our research. The goal is to clarify the framework behind the creation of an innovative Sepedi-English code-switched ASR system, enabling a deeper understanding of the techniques and processes involved. The following sections will delve deeper into the methodology, covering data preprocessing, network architecture design, training procedures, and evaluation metrics.

3.2. Code-Switched Speech Corpus

This research utilizes the Sepedi Prompted Code Switching [1] (SPCS) corpus as described by Modipa et al. [27]. we chose to use this corpus in this research because it is the only sepedi code switched data that is available. The SPCS Speech Corpus comprises a curated collection of spoken language data specifically assembled and annotated for speech research purposes. The prompts used in this corpus are derived from transcribed code-switched radio data. The speech recordings were gathered using Woefzela, a smartphone-based tool developed locally for broadband speech data collection [79]. The SPCS corpus includes recordings from a diverse cohort of Sepedi speakers aged between 17 and 27 years.

The SPCS Sepedi data was collected in a form of recordings and then transcribed and segmented into smaller units of speech called utterances. Directories have audio files for all the recordings and transcriptions. Out of all the audio files, 7615 are captured by men, and 4771 are captured by women, producing a gender ratio of 60% males and 40% females.The speech corpus comprises around 12359 audio files

[1] https://repo.sadilar.org/handle/20.500.12185/530?show=full

which has a total duration of approximately 10 hours.

The dataset is segmented into three groups: the training, validation, and testing sets. Specifically, it is divided into proportions of 70% for training, 20% for validation, and 10% for testing. Table 3.1 visualizes how utterances are distributed among these three distinct sets: training, validation, and testing.

Table 3.1: The number of utterances of the SPCS corpus

Set	Number of utterances	Number of speakers
Training	6644	12
Validation	3014	4
Testing	2701	4

3.3. Model setup using different dataset

The NCHLT Sepedi corpus primarily consists of speech prompted in Sepedi, interspersed with English speech, resulting in a code-switched corpus. Our analysis specifically targets the testing data from the NCHLT corpus, which includes a total of 8 speakers of which 4 are females and 4 are males. Additionally, we include testing data from the SPCS corpus, which involves four speakers—two females and two males. This distribution ensures a balanced representation of genders. Details of the test data distribution can be found in Table 3.2.

Table 3.2: Gender distribution in SPCS and NCHLT test data

Test data	SPCS speakers	NCHLT speakers
Female	2	4
Male	2	4

To evaluate the model's performance, we utilized both the NCHLT dataset and the SPCS corpus, as shown in Table 3.3. This approach enabled a comprehensive assessment of the model's capabilities across diverse datasets, considering the number of utterances in the testing data. The data indicates a relatively balanced distribution of utterances between male and female speakers in both datasets, with males slightly more represented in the NCHLT dataset and females slightly more represented in

the SPCS dataset. Additionally, the NCHLT test data has a higher total number of utterances compared to the SPCS test data.

Table 3.3: The distribution of utterances in SPCS and NCHLT test data by gender

Test data	Number of SPCS utterances	Number of NCHLT utterances
Female	610	1382
Male	516	1447
Total	**1126**	**2829**

In Table 3.4, we show the distribution of Sepedi words to English words in the corpus. It is clear that the corpus is dominated by Sepedi words. However, the English words were equally present. Only a small number of words which do not form part of either Sepedi or English were also found in the corpus. These words are known as bilingual words, which are words that do not fit the grammatical pattern of either of the languages.

Table 3.4: The SPCS Sepedi-English words distribution

Language	Number of words
Sepedi	384
English	346
Other	58

Table 3.5 presents the distribution of sentences in the SPCS (Sepedi-English) corpus, categorizing them into Sepedi, English, and Code-switched sentences. The data shows that there are 139 sentences written entirely in Sepedi and 542 sentences written entirely in English. Notably, the majority of the sentences, amounting to 11,678, involve code-switching, where both Sepedi and English are used within the same sentence. This significant prevalence of code-switched sentences highlights the bilingual nature of the corpus and indicates a high level of linguistic interplay between Sepedi and English within the dataset.

Figure 3.1 shows the frequency of frequently appearing words in the SPCS corpus. The corpus is dominated by words of length two and few occurrences of single-character words. These words with the highest frequency in the corpus are referred

Table 3.5: The SPCS number of sentences distribution

Language	Number of sentences
Sepedi	139
English	542
Code-switched	11678

to as connectors. Connectors contribute to the coherence and cohesion of sentences and texts, serving to bridge gaps and uphold a logical and comprehensible structure in the written content [80]. Also, the most frequent words are Sepedi words.

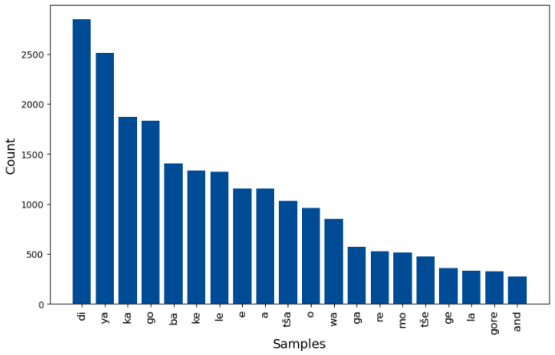

Figure 3.1: Distribution of frequently appearing words

We also analyzed the occurrences of the words in the sentences and counted the frequency of the sentences. Figure 3.2 shows the number of words per sentence and the respective frequencies of the sentences. Sentences with only one word occur 27 times whereas the corpus is dominated by sentences with a length of five words which has 7385 occurrences.

The majority of sentences have a length of 27 characters (see Figure 3.3). Interestingly, the longest sentences, consisting of 33 characters, have the lowest counts. On the other hand, sentences with lengths between 27 and 28 characters exhibit the highest frequencies in the dataset.

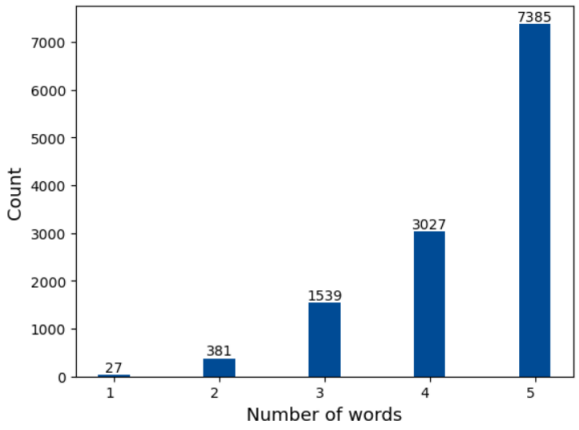

Figure 3.2: Distribution of words per sentence

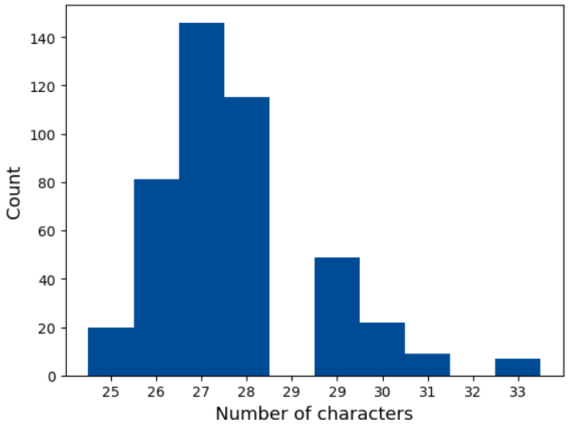

Figure 3.3: Distribution of characters per sentence

3.4. System Development Process

By following a sequence of steps, we have built an ASR system using the CTC approach on Kaggle environment [2]. This process encompasses several pivotal phases: pre-processing, dataset generation, model development, training, and evaluation. Below is a brief overview of each stage.

The first step involves installing essential Python libraries and modules necessary for building the ASR system. This includes integrating deep learning frameworks such as TensorFlow and Keras. In the preprocessing phase, we carefully commenced by defining the valid set of characters that can appear within the transcriptions of our

[2]https://www.kaggle.com/

speech data. This resultant character set is then stored within a designated variable. Following this initial step, we proceed to establish two essential Keras layers. Keras, as a widely used deep learning framework that offers an abstraction layer for constructing and training neural networks. In Keras, a layer represents a fundamental component of a neural network. Each layer is responsible for performing specific operations and transformations on the input data. These operations are essential for learning and modeling complex patterns within the data. Each keras has a distinct role in facilitating the transformation between characters.

The first of these layers assumes the pivotal role of translating characters into their corresponding integer representations. It relies on the vocabulary parameter, a configuration designed to encompass the predetermined characters that our ASR system is expected to encounter during transcription. Any character values found in the input data that fall outside this pre-defined vocabulary are intelligently mapped to an empty string, ensuring that our system can gracefully handle unexpected inputs without disruption.

Conversely, the second Keras layer functions as the counterpart to the first. Its primary function is to perform the inverse mapping, converting integers back to their original character forms. This is achieved through the utilization of the same 'vocabulary' information, thus providing a comprehensive and seamless bidirectional transformation between characters and their corresponding integer representations. These two layers play a crucial role in preparing the transcription data for training and decoding within our ASR system.

In the process of creating dataset objects, we organize our data into three distinct sets: training, validation, and testing. Each of these subsets plays a unique and crucial role. The training set, which is the largest, serves as the foundation for the model's learning process. It's in this set that the model gains insights into patterns and refines its recognition capabilities. The validation set, separate from training, acts as a critical tool for parameter tuning and performance assessment during the training process. The testing set, equally important, provides the ultimate evaluation platform. We measure the ability of the model to generalize to unseen data, and in our case, this set introduces diversity by drawing samples from two different corpora.

Once our data is organized, we process each encoded sample within the dataset. Here, each encoded sample within the dataset undergoes a series of tasks using the map method. These tasks encompass various fundamental processes. Firstly, the method reads the audio files associated with the dataset samples, granting access to the raw audio data. Subsequently, the computation of spectrograms ensues. These spectrograms capture the essential spectral features that underlie speech recognition. Data normalization is the next step, ensuring uniformity and eliminating inconsistencies in the input data. Additionally, the mapping of transcriptions into sequences of integers is executed, making the text data amenable for ASR model processing, as these models typically require numeric input.

To optimize the dataset for efficient model training, we apply the technique of batching. This process involves grouping multiple samples into batches, a strategy that expedites parallel data processing, enhances efficiency, and maintains consistency. Furthermore, to maximize resource utilization and expedite the training process, we employ the pre-fetch method. This technique cleverly overlaps data preprocessing with model training, ensuring that while one batch is being processed, the next is being pre-processed in the background.

In parallel, the creation of a separate test dataset is a vital component in the system development process. This dataset, unlike the training and validation sets, is constructed using test file names and their associated transcriptions. It plays a pivotal role in the evaluation of the model's performance. By assessing the model's accuracy on unseen data, it simulates real-world conditions where the model encounters novel and unpredictable inputs, ultimately ensuring the robustness of the ASR system.

We implemented a speech recognition model inspired by the DeepSpeech2 architecture, utilizing a combination of CNNs and RNNs. The input to the model is in the form of spectrograms, which are reshaped to add a channel dimension, allowing for compatibility with CNN layers. Two convolutional layers are applied to extract temporal and spectral features from the input. Batch normalization follows each convolutional layer to stabilize the learning process, and ReLU activations are used to introduce non-linearity.

After the convolutional layers, the output is reshaped to prepare it for sequential

modeling by the RNN layers. The RNN component of the model consists of multiple bidirectional Gated Recurrent Unit (GRU) layers, which are employed to capture both forward and backward temporal dependencies in the input sequence. To prevent overfitting, dropout layers are applied between the RNN layers. The output of the recurrent layers is passed through a fully connected dense layer with a non-linear activation and additional dropout for regularization.

The final layer is a softmax-activated classification layer that outputs a probability distribution over possible character predictions, including an extra label for the blank character used in the CTC loss function. This architecture, combined with the use of CTC loss, allows the model to handle variable-length input and output sequences. The model is trained using an appropriate optimizer to ensure efficient convergence.

3.5. Training

During the training phase, the neural network undergoes a rigorous learning process. It receives batches of speech recordings, each accompanied by their corresponding transcriptions. The primary objective of the network is to determine and internalize the intricate patterns existing within the data. These patterns represent the relationships between audio features and their associated textual representations.

To achieve this objective, the network engages in a continual process of parameter adjustment. It strives to minimize the disparity between the transcriptions it generates and the actual, ground truth transcriptions provided in the dataset. This optimization is achieved through a technique known as back-propagation. In this process, the model calculates the gradient of a loss function concerning its parameters. The loss function quantifies the disparity between predicted transcriptions and the actual ones. The gradient represents the direction and magnitude of change needed for each parameter to reduce this disparity.

The training phase is iterative and unfolds over multiple epochs. During each epoch, the model handles a batch of data, calculates the loss, and updates its parameters using the computed gradient. This iterative process of data processing, loss calculation, and parameter adjustment continues for a set number of epochs, which

is commonly specified in advance based on experimentation and model convergence characteristics. Convergence occurs when the model's performance stabilizes, and further training does not significantly improve its accuracy or reduce the loss. Alternatively, training may also conclude when a specific time limit for training has been reached, depending on the research requirements.

In this study we utilized a total of 150 epochs for training. This number represents the extent of the iterative learning process and, along with the other training details, is crucial for understanding how the system learned to recognize and transcribe speech patterns. The training phase is where the neural network evolves, adapting its internal parameters to make accurate predictions based on the patterns it discovers in the training data.

3.6. Regularization effects on Code-Switched speech

Regularization is essential to improve performance and generalization of ASR systems dealing with code-switched speech. This technique helps the model generalize better to unseen code-switched speech by preventing overfitting.

To prevent overfitting during training of deep neural networks, dropout is a commonly used regularization technique. By randomly dropping units during training, dropout forces the network to learn more robust features. This technique can be particularly beneficial for code-switched speech recognition, where the model needs to generalize across different languages and switching patterns.

Dropout randomly drops a percentage of neurons during each training iteration, forcing the network to learn redundant representations. This prevents the network from relying too heavily on specific neurons, thus enhancing its ability to generalize from the training data. It helps the model to learn more robust features that are useful across different languages and switching points, mitigating the risk of overfitting to the peculiarities of the training data.

The dropout rate in a CTC-based ASR system plays a critical role in regularizing the model, enhancing its generalization, and maintaining training stability. By effectively balancing the dropout rate and combining it with other regularization techniques,

you can significantly improve the performance and robustness of ASR systems in handling code-switched speech.

3.7. Learning rate effects on Code-Switched speech

The learning rate is a fundamental hyperparameter in training neural networks, controlling the size of the steps taken in the parameter space during optimization. High learning rate accelerates convergence but can lead to overshooting the minima, causing the training process to diverge or result in a model that does not converge to a good solution. Low learning rate slows down the convergence process, making the training more stable and precise but requiring more epochs to reach the optimal solution. The optimal learning rate strikes a balance between convergence speed and stability. It ensures the model updates parameters effectively without overshooting minima.

In gradient descent optimization, the step size is determined by the learning rate. It controls how much the model's parameters are updated in response to the gradients calculated during backpropagation. The learning rate in a CTC-based ASR system is crucial for balancing the training process's convergence speed, stability, and generalization. It influences how effectively the model learns to handle the complexities of code-switched speech, capturing the necessary phonetic and linguistic features.

3.8. Effects of the complexity using kernel size on Code-Switched speech

The kernel size in a convolutional layer refers to the dimensions of the filter applied to the input data. They greatly influence the ASR system's capability to process code-switched speech. The choice of kernel size affects the capacity of the model to capture local and global features, balancing the complexity and ensuring effective learning of the diverse and intricate patterns in code-switched data.

The kernel size in convolutional layers is pivotal for extracting relevant features from input speech data. Small kernels are effective for capturing fine-grained, local features like edges and phonetic details, making them ideal for early layers to detect basic sound elements. Larger kernels capture broader, global features, enabling

the network to recognize more extensive patterns and contextual information, which is crucial for understanding complex speech phenomena, including code-switching. Varying kernel sizes in the network builds a hierarchical feature representation, from low-level details to high-level abstractions, enhancing its ability to learn both local and global dependencies in the speech signal.

3.9. Effects of Filter Quantity on Complexity on Code-Switched speech

Filters are the components in convolutional layers that slide over the input data, performing convolution operations to extract features. Each filter is responsible for detecting specific patterns or features in the input data. The quantity of filters significantly impacts its complexity and effectiveness in handling code-switched speech. Increasing the number of filters enhances the capacity of the model to learn diverse and complex features, crucial for recognizing the nuanced patterns in code-switched data. Increasing filters adds more capacity to the model, enabling it to learn and represent more complex features.

The number of filters in convolutional layers is essential in shaping the ability of the model to capture and learn features from the speech signal. Adding more filters enhances the model's capacity to extract a diverse set of features, enabling it to detect a wide range of patterns, from simple phonetic details to complex linguistic structures.

3.10. Effects of the complexity using RNN on Code-Switched speech

RNN layers play a crucial role in capturing temporal dependencies within the speech signal and text by processing sequential input. RNNs operate on data in sequential form, unlike feed forward neural networks that handle fixed-length vectors. In RNNs, each vector in the sequence is processed based on the hidden state from the preceding step.

RNN layers maintain a hidden state that evolves over time as each audio frame is processed. This hidden state serves as a memory of the past inputs, allowing the network to capture temporal dependencies in the speech signal. This temporal modeling is crucial for understanding the context and timing of phoneme transitions

within spoken words.

RNN layers in ASR systems using CTC, the network learn to align the input audio frames with the corresponding output labels, the Gated Recurrent Unit (GRU) variant is used for learning such alignments over long sequences of data.

The number of RNN layers in an ASR system refers to the depth of the recurrent neural network architecture employed in the model. Each layer in the RNN stack processes the input sequentially, passing its output to the next layer. Increasing the number of layers in the RNN stack allows the model to capture more complex temporal dependencies and hierarchical features in the input data.

3.11. Evaluation

Throughout the evaluation phase, a callback function is invoked after each epoch to compute the WER and showcase a subset of transcriptions from the validation set. This approach allows continuous assessment of the model's performance with data that was not used during training, enabling adjustments to the model or hyperparameters as needed. Additionally, the behavior of the model is evaluated using data from two separate corpora.

In the model.fit function, the validation data parameter specifies the dataset used for evaluation during training. Specifically, the WER, as defined in Equation 2.3, measures transcription accuracy by quantifying discrepancies between predicted words and their actual references [81].

To compute the CTC loss, the function is represented by Equation 3.1:

$$P_{CTC}(Y|X) = \sum_{A \in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t|h_t) \tag{3.1}$$

For a given input pair (X, Y), all potential alignments that could yield the output sequence Y must be considered. The formula sums the probabilities of generating each label at each time step, considering the corresponding hidden state of the model, as outlined in [81].

For each alignment A, we calculate the likelihood of generating each label at every

time step, denoted as $p(a_t|h_t)$. Here, the output sequence Y consists of labels (such as characters or phonemes) with length denoted as T, and the input sequence X represents the model's input. A valid alignment A maps the input sequence X to the output sequence Y, while $B^{-1}(Y)$ represents the set of all valid alignments resulting in output sequence Y. By aggregating the probabilities of generating each label across all potential alignments, we derive the CTC loss function.

We employed a greedy decoding strategy call as a fundamental component of the system. Greedy decoding is a straightforward and widely used approach for transcribing speech data. This approach focuses on selecting the most likely transcription output at each time step without considering future context or dependencies. It operates under the assumption that at each moment, the system should choose the most probable character or subword unit based on the current acoustic features and the model's internal state.

To complement this decoding strategy, we also utilized the CTC Loss function during evaluation. The CTCLoss function is essential for training and evaluating ASR models, as it allows the model to align the predicted transcriptions with the ground truth transcriptions effectively, especially when dealing with sequences that contain varying lengths and code-switching phenomena.

With this combination of greedy decoding and the CTCLoss function, the system demonstrated its capability to transcribe spoken language efficiently and accurately. Greedy decoding, while being relatively straightforward, offers a strong foundation for ASR, and the CTCLoss function plays a pivotal role in refining the alignment and transcription quality. Together, these components formed the core of our system's decoding and evaluation methodology, enabling it to achieve the desired transcription accuracy.

3.12. Chapter Summary

In this chapter, we introduced the primary objectives and the significance of our research. We explained the data used in the study and how it was distributed. Subsequently, we presented a detailed breakdown of the preprocessing phase. This phase

served as the fundamental step for preparing our data to be amenable for ASR model training.

We also examined the creation of dataset objects, an essential process that involved data splitting, thorough data preprocessing, and optimization techniques like batching and pre-fetching.Greedy decoding was highlighted as the chosen approach for transcribing speech data.

Moreover, we discussed the use of the CTCLoss function in the evaluation of our ASR system, emphasizing its critical role in aligning predicted transcriptions with ground truth transcriptions, especially in the context of code-switching and varying sequence lengths. This chapter summary encapsulates the core methodology of our study, reflecting the systematic and rigorous approach undertaken to develop an effective ASR system for Sepedi-English code-switched speech.

4. EXPERIMENTAL RESULTS

4.1. Introduction

The objective of this chapter is to analyze the performance of the ASR system using the CTC approach. Various scenarios are examined during this evaluation. We assessed the performance based on several parameters, including the impact of regularization on code-switched speech, the influence of learning rates on code-switched speech, and the effects of complexity on code-switched speech with respect to kernel size, the number of filters, and the RNN architecture. By systematically evaluating these parameters, the chapter aims to provide a comprehensive understanding of how different factors impact the ASR model's performance, particularly for code-switched speech recognition. The intention is to determine how complex the ASR model should be to handle the CS speech patterns. Also, the generalizability of the model to handle sepedi only speech patterns. The results below illustrate the loss across 100 and 150 epochs for both the training and validation sets. The training loss refers to measurements on the training dataset, while the validation loss refers to measurements on the validation dataset. Validation loss is a crucial metric indicating how well the model generalizes to unseen data. Additionally, the results analyze the WER on the validation set for the SPCS dataset and on the testing set for both the SPCS and NCHLT Sepedi corpus, evaluating the model's generalizability across these datasets.

4.2. Regularization effects on Code-Switched

This section presents and discusses the effectiveness of the ASR model employing the CTC approach with different dropout rates. The dropout rate is a hyperparameter that specifies the fraction of neurons to be randomly dropped during training. During each training step, an experiment is conducted where a set percentage of neurons in a particular layer are randomly chosen, and their activations are set to zero. The percentages used in this experiment are 10%, 30%, and 50%. This means that the contribution of these neurons to the forward and backward passes is temporarily re-

moved. Dropout is only active during the training phase. During validation and testing, all neurons are used, but their activations are scaled by the dropout rate to ensure the overall output level remains consistent.

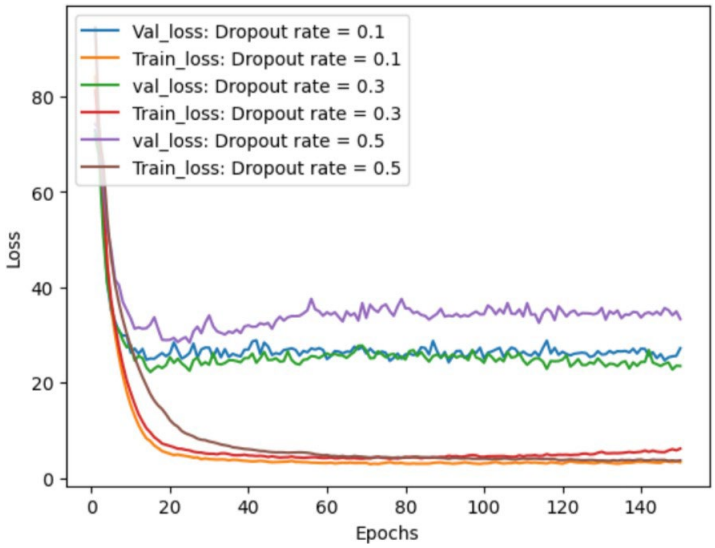

Figure 4.1: SPCS loss values for various dropout rate on validation set

Figure 4.1 presents the trends in loss over 150 epochs when using training and validation datasets. The loss exhibits a downward trajectory as the number of epochs progresses, punctuated by occasional fluctuations denoted by spikes. When the dropout rate of 0.3 is used, the model yields the lowest validation loss of 22.18 at 14 epochs. This suggests that employing the dropout rate of 0.3, effectively enhances generalization by mitigating overfitting to some degree as the performance remains consistent until epoch 150. The plateau of the graph such that no effect was observed beyond epoch 20. The loss of 24.21 was obtained with the 0.1 dropout rate and 28.35 with the 0.5 dropout rate, respectively.

These results show that employing a dropout rate of 0.3 achieves an optimal state from acquiring knowledge from the training dataset to extrapolating to unfamiliar data. However, when the dropout rate is decreased to 0.1 and changing it to 0.5 lead to increased validation losses, suggesting diminishing benefits and the possibility of overfitting. These results underscore the importance of carefully adjusting the dropout rate to optimize model performance and prevent overfitting. Notably, instances of abrupt spikes in validation loss indicate challenges when applying knowledge to new, unseen data at specific epochs, particularly evident in the graph with a dropout rate of 0.5.

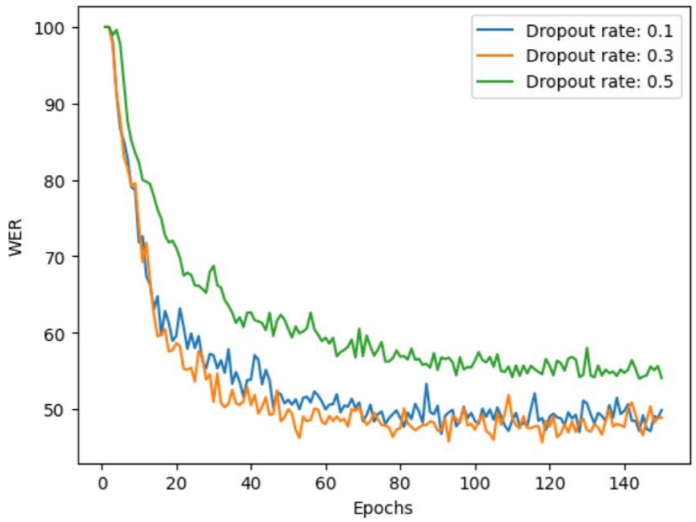

Figure 4.2: SPCS WER for various dropout rates on validation set

Figure 4.2 depicts the WER over varying number of dropout rates when using the SPCS corpus throughout the testing phase. The lowest WER, of 45.64% at epoch 117 which does not correspond to the epoch of the lowest loss. The lowest loss at an earlier epoch and the lowest WER at a later epoch indicate that the system initially learned the alignment well but required more training to optimize its real-world performance in terms of transcription accuracy when employing a 0.3 dropout rate. Conversely, utilizing 0.1 dropout rate results in a slight increase in WER to 49.15%, while opting for 0.5 dropout rate leads to a remarkably higher WER of 53.96%. When the dropout rate is 0.3, the model performed optimally suggesting that this level of regularization effectively balanced between preventing overfitting and maintaining model capacity for learning. However, when the dropout rate was reduced to 0.1 and changed to 0.5, the performance showed a slight degradation, particularly noted in the rise of WER. This indicates that the choice of dropout rate significantly influences the way the model generalizes from the data used for training to unseen samples.

A dropout rate of 0.1 suggests a minimal regularization approach, allowing the model to capture more intricate patterns from the training data. However, this led to a slight increase in WER, suggesting that the model may have started to overfit to the training set. On the other hand, a dropout rate of 0.5 indicates a stronger regularization strategy, which should prevent overfitting by encouraging the model to learn more robust features. Nevertheless, the observed rise in WER suggests

that excessive dropout may have constrained the model's learning capacity too much, hindering its ability to generalize effectively. These findings emphasize the importance of carefully tuning the dropout rate to strike the right balance between regularization and model capacity, especially when dealing with limited training data.

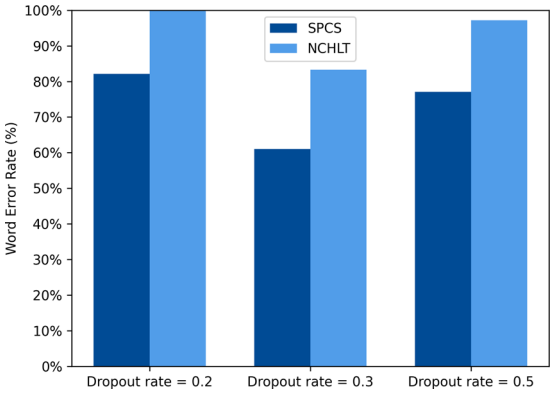

Figure 4.3: WER on testing set for SPCS and NCHLT Sepedi with various dropout rate

During testing, when evaluating the model with unseen data from the SPCS corpus, we observe similar trends only with the 0.3 dropout rate. A dropout rate of 0.3 shows the lowest WER at 61.01%, followed by a dropout rate of 0.5, which produces a WER of 77.09%. As it was the case with the SPCS test set, the lowest WER is attained with a dropout rate of 0.3. In contrast, a dropout rate of 0.1 leads to a much higher WER of 82.11%. This observation is different from what was obtained with the SPCS test set for a dropout rate of 0.1. This dropout rate had the second best WER while in this case it was the highest error rate. These results underscore the crucial role of dropout rates in shaping the model's performance across different datasets. They highlight the necessity of meticulously choosing an appropriate dropout rate during the model architecture design phase to achieve optimal performance. Moreover, the variation in performance between the SPCS and NCHLT corpora indicates that the model finds it challenging to adapt when encountering data from a different context, such as the Sepedi language in the NCHLT corpus. This points to the difficulties of training ASR models on diverse and representative datasets and underscores the need for robust adaptation mechanisms to manage various linguistic contexts effectively.

Applying the dropout rate to the NCHLT pure Sepedi data during testing, the model follows the same trend as the dropout rate changes. The lowest WER is observed with a 0.3 dropout rate for both datasets, at 83.27%. Higher WERs are noted with a 0.5 dropout rate of 97.22%, and the highest WER occurs with a 0.2 dropout rate of 100%. A dropout rate of 0.3 results in the lowest WER across different datasets, suggesting that this rate provides the best balance between preventing overfitting and maintaining model performance. A dropout rate of 0.5 results in a higher WER compared to 0.3. This suggests that setting the dropout rate excessively high could result in underfitting, causing the model to inadequately learn from the training data and consequently perform poorly on unseen data.

When applying the dropout rate to the NCHLT pure Sepedi data during testing, the model achieves the lowest WER of 83.27% with a dropout rate of 0.3. The highest dropout rate is obtained when a dropout rate of 0.5 results in a WER of 97.22% and 0.2 yields the highest WER of 100%. These findings suggest that a 0.3 dropout rate provides the optimal balance between preventing overfitting and maintaining strong model performance. On the other hand, a dropout rate of 0.5 appears to be too high, likely causing underfitting. This means the model fails to learn adequately from the training data, leading to poorer performance on unseen data.

The varying performance between different corpora, such as the SPCS and NCHLT datasets, highlights the model's struggle to adapt to distinct linguistic contexts, such as the Sepedi language. This indicates a need for more sophisticated adaptation mechanisms in ASR systems to handle diverse datasets effectively. The Sepedi NCHLT test corpus represents a different domain or context compared to the training data, making adaptation challenging for the system. ASR systems often perform better when the training and testing data are closely aligned in terms of domain and context, explaining why the SPCS dataset performs well since it was used for training.

4.3. Learning rate effects on Code-Switched speech

Learning rate is a hyperparameter that regulates the speed at which the model's weights are updated during training. The learning rate setting has specific implications for how the model learns during training. In Figure 4.4, the graph shows the

CTCLoss as a function of the learning rate for three models that have been trained at a 1e-3, 1e-4 learning rate as well as a 1e-5 learning rate. The graph displays the validation loss and training loss for all the learning rates. However,the smallest evaluated loss value achieved is 31.35 on epoch 15 when using 1e-3 learning rate. Decreasing the learning rate leads to a corresponding increase in the loss value. This is evident from the fact that the lower learning rates of 1e-4 and 1e-5 lead to higher losses of 41.70 and 50.47, respectively.

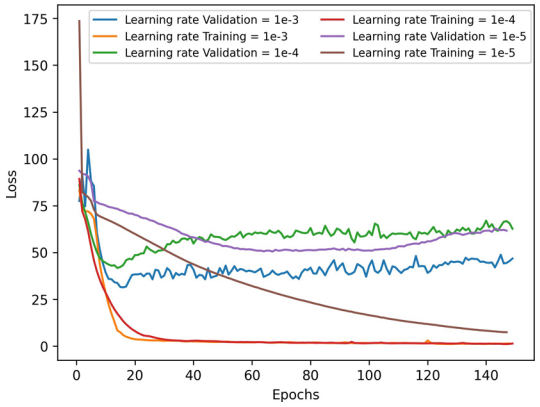

Figure 4.4: SPCS loss values for various learning rates on validation set

The graphs for learning rates 1e-3 and 1e-4 showed signs of overfitting earlier, starting at epoch 20, whereas the overfitting for 1e-5 began later. The earlier onset of overfitting suggests that these higher learning rates caused the models to fit too closely to the training data, leading to reduced generalization on unseen data. Additionally, there is some noise present in the graphs for 1e-3 and 1e-4, whereas there is no noticeable noise in the graph for 1e-5. The lower learning rate of 1e-5 allowed the model to train more slowly and steadily resulting in a smoother loss curve without significant noise.

The model achieved its lowest loss at epoch 15 when using a learning rate of 1e-3, as depicted in Figure 4.5. At this point, the WER was observed to be 39.15%. This highlights the significant impact of the learning rate on model performance. Thus, reducing the learning rate to 1e-4 increased the WER to 60.37%, and further lowering it to 1e-5 resulted in a WER of 87.46%. These outcomes show how the choice of learning rate influences the ability of the model to effectively learn from the training data.

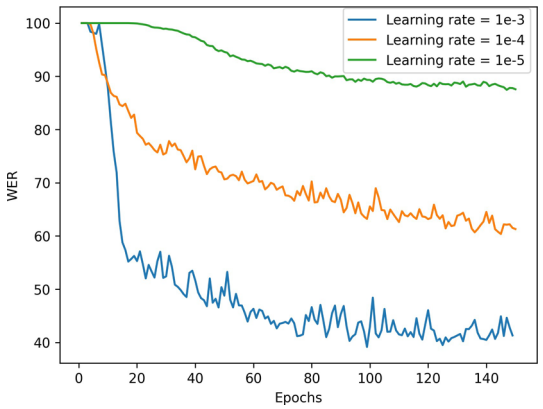

Figure 4.5: SPCS WER for various learning rates on validation set

These findings illustrate the critical role of the learning rate in training neural networks. A high learning rate can lead to instability, whereas a low learning rate can result in slow convergence or becoming trapped in suboptimal solutions. The learning rate impacts how quickly the model converges. The learning rate of 1e-3 resulted in rapid initial improvements, leading to low loss early on at epoch 15, but it took longer to fine-tune the parameters to achieve the lowest WER at epoch 98. This indicates that higher learning rates enable faster initial improvements but may require more time to fine-tune for optimal performance. Lower learning rates provide more stable convergence but slower learning, highlighting the importance of selecting an appropriate learning rate for effective ASR model training.

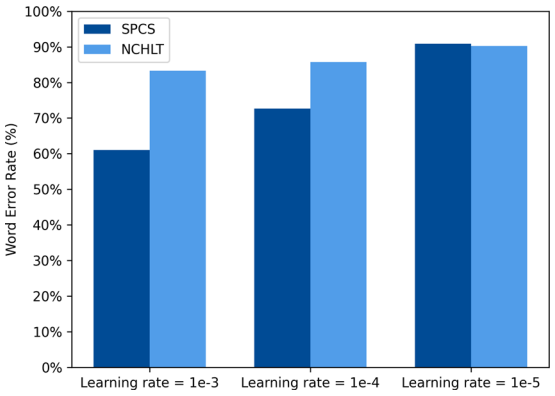

Figure 4.6: WER on testing set for SPCS and NCHLT Sepedi with various learning rates

For the testing sets, when the learning rate is set to 1e-3, the model achieves the

lowest WER of 61.01% in SPCS which is the general testing and 83.27% when tested specifically with the NCHLT corpus on epoch 98. This indicates that a learning rate of 1e-3 allowed the model to strike a balance between rapid learning and avoiding overshooting optimal parameters. Contrastingly, with 1e-4 learning rate, the WER increased to 72.67% in general testing and 85.70% with the NCHLT corpus, indicating slower convergence and potentially settling in suboptimal states. At 1e-5, the model exhibited the poorest performance, with WERs of 90.90% and 90.25% in general testing and with the NCHLT corpus, respectively. This indicates that the learning rate is too low, causing the model to converge very gradually or become stuck in suboptimal local minima, resulting in poor performance.

The results emphasize the critical role of learning rate in determining the effectiveness and efficiency of ASR model training. Selecting the right learning rate is essential for achieving the best possible performance. Learning Rate of 1e-3 is the most effective learning rate, providing the best balance between learning speed and model accuracy. The model achieves the lowest WER with this setting, indicating that it efficiently reaches optimal performance. A learning rate that is too low also leads to inefficient training, requiring more time and resources to potentially achieve suboptimal performance.

4.4. Code-Switched model complexity using kernel size

The kernel sizes of the convolutional layers play a crucial role in feature extraction from the input audio spectrograms. Smaller kernels focus on more localized temporal patterns. They capture very fine-grained temporal details, which are crucial for recognizing short-term dependencies and rapid changes in the audio signal. Larger kernels in the frequency dimension capture a wider range of spectral features, which is crucial for identifying broader frequency-based patterns. Utilizing a combination of kernel sizes allows for a hierarchical and comprehensive feature extraction, balancing between capturing global and local patterns effectively.

Local patterns refer to the short-term dependencies and fine-grained details in the speech signal. These patterns are typically associated with rapid changes and short sequences, such as phonemes, syllables, or short transitions between sounds.

Global patterns refer to the long-term dependencies and broader contextual features in the speech signal. These patterns encompass longer sequences and broader contexts, such as words, phrases, intonation patterns, and overall sentence structure. Various kernel sizes are tested in the convolutional layers to assess their impact on recognition accuracy: [5,5][4,4] (Kernel 1), [7,7][5,5] (Kernel 2), [9,9][7,7] (Kernel 3), and [11,11][9,9] (Kernel 4).

Figure 4.7 below shows the loss throughout 150 epochs for both the training and validation sets. The overall trend shows a decreasing loss with increasing steps, despite occasional fluctuations indicated by spikes. The lowest validation loss of 29.25 is achieved with kernel 2. Using kernel 3 filters results in a validation loss of 31.58. The loss increases slightly to 32.30 with kernel 1 and further to 33.51 with kernel 4 filters. This highlights the influence of various kernel sizes on the performance of the model, balancing between capturing fine-grained temporal details and broader spectral features.

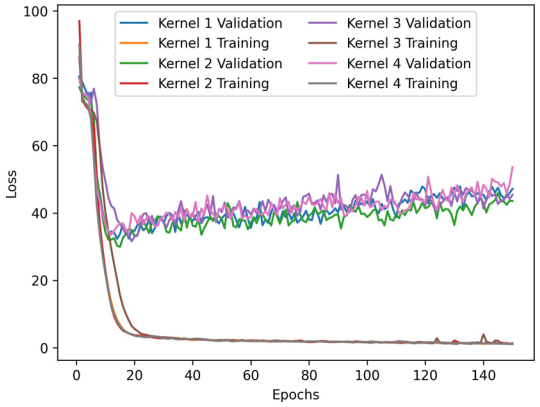

Figure 4.7: SPCS loss values for various kernel sizes on validation set

Figure 4.8 displays the WER for varying kernel sizes. The lowest WER, 37.92%, is observed with kernel 2 configuration. This suggests that this combination effectively captures both fine-grained temporal details and broader spectral features, leading to better performance in recognizing audio patterns. When larger kernels are used, such as kernel 3 and kernel 4, the WER increases to 39.21%. This indicates that while these larger kernels capture broader frequency-based patterns, they may not be as effective in capturing the fine-grained details necessary for optimal performance. Additionally, a slight rise in WER to 39.45% is noted with different kernel configurations,

suggesting a delicate balance between kernel size and the model's ability to generalize well over different audio features. This highlights the importance of selecting appropriate kernel sizes to balance the extraction of both local and global patterns in the audio spectrogram.

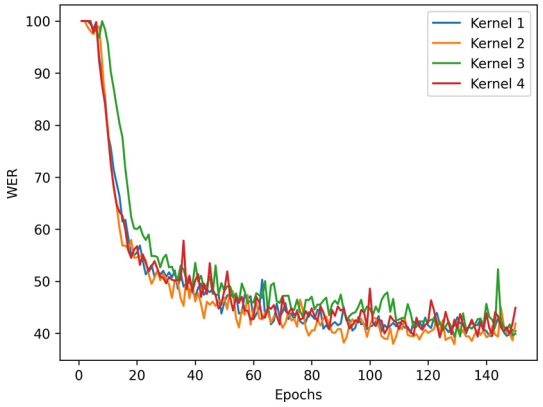

Figure 4.8: SPCS WER for various kernel sizes on validation set

Figure 4.9 depicts the results of using different kernel sizes in an ASR system to illustrate the critical impact kernel dimensions have on model performance, particularly in terms of WER. When the kernel size is set to kernel 3, the model achieves the lowest WER, recording 59.07% in SPCS and 86.36% with the NCHLT corpus. This optimal performance suggests that this specific kernel size enables the model to effectively capture and learn the relevant features from the training data, balancing between generalization and overfitting. In contrast, reducing the kernel size to kernel 1 results in a slight increase in WER to 59.53% in general testing and 82.31% with the NCHLT corpus. This indicates a marginal decrease in the ability of the model to capture sufficient contextual information. Similarly, increasing the kernel size to kernel 4 also raises the WER to 60.72% in general testing and 81.73% with the NCHLT corpus, suggesting that the model may begin to capture too much noise along with the relevant features. The worst performance is observed with kernel 2 where the WER reaches 61.58% in general testing and 84.64% with the NCHLT corpus, likely due to insufficient or excessive feature extraction, which hampers the model's ability to generalize effectively from the training data. These results underscore the importance of selecting an appropriate kernel size to balance detail and context in feature extraction, crucial for optimizing ASR performance.

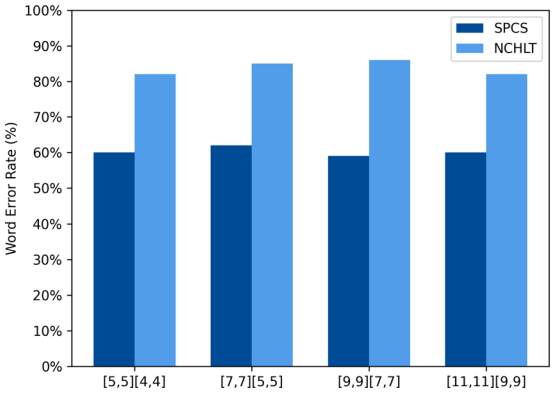

Figure 4.9: WER on testing set for SPCS and NCHLT Sepedi with various kernel sizes

4.5. Effects of Filter Quantity on Complexity in Code-Switched Data

Filters within convolutional layers are designed to extract temporal and frequency-based features from the spectrogram representations of input audio data. These filters are pivotal in extracting pertinent acoustic features utilized in subsequent stages for speech recognition and decoding. The quantity of filters impacts the model's complexity by introducing additional parameters to be learned during training. Consequently, they exert a substantial influence on both the complexity and performance of the model.

Figure 4.10 displays the loss over 100 epochs for both training and validation sets. The use of 16 filters yields the lowest validation loss, specifically at 21.96. This suggests that this configuration best enhances the model's generalizability to unseen data. The loss increases to 22.24 when using 32 filters, and rises further to 22.82 with 64 filters. The graph shows a plateau for 16 and 32 filters from epoch 20.

The results indicate that, within the specified model architecture and dataset, using 16 filters strikes a balance between learning from the training data and generalizing to data not previously encountered. Both 16 and 32 filters show higher validation losses, indicating diminishing returns in performance. However, spikes in the validation loss suggest that the model struggles to generalize to new, unseen data at certain epochs, particularly evident with 64 filters. Despite this, the plot displaying smooth training loss indicates that the model fits the training data well. Moreover, it's important to note that

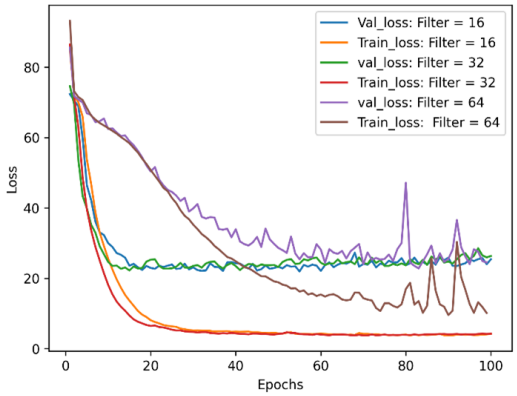

Figure 4.10: SPCS loss values for various number of filters on validation set

for both 16 and 32 filters, the validation error surpasses the training error after 20 epochs, indicating signs of overfitting. Conversely, the graphs for both training and validation with 64 filters show good fitting initially, but noise starts appearing around epoch 80, indicated by spikes in the graphs.

Figure 4.11 illustrates the WER across different numbers of filters. The model achieves its lowest WER of 41.9% with 16 filters. Increasing the number of filters to 32 results in a slight rise in WER to 43.48%. However, employing 64 filters significantly raises the WER to 47.47%, indicating diminishing performance as complexity increases in the model architecture.

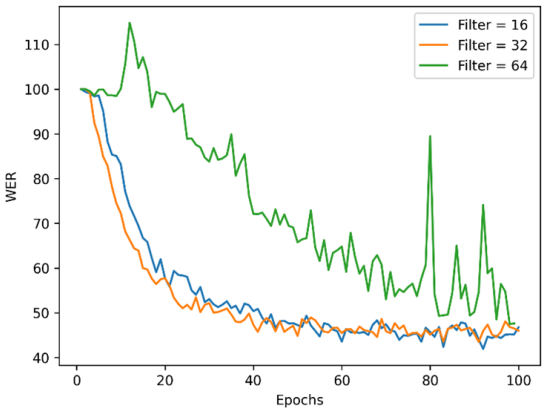

Figure 4.11: SPCS WER for various number of filters on validation set

The model performed well when using 16 filters, indicating that for this dataset, fewer filters in the convolutional layers were advantageous. This configuration effectively captured relevant features and patterns in the data. However, as the number of filters increased to 32, a slight increase in WER was observed. This suggests that

with increased complexity, the model began capturing more intricate patterns that did not generalize well to the test set. Furthermore, employing 64 filters resulted in a notably higher WER with increased noise. This indicates that the model became overly complex, potentially overfitting to the training data and failing to generalize effectively to unseen data. These challenges stem from the limited training data, which makes it difficult to generalize to new, unseen data.

When assessing the SPCS dataset, utilized for training, validation, and testing in this study, the WER displays varying outcomes when the model is tested on unseen data from the same corpus. Using 16 filters, the WER is recorded at 50.05%, while employing 32 filters results in a slightly lower WER of 50.24%. However, utilizing 64 filters leads to a higher WER of 53.51%. In contrast, testing on the NCHLT corpus shows that with 16 filters, the WER is at its lowest at 84.59%, followed by 85.06% with 32 filters, and 89.89% with 64 filters. These results show the critical role of selecting an optimal number of filters in the model architecture, as this choice has a significant impact on the model's performance across different corpora. The system's lack of exposure to a diverse dataset for the Sepedi NCHLT test corpus contributes to less favorable results. Thus, the NCHLT text corpus presents a unique context compared to the training data, leading to observable performance discrepancies. These findings highlight the challenges the model faces in adaptation.

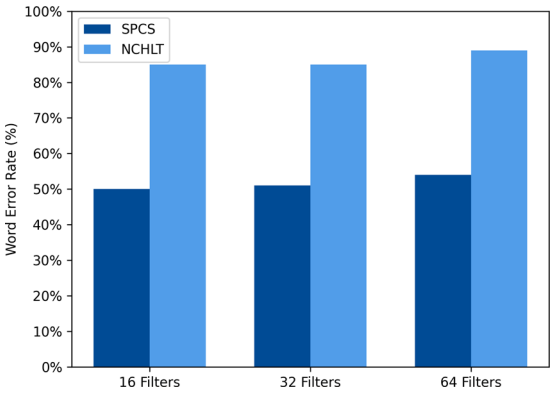

Figure 4.12: WER on testing set for SPCS and NCHLT Sepedi with various number of filters

The WERs for the NCHLT corpus are consistently higher than those for the SPCS dataset, regardless of the number of filters used. This indicates that the model strug-

gles with the NCHLT data, underscoring the necessity of training on diverse and representative datasets.

The distinct linguistic characteristics of the NCHLT corpus, especially the Sepedi language data, pose additional challenges for the model. This further highlights the need for careful selection of the number of filters and the implementation of robust adaptation mechanisms to improve the model's performance in varied linguistic contexts. Since code-switching is the practice of alternating between two or more languages within a conversation, it introduces complexity to ASR systems. The system encountered difficulties in accurately recognizing the spoken language due to the more intricate code-switching patterns present in the Sepedi NCHLT test corpus compared to the training data.

4.6. Effects of the depth using RNN layers on Code-Switched

RNNs are specifically designed to handle sequences by maintaining a memory of previous inputs through hidden states, enabling them to capture temporal dependencies. Figure 4.13 depicts training and validation loss over 150 epochs. The graph of the loss exhibits a downward trend as the epochs increases, although it occasionally displays fluctuations with noticeable spikes. The lowest loss occurs when employing 3 RNN layers, with a value of 31.35%. The loss increased when 2 layers of RNN layers were used to 34.67%. It further increased to 47.67% when 4 RNN layers were employed and 54.56% when 5 RNN layers were used. The results show that using three layers is effective because it resulted in the lowest loss.

The cause of this behavior is promoted by the complexity of the model. A model with too few layers does not capture the underlying patterns in the data well, resulting in higher loss. While a model with too many layers overfit to the training data, performing badly on new, unseen data. For models with 2 and 3 RNN layers, the graphs start to show signs of overfitting from epoch 10 onwards, despite achieving the lowest loss shortly after epoch 10. The lowest loss after epoch 10 indicates that the model fits the training data very well. For models with 4 and 5 filters, the graphs show significant noise, indicated by spikes, and they display overfitting after 20 epochs.

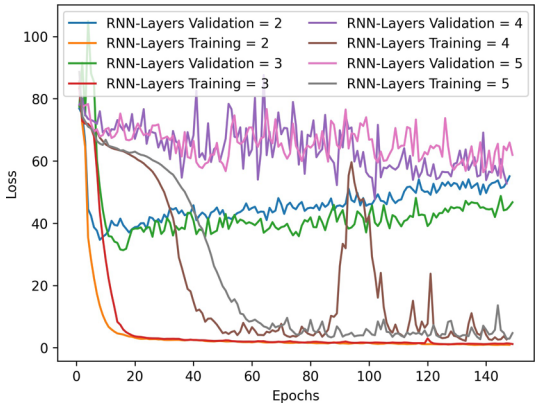

Figure 4.13: SPCS loss values for various RNN layers on validation set

Figure 4.14 displays the WER for varying numbers of RNN layers. The lowest WER is observed when using 4 RNN layers, with a value of 38.15%. This setup accurately captured pertinent trends and features in the data. However, when the number of RNN layers decreases to 3, the WER increases slightly to 38.48%. The graph for 3 layers consistently stayed lower than others until epoch 131, after which the graph for 4 layers dropped below it at epoch 132. The WER increased to 44.7% with 5 RNN layers. The highest WER of 48.58% was observed where 2 RNN layers were used.

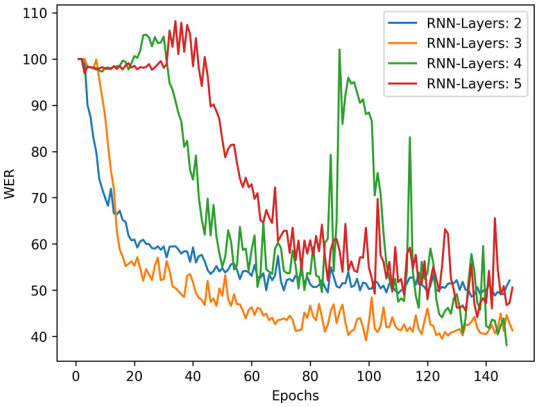

Figure 4.14: SPCS WER for various RNN layers on validation set

There is an optimal point in the number of RNN layers, and deviating from this point in either direction, too few or too many layers negatively impacts the ability of the model to accurately transcribe words in the given data. The relationship between WER and the number of RNN layers reflects the trade-off between model capacity and generalization. Too few layers and too many layers lead to overfitting, and there's

an optimal point where the model achieves the right balance for the given task and dataset.

The test results from the two corpora used are shown below by Figure 4.15. It shows the WER of both the datasets in the model with varying RNN layers of 2,3,4 and 5. The model does not perform well with unseen data. We observe lesser WER when the model is tested using the SPCS data which was also used for training. It is a different case when the NCHLT data is used. The performance varies across RNN layers for both corpora.

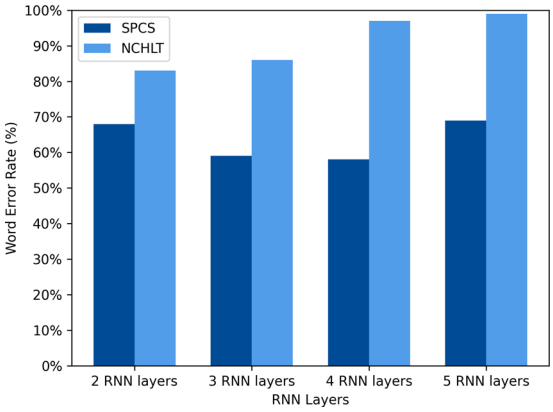

Figure 4.15: WER on testing set for SPCS and NCHLT Sepedi with various number of RNN Layers

The results show that in all the considered scenarios the model exhibits strong performance on the SPCS data, which was also part of the training set. Specifically, the WER on the SPCS dataset shows low WER compared to the WER recorded on the NCHLT corpus. It's noteworthy that the model had not been exposed to the NCHLT data during training and was solely utilized for testing purposes.

The use of two RNN layers significantly enhances the model's performance on the SPCS dataset, reducing the WER to 68%, a considerable improvement over the 83% WER observed on the NCHLT corpus. With three RNN layers, the WER on the SPCS dataset further decreases to 59%, compared to 86% on the NCHLT corpus. The model achieves its lowest WER of 58% on the SPCS testing data with four RNN layers, although it still records a WER of 97% on the NCHLT corpus. Despite these improvements, the model's WER remains at 69% on the SPCS dataset and peaks at

99% on the NCHLT corpus, indicating difficulties in generalizing unseen data from the NCHLT corpus. When the model size was too small (2 rnn layers) and not comlicated, SPCS records a very high WER 69% which is the same as when the network is too big and complicated (5 RNN layers). The lowest WER is recorded when the model it is neither too complicated or too simple (4 RNN layers). It recorded a low WER of 51%.

The model performs well on the SPCS dataset used for training, with lower WER and improved performance when utilizing 4 RNN layers. However, it faces challenges with generalization to the NCHLT dataset, recording higher WER and indicating potential overfitting. The high WER on the NCHLT data is influenced by the fact that the system is not trained on a different and representative dataset for the Sepedi NCHLT test corpus, posing a challenge for the model to perform well on NCHLT data.

4.7. The Analysis of the model's generilizability

The results produced by the model in the testing phase for both the SPCS and NCHLT datasets, are depicted below in Table 4.1. There was higher accuracy in recognizing words in the SPCS corpus than in the NCHLT corpus.

In Table 4.1 for the SPCS the ASR model demonstrates strong performance with high accuracy in recognizing and transcribing the provided sentences. It handles proper nouns and simple phrases well, as seen with *"brithani"* and *"go tla interview"* However, it exhibits minor errors with verb forms and occasional spelling mistakes, such as *"re"* becoming *"e"* and *"thousand"* becoming *"thoursand"*. Overall, the model shows promise with slight areas for improvement in verb form recognition and spelling accuracy.

For NCHLT, the results shown in table 3.4 the model demonstrates variable performance in recognizing and transcribing the provided sentences. For instance, it shows significant struggles with simple sentences, as seen in "a go ya go a" where it fails to maintain proper word separation and formation. Conversely, it performs well on sentences composed of multiple connectors, as evidenced by the perfect match in "re na le bana ba" However, the model has considerable difficulty with longer, common

Table 4.1: Comparison of target and predicted transcriptions across the SPCS and NCHLT test corpus

Corpus	Reference and Predicted text
SPCS test corpus	**Target** : go nale **dignostic indicators**
	Prediction: go nale **dignostic indicators**
	Target : go tla **interview**
	Prediction: go tla **interview**
	Target : **brithani**
	Prediction: **brithani**
	Target : **o ithopela two thousand rand**
	Prediction: **o ithopela two thoursand rand**
	Target : **bohlaba bja toropo ya pretoria**
	Prediction: **bo hlaaba thopo ypertorin**
NCHLT test corpus	**Target** : la ka ga le mpee
	Prediction: la ka ralempel
	Target : a go ya go a
	Prediction: agwoo ya goo sa
	Target : le tona e sa le
	Prediction: le thona e sale
	Target : re na le bana ba
	Prediction: re na le bana ba
	Target : ka kgopolo ya gore mogongwe
	Prediction: ka kgotsle ya go rangogogwa

words and phrases, as seen in "ka kgopolo ya gore mogongwe" indicating a need for improvement in handling phonetically complex and less common words. While it makes minor errors in some cases, such as "le tona e sa le" where "tona" is predicted as "thona" it also exhibits major inaccuracies in others, like "la ka ga le mpee" being transcribed as "ralempel" showing inconsistency in its ability to recognize and transcribe accurately. These results suggest that while the model can handle certain structures well, it requires enhancement to better manage word boundaries, phonetic complexity, and less frequent vocabulary.

The model demonstrates skill in predicting the words at the beginning of the sentences accurately, suggesting a strong ability to recognize sentence beginnings. The model shows notable precision in predicting many Sepedi terms, indicating effective training on Sepedi language data. The model performs better on longer sentences (five words), suggesting that it might be leveraging context to improve accuracy. The model accurately predicts shorter words and connectors, which are essential for maintaining coherence and logical structure in sentences.

The model demonstrates adeptness in predicting the initial words of a provided text accurately and exhibits a notable precision in forecasting the majority of Sepedi terms within the sentence. This suggests the model has been trained effectively on Sepedi language data, enabling it to generate pertinent predictions within this linguistic framework. Particularly noteworthy is its effectiveness in predicting longer sentences (comprising five words), which exceeds its precision in predicting shorter ones. The model adeptly predicts shorter words, commonly referred to as connectors, which is essential for maintaining the coherence and cohesion of sentences and texts. Connectors serve to bridge gaps and up-hold a logical and understandable structure in written content . The model's proficiency in predicting these connectors significantly contributes to the overall readability and logical structure of the generated text.

The model exhibits strong performance on the SPCS corpus with minor errors, but struggles with the NCHLT corpus, particularly with complex and less common words. This is due to the reason that the only pure Sepedi sentences in the data are 139 out of 12 000 sentences as discussed in chapter 3 table 3.5. This highlights the need for more diverse and representative training data, especially for the Sepedi language in the NCHLT corpus. The ability to predict initial words and connectors accurately is a positive aspect of the model, contributing to the overall readability and logical structure of the generated text. However, the significant errors observed in the NCHLT corpus indicate a requirement for improved adaptation mechanisms to handle diverse linguistic contexts effectively. Insufficient training data can result in challenges when generalizing to new, unseen data.

4.8. Chapter Summary

The ASR system's performance on two distinct corpora using WER and CTCLoss, examining various parameters such as regularization, learning rate, kernel size, filters, and RNN layers. The model demonstrates strong performance on the SPCS data, utilized for both training and testing. However, it encounters challenges with the NCHLT data, used exclusively for testing, primarily due to differing data patterns between the two corpora.

The study further explores the generalizability of the ASR model by comparing its performance on these two corpora, highlighting the challenges in adapting to diverse linguistic contexts.The ASR model exhibits higher accuracy with the SPCS corpus, effectively handling proper nouns and simple phrases but making minor errors in verb forms and spelling. Conversely, it struggles more with the NCHLT corpus, facing significant challenges with word separation, phonetic complexity, and less common words given the amount of pure Sepedi sentences in the training dataset. Despite these difficulties, the model shows strength in predicting initial words and connectors, particularly in longer sentences, indicating effective training on Sepedi language data.

5. CONCLUSION

This chapter contains the summary of the findings, contributions and final conclusion. There is a lack of studies for under-resourced languages due to insufficient data. These limitations make it difficult to achieve the same level of accuracy in ASR systems as seen with well-resourced languages like English. An ASR system was developed using different testing datasets that are from different corpora one which was a code switched Sepedi-English corpus and the other one is pure Sepedi data which was only used for testing purposes. The model does not perform that well when a different corpus is used than the one used to train the model. The lack of training on a diverse and representative dataset adversely affects the system's performance when tested with the Sepedi NCHLT test corpus. Consequently, the outcomes are less favorable. The observed differences in performance can be attributed to the unique context presented by the NCHLT text corpus in comparison to the training data. These results highlight the challenge the model faces in adapting to contexts.

5.1. Summary of results

The purpose of this study was to analyze the SPCS corpus, which primarily consists of short sentences ranging from one to five words. Sentences with five words are the most common, while one-word sentences are the least frequent. The corpus features a mix of Sepedi, English, and borrowed words from other languages. Notably, the most frequent words are two-character connectors, predominantly in Sepedi. The utterances vary in length from one to 33 characters, with most sentences averaging 27 characters. This analysis provides insights into the prevalent sentence lengths, linguistic composition, and character length of utterances within the SPCS corpus.

The automatic speech recognition system for code-switched language using the CTC approach was trained with the SPCS data. The implemented code-switched ASR system was evaluated on two datasets: the SPCS and the NCHLT. Using WER as the evaluation metric, we found that the model struggles to generalize to new,

untrained data due to limited training data, which restricts its ability to handle unseen inputs. The Sepedi NCHLT test corpus posed a significant challenge because of its distinct context compared to the training data and the limited amount of pure Sepedi utterances available in the training data. Optimal performance in ASR is typically achieved when the training and testing data closely align in terms of domain and context. Code-switching, involving the alternation of multiple languages within a conversation, adds an additional layer of difficulty for ASR systems. The system will struggle to accurately recognize the Sepedi NCHLT test corpus because it has different patterns to the SPCS that is used during training data.

5.2. Limitations and Challenges

Despite significant advancements in ASR technology, the majority of research and development efforts have focused on high resourced languages. This study specifically addresses the Sepedi language, a low-resource language, which poses a challenge due to the limited availability of annotated speech data. This limitation can impact the robustness and accuracy of the developed ASR system.

Code-switching involves the use of multiple languages within a single conversation, making it challenging to accurately transcribe speech due to the variability and unpredictability of language switching. This adds another layer of difficulty in training and evaluating the ASR system.

5.3. Contribution of the study

The results from this study can serve as a foundational reference point for forthcoming endeavors focused on developing code-switched ASR systems for under-resourced languages. A shortage of accessible data challenges these languages, hindering the training and development of various language technology systems that rely on it. CTC-based model performs better even with limited training data when exposed to new, unseen data. This finding is particularly evident in the context of the Sepedi NCHLT test corpus, which posed a significant challenge due to its unique context and the limited amount of pure Sepedi utterances available in the training data.

The study highlights the significance of maintaining minimal model complexity to effectively tackle challenges encountered in under-resourced languages. These findings suggest potential applications across diverse linguistic contexts, where the utilization of deep learning methods could prove advantageous despite limitations in data availability. It has been demonstrated that the model performs well under conditions of minimal complexity, such as employing a 0.3 dropout rate, a learning rate of 1e-3, [7,7][5,5] (Kernel 2) kernel sizes, 16 filters, and 4 RNN layers. This model's success indicates its potential applicability to other under-resourced languages, allowing them to harness deep learning techniques even with limited datasets and minimal model complexity.

5.4. Future Work and Recommendation

Further research is essential for conducting comprehensive experiments and data analysis on the Sepedi language. Additionally, exploring other under-resourced languages is necessary to assess the model's generalizability. These languages face significant challenges due to the limited availability of data, which is essential for the development and improvement of various language technology systems. Consequently, future work should prioritize the collection and analysis of more extensive datasets, as well as the development of methodologies tailored to the unique linguistic characteristics of these languages. This will help in improving the performance and reliability of ASR systems in multilingual and code-switching contexts, thereby contributing to the broader field of language technology.

BIBLIOGRAPHY

[1] J. Lee and S. Watanabe, “Intermediate loss regularization for ctc-based speech recognition,” in IEEE International Conference on Acoustics, Speech and Signal Processing (IC 2021, pp. 6224–6228.

[2] D. Eledath, V. Pavithra, N.R. Thurlapati, T. Banerjee, and V. Ramasubramanian, “Few-shot learning for cross-lingual end-to-end speech recognition,” in Workshop on Machine Learning in Speech and Language Processing 2021 (MLSLP 2021), Sat 2021.

[3] M. Muller, S. Stüker, and A. Waibel, “Language adaptive multilingual CTC speech recognition,” in Speech and Computer: 19th International Conference, SPECOM, 2017, pp. 473–482.

[4] A. Magueresse, V. Carles and E. Heetderks, “Low-resource languages: A review of past work and future challenges,” in arXiv preprint arXiv:2006.07264, 2020.

[5] Y. Miao, M. Gowayyed, and F. Metze, F, “EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding,” in 2015 IEEE workshop on automatic 2015, pp. 167–174.

[6] J. Makhoul and R. Schwartz, “State of the art in continuous speech recognition,” in Proceedings of the National Academy of Sciences, 1995, pp. 9956–9963.

[7] T. Moeng, S. Reay, A. Daniels, and J. Buys, “Canonical and surface morphological segmentation for Nguni languages,” in Southern African Conference for Artificial Intelligence 2021, pp. 125–139.

[8] A. Biswas, E. Yılmaz, F. De Wet, E. van der Westhuizen and T. Niesler, “Semi-supervised Development of ASR Systems for Multilingual Code-switched Speech in Under-resourced Languages,” in arXiv:2003.03135v1 [eess.AS], 2020.

[9] Z. Xiao, Z. Ou, W. Chu, and H. Lin, “EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding,” in 2018 11th International Symposium on Ch 2018, pp. 146–150.

[10] H. Su, G. Li, D. Yu, and F. Seide, “Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription,” in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6664–6668.

[11] S. Zhang, M. Lei, Y. Liu, and W. Li, “Investigation of modeling units for mandarin speech recognition using dfsmn-ctc-smbr,” in IEEE International Conference on Acoustics, Spee 2019, pp. 7085–7089.

[12] A. Biswas, E. van der Westhuizen, T. Niesler and F. De Wet, “Improving ASR for Code-Switched Speech in Under-Resourced Languages Using Out-of-Domain Data,” in Spoken Language Technologies for Under-Resourced Languages (SLTU), 2018, pp. 122–126.

[13] W.Q. Yow, J.S. Tan, and S. Flynn, “Towards end-to-end code-switching speech recognition,” Bilingualism: Language and Cognition, vol. 21(5), pp. 1075–90, 2018.

[14] A. Briggs, and M. Sculpher, “An introduction to Markov modelling for economic evaluation,” Pharmacoeconomics, vol. 13(4), pp. 397–409, 1998.

[15] Z. Li, C.Z. Lu, J. Qin, C.L. Guo and M.M Cheng, “Towards an end-to-end frame-work for flow-guided video inpainting,” in Proceedings of the IEEE/CVF conference on computer 2022, pp. 17 562–17 571.

[16] H. Adel, N.T. Vu, and T. Schultz, “Combination of recurrent neural networks and factored language models for code-switching language modeling,” Proceedings of the 51st Annu vol. 2, pp. 206–211, 2013.

[17] K. Li, J. Li,G. Ye, R. Zhao and Y. Gong, “Towards code-switching ASR for end-to-end CTC models,” in IEEE International Conference on Acoustics, Speech and Signal Processin 2019, pp. 6076–6080.

[18] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model,” Interspeech, vol. 2, No. 3, pp. 1045–1048, 2010.

[19] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Extensions of recurrent neural network language model,” in 2011 IEEE international conference on acoustics 2011, pp. 5528–5531.

[20] C. Nilep, “Code Switching in Sociocultural Linguistics,” Colorado Research in Linguistics, vol. 19, pp. 1–22, 2006.

[21] Y. Li and P. Fung, “Code-switch language model with inversion constraints for mixed language speech recognition,” in Proceedings of COLING 2012, 2012, pp. 1671–1680.

[22] N. Dongen, “Analysis and prediction of Dutch-English code-switching in Dutch social media messages,” Master’s thesis, Universiteit van Amsterdam, Amsterdam, Netherlands, 2017.

[23] H. Adel,N.T. Vu, F. Kraus,T. Schlippe, H. Li and T. Schultz, “Recurrent neural network language modeling for code switching conversational speech,” in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8411–8415.

[24] G. Lee, X. Yue,and H. Li, “Linguistically Motivated Parallel Data Augmentation for Code-Switch Language Modeling,” in Interspeech, 2019, pp. 3730–3734.

[25] K. Ihemere, “In support of the Matrix Language Frame Model: evidence from Igbo-English intrasentential code-switching,” Language Matters, vol. 47(1), pp. 105–127, 2016.

[26] G.I. Winata, A. Madotto, C.S. Wu, and P. Fung, “Learn to code-switch: Data augmentation using copy mechanism on language modeling,” in arXiv preprint arXiv:1810.10254, 2018.

[27] T. I. Modipa and M. H. Davel, “Two sepedi-english code-switched speech corpora,” Language Resources and Evaluation, vol. 56, pp. 703–727, 2022.

[28] S. Dalmia, Y. Liu, S. Ronanki and K. Kirchhoff, “Transformer-transducers for code-switched speech recognition,” in ICASSP 2021-2021 IEEE International Conference on Ac 2011, pp. 5859–5863.

[29] S. Shah, B. Abraham, S. Sitaram and V. Joshi, “Learning to recognize code-switched speech without forgetting monolingual speech recognition,” in arXiv preprint arXiv:2006. 2020.

[30] I. Hamed, P. Denisov, C.Y. Li, M. Elmahdy, S. Abdennadher and N.T. Vu, “Investigations on speech recognition systems for low-resource dialectal Arabic–English code-switching speech,” in Computer Speech and Language, 72, 2022, pp. 101 278–101 278.

[31] X. Yue, G. Lee, E. Yılmaz, F. Deng and H. Li, “End-to-end code-switching ASR for low-resourced language pairs,” in 2019 IEEE automatic speech recognition and understandin 2019, pp. 972–979.

[32] N. Nguyen and D. Nguyen, “Hidden markov model for stock selection,” Risks, vol. 3(4), pp. 455–473, 2015.

[33] L.R. Rabine, “A tutorial on hidden Markov models and selected applications in speech recognition,” Proceedings of the IEEE, vol. 77(2), pp. 257–286, 1989.

[34] G. Xuan, W. Zhang and P. Chai, “EM algorithms of Gaussian mixture model and hidden Markov model,” Proceedings 2001 international conference on image processing (Cat. N vol. 1, pp. 143–146, 2001.

[35] H.J. Nock and S.J. Young, “Loosely coupled HMMs for ASR,” in INTERSPEECH, 2000, pp. 143–146.

[36] J. Liu, D. Cai, and X. He, “ Gaussian mixture model with local consistency,” Proceedings of the AAAI conference on artificial intelligence, vol. 24, pp. 512–517, 2010.

[37] A. Sholokhov, M. Sahidullah and T. Kinnunen, “ Semi-supervised speech activity detection with an application to automatic speaker verification,” Computer Speech and Language vol. 47, pp. 132–156, 2018.

[38] R. Togneri and D. Pullella, “An overview of speaker identification: Accuracy and robustness issues,” IEEE circuits and systems magazine, vol. 11(2), pp. 23–61, 2011.

[39] D.A. Reynolds and R.C. Rose, “Robust text-independent speaker identification using Gaussian mixture speaker models,” IEEE transactions on speech and audio processing, vol. 3(1), pp. 72–83, 1995.

[40] R. C. Rose and D. A Reynolds, “Text independent speaker identification using automatic acoustic segmentation,” in In International Conference on Acoustics, Speech, and Sig 1990, pp. 293–296.

[41] B. L. Tseng, F. K. Soong and A. E. Rosenberg, “Continuous probabilistic acoustic map for speaker identification,” Proceedings ICASSP-92: 1992 IEEE International Conference vol. 2, pp. 161–164, 1992.

[42] R.O. Duda and P.E Hart, “Pattern classification and scene analysis,” New York: Wiley, vol. 3, pp. 731–739, 1973.

[43] M.O.E. Elfahal, “Automatic recognition and identification for mixed sudanese arabic–english languages speech,” in arXiv preprint arXiv:2006.00782, 2019.

[44] M. Müller, S. Stüker, and A. Waibel, “Language Adaptive Multilingual CTC Speech Recognition,” in Speech and Computer: 19th International Conference, SPECOM 2017, Hatfield 2017, pp. 473–482.

[45] J. Li, R. Zhao, J.T Huang, and Y. Gong, “Learning small-size DNN with output-distribution-based criteria,” in Fifteenth annual conference of the international speech communic 2014.

[46] A.R. Mohamed, “Deep Neural Network Acoustic Models for ASR,” Doctoral dissertation, Universit 2014.

[47] L. Zheng, “The deep neural network and content transcription-based speech recognition algorithm in keyword detection,” Journal of Physics: Conference Series, vol. 1544 No.1, pp. 12 150–12 150, 2020.

[48] I. Goodfellow, Y. Bengio and A. Courville, “Deep learning,” in MIT press, 2016.

[49] C. Fan, J. Wang, W. Gang and S. Li, “Assessment of deep recurrent neural network-based strategies for short-term building energy predictions,” in Applied energy, 236, 2019, pp. 700–710.

[50] Z. Cui, R. Ke, Z. Pu and Y. Wang, “Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values,” in Transportation Research Part C: Emerging Technologies 118, 2020, pp. 102 674–102 674.

[51] M. Abazari Kia, “Question-driven text summarization with extractive-abstractive frameworks,” in Doctoral dissertation, University of Essex, 2023.

[52] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C.L.Y. Bengio, and A. Courville, “Towards end-to-end speech recognition with deep convolutional neural networks,” arXiv preprint arXiv:1701.02720., 2017.

[53] A.N. Mon, W. Pa Pa and Y.K. Thu, “Improving Myanmar automatic speech recognition with optimization of convolutional neural network parameters,” International Journal o vol. 7, 2018.

[54] D. Palaz and R. Collobert, “Analysis of CNN-based speech recognition system using raw speech as input,”, 2015.

[55] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in 2016 IEEE international confere 2016, pp. 4960–4964.

[56] Z. Chen, M. Jain, Y. Wang, M.L.Seltzer and C. Fuegen, “Joint Grapheme and Phoneme Embeddings for Contextual End-to-End ASR,” in Interspeech, 2019, pp. 3490–3494.

[57] L. Deng, G. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: An overview,” in 2013 IEEE international conf 2013, pp. 8599–8603.

[58] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, ... and D. Zhao, “A streaming on-device end-to-end model surpassing server-side conventional model quality and latency,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, 2020, pp. 6059–6063.

[59] R. Joshi, and V. Kannan, “Attention based end to end speech recognition for voice search in hindi and english,” in Proceedings of the 13th Annual Meeting of the Forum for In 2021, pp. 107–113.

[60] N. Luo, D. Jiang, S. Zhao, C. Gong, W. Zou, and X. Li, “Towards end-to-end code-switching speech recognition,” in arXiv preprint arXiv:1810.13091, 2018.

[61] A. Tjandra, S. Sakti, and S. Nakamura, “Multi-scale alignment and contextual history for attention mechanism in sequence-to-sequence model,” in 2018 IEEE Spoken Languag 2018, pp. 648–655.

[62] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos, “Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference,” in 020 53rd Annual IEEE/ACM In 2020, pp. 811–824.

[63] S. Naeem, M. Iqbal, M. Saqib, M. Saad, M.S. Raza, Z. Ali, ... and M. U Arshad, “Subspace gaussian mixture model for continuous urdu speech recognition using kaldi,” in 2020 14th International Conference on Open Source Systems and Technologies (IC 2020, pp. 1–7.

[64] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proceedings of the 23rd international conference on Machine learning, 2006, pp. 369–376.

[65] Y. Miao, M. Gowayyed, X. Na, T. Ko, F. Metze, and A. Waibel, “An empirical exploration of CTC acoustic models,” in 2016 IEEE international conference on acoustics, speech a 2016, pp. 2623–2627.

[66] A. Graves, and A. Graves, “Connectionist temporal classification,” in Supervised sequence labelli 2012, pp. 61–93.

[67] D. Wang, X. Wang and S. Lv, “An overview of end-to-end automatic speech recognition,” Symmetry 11, vol. 8, pp. 1018–1018, 2019.

[68] Y. Sharma, B. Abraham, K. Taneja and P. Jyothi, “Improving low resource code-switched ASR using augmented code-switched TTS,” in arXiv preprint arXiv:2010.05549, 2020.

[69] Y. Kashiwagi, E. Tsunoo and S. Watanabe, “Gaussian kernelized self-attention for long sequence data and its application to ctc-based speech recognition,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processin 2021, pp. 6214–6218.

[70] L. Tredoux, “End-to-end Automatic Speech Recognition of Code-switched Speech,” in arXiv preprint arXiv:2010.05549, 2023.

[71] G.I. Winata, A. Madotto, C.S. Wu and P. Fung, “Towards end-to-end automatic code-switching speech recognition,” in arXiv preprint arXiv:1810.12620 , 2018.

[72] C.T. Do, R. Doddipatla and T. Hain, “Multiple-hypothesis CTC-based semi-supervised adaptation of end-to-end speech recognition,” in ICASSP 2021-2021 IEEE International Confere 2021, pp. 6978–6982.

[73] E. Loweimi, A. Carmantini, P. Bell,S. Renals and Z. Cvetkovic, “Phonetic Error Analysis Beyond Phone Error Rate,” in IEEE/ACM Transactions on Audio, Speech, and Languag 2023.

[74] L. Gelin, T. Pellegrini, J. Pinquier and M. Daniel, “Simulating reading mistakes for child speech Transformer-based phone recognition,” in Annual Conference of the Internationa 2021.

[75] F. Eyben, M. Wöllmer, B. Schuller and A. Graves, “From speech to letters-using a novel neural network architecture for grapheme based ASR,” in 2009 IEEE Workshop on Autom 2009, pp. 376–380.

[76] A. Fang, S. Filice, N. Limsopatham and O. Rokhlenko, “Using phoneme representations to build predictive models robust to ASR errors,” in Proceedings of the 43rd Internation 2020, pp. 699–708.

[77] S. Kim, T. Hori, and S. Watanabe, “Joint CTC-attention based end-to-end speech recognition using multi-task learning,” in 2017 IEEE international conference on acoustics, speec 2017, pp. 4835–4839.

[78] T. Pellegrini and I. Trancoso, “Error detection in broadcast news asr using markov chains,” in Human Language Technology. Challenges for Computer Science and Linguistics: 4th 2011, pp. 59–69.

[79] N.J. De Vries, J. Badenhorst, M.H. Davel, E. Barnard and A. De Waal, “Woefzela-an open-source platform for ASR data collection in the developing world,” in 12th Annual Conference of the International Speech Communication Association (INTERSPEE 2011.

[80] L.A. Figueroa Suárez, and K.F. Pepper Loza, “Use of sequence connectors in the improvement of writing,” Bachelor’s thesis, Universidad de Guayaquil. Facultad de Filosofía, 2020.

[81] D. Jurafsky and J. H. Martin, Speech and Language Processing: An introduction to speech recog
3rd ed. Upper Saddle River, NJ: Prentice Hall: Stanford University, 2008.